Applied Artificial Intelligence

ISSN: 0883-9514 (Print) 1087-6545 (Online) Journal homepage: https://www.tandfonline.com/loi/uaai20

A PERSONALIZED TOURIST TRIP DESIGN ALGORITHM FOR MOBILE TOURIST GUIDES

Wouter Souffriau , Pieter Vansteenwegen , Joris Vertommen , Greet Vanden Berghe & Dirk Van Oudheusden

To cite this article: Wouter Souffriau , Pieter Vansteenwegen , Joris Vertommen , Greet Vanden Berghe & Dirk Van Oudheusden (2008) A PERSONALIZED TOURIST TRIP DESIGN ALGORITHM FOR MOBILE TOURIST GUIDES, Applied Artificial Intelligence, 22:10, 964-985, DOI: 10.1080/08839510802379626

To link to this article: https://doi.org/10.1080/08839510802379626

Published online: 20 Oct 2008.

Applied Artificial Intelligence, 22:964–985

ISSN: 0883-9514 print/1087-6545 online
DOI: 10.1080/08839510802379626

Taylor & Francis
Taylor & Francis Group

A PERSONALIZED TOURIST TRIP DESIGN ALGORITHM FOR MOBILE TOURIST GUIDES

Wouter Souffriau[1,2], Pieter Vansteenwegen[2], Joris Vertommen[2], Greet Vanden Berghe[1], and Dirk Van Oudheusden[2]
[1]*KaHo St.-Lieven, Information Technology, Ghent, Belgium*
[2]*Katholieke Universiteit Leuven, Centre for Industrial Management, Heverlee, Belgium*

□ *Mobile tourist guides evolve towards automated personalized tour planning devices. The contribution of this article is to put forward a combined artificial intelligence and metaheuristic approach to solve tourist trip design problems (TTDP). The approach enables fast decision support for tourists on small footprint mobile devices. The orienteering problem, which originates in the operational research literature, is used as a starting point for modelling the TTDP. The problem involves a set of possible locations having a score and the objective is to maximize the total score of the visited locations, while keeping the total time (or distance) below the available time budget. The score of a location represents the interest of a tourist in that location. Scores are calculated using the vector space model, which is a well-known technique from the field of information retrieval. The TTDP is solved using a guided local search metaheuristic.*

In order to compare the performance of this approach with an algorithm that appeared in the literature, both are applied to a real data set from the city of Ghent. A collection of tourist points of interest with descriptions was indexed and subsequently queried with popular interests, which resulted in a test set of TTDPs. The approach presented in this article turns out to be faster and produces solutions of better quality.

INTRODUCTION

Many tourists visit a region or a city for 1 or more days. It is not possible to visit every tourist attraction or cultural heritage site during such a limited period, so the tourist has to make a selection of what he believes to be the most valuable points of interest (POI). This personal selection is based on information found on websites, in articles, magazines, or guidebooks from specialized bookstores or libraries. Once the selection is made, the tourist decides on a route, keeping in mind the opening hours of the POIs and the available time.

Address correspondence to Wouter Souffriau, Information Technology, Department of Engineering, KaHo Sint-Lieven, Gebr. Desmetstraat 1, Gent B-9000, Belgium. E-mail: wouter.souffriau@Kahosl.be

Tourists face several difficulties when following that procedure. Information provided in guidebooks can be out-of-date, e.g., opening hours may change due to a newly elected city council. Also, guidebooks cannot provide temporal information: temporary exhibitions in museums change all the time, some attractions are (partly) closed due to renovation, and theaters change their program regularly (Dunlop et al. 2004). Tourists have to combine the information from different sources and decide which information is the most reliable. Moreover, selecting the most valuable attractions, i.e., those of the greatest interest to the tourist, is not easy. Usually tourists will be happy if they devise a somewhat attractive and feasible schedule, but they have no idea whether better schedules are possible.

Some guidebooks acknowledge these problems and propose generic visitor tours through a city or region. Of course these tours are constructed to satisfy the interests of the majority rather than the specific interests of individuals (Cheverst et al. 2000). Furthermore, tourists today want to maximize their free time and expect to be well informed on what a city or specific attraction can offer (Oppermann and Specht 1999; Keyson 2004).

Mobile tourist guides (MTG) are excellent aids for tourists who want real support for this kind of problem (Malaka and Zipf 2000; Cheverst, Davies, and Mitchell 2002). Based on an interest profile, up-to-date attraction information, and trip information, the MTG can suggest a (near-) optimal and feasible selection of attractions and a route between them (Vansteenwegen and Van Oudheusden 2007). Generic visitor tours do not take user context into account, e.g., the start and end location, available time, current time, weather, etc. Adding context and location awareness are practical challenges in developing intelligent mobile applications (Kröger and Malaka 2004). Kramer et al. (2006) have analyzed the diversity of gathered tourist interest profiles and conclude that they are surprisingly diverse. This conclusion supports the idea of creating personalized tours instead of proposing generic visitor tours. Moreover, most tourists today move within a limited crowded area of very attractive POIs. Kramer et al. (2006) state that an MTG equipped with personal selection and routing of attractions will help to spread tourists more evenly across the destination region.

This article addresses the very challenging problem of providing intelligent decision support in a fast and dynamic way. Using AI techniques, it matches tourist information regarding POIs (based on pure text descriptions) with the individual tourist's interest and generates a route through a selection of the most interesting POIs such that the total satisfaction is maximized, given certain limitations on time and distance.

STATE OF THE ART

Current state-of-the-art MTGs support the comparison of up-to-date information and allow the user to quickly navigate through related pieces of information. They help tourists in choosing their destinations, they provide the shortest routes between given destinations, etc. Previously developed MTGs are discussed and compared here.

One of the first attempts to develop an MTG was Cyberguide, which is a context-sensitive MTG (Abowd et al. 1997). It provides information to tourists based on position and orientation knowledge. The primary objective of Cyberguide is to provide some of the services expected from a human guide. Gulliver's Genie is a context-aware tourist guide that assists roaming tourists (O'Hare and O'Grady 2002). It takes into account the current position of tourists and their dynamically updated profile to provide personalized information. While visiting a city, Gulliver's Genie provides tourists with up-to-date information about the various sights in that place. The tool helps tourists to select the attractions they want to visit and to plan their trip throughout the city. After visiting an attraction, Gulliver's Genie guides tourists to the next selected attraction. The user profile is updated dynamically, based on what a tourist actually selects and does not select.

More recently, location-aware museum guides, such as HIPPIE (Oppermann and Specht 1999) and Hyper-Audio (Petrelli et al. 1999) and the mobile location-based fair guide SaiMotion (Hermann and Heidmann 2002) became available. HyperAudio provides visitors was personalized and location-based information for the tourist. HIPPIE and SaiMotion additionally offer the functionalities of guiding visitors around and helping them to make a selection.

GUIDE is an intelligent MTG (Cheverst et al. 2002). Like Gulliver's Genie and Cyberguide, GUIDE is a PDA-based system. Much more than Gulliver's Genie, GUIDE takes into account the user profile and makes it easy to select the attractions to visit. For a personal tour, however, tourists still have to select the places of interest themselves; GUIDE only provides a choice list. GUIDE enables creating a tour in a city taking into account the selected attractions, the preferences of the user, the opening times of attractions, the best time to visit attractions, etc. The tour can be changed dynamically, based on the actual time the tourist already spent at some attractions or on weather conditions. The user profile is also continuously updated, using information from previous trips. GUIDE was tested in the city of Lancaster and the user feedback provided a lot of useful information for further development (Davies et al. 2001). The primary reason for negative reactions was the effort required to enter personal preferences to create a custom-built tour.

DEEP MAP is a German research project that aims at developing the prototype of a digital personal MTG which integrates research from various areas of computer science (Malaka and Zipf 2000). The project focuses on geo-information systems, user modelling, user interfaces, etc., but or no little attention has been paid to composing optimal schedules for tourists. Creation of User-Friendly Mobile Tourism Services (CRUMPET) is a European project (Poslad et al. 2001) that deals with user modelling, interactive maps, and the possible benefits of mobile tourism services.

The commercial software of eNarro[1] uses standard predetermined tours to guide tourists in historic cities. Tourists select a route based on a thematic preference and the system guides the tourist. At tourist attractions, the user can browse information, view photographs, and listen to a narration.

The Dynamic Tour Guide (DTG) described in ten Hagen et al. (2005) is the first MTG that calculates personal tours on-the-fly. A tour is a collection of so-called Tour Building Blocks (TBB): sights, restaurants, etc. An ontology consists of a tree of concept classes and describes each TBB. Tour Building Blocks receive interest matching points by a semantic matching algorithm, which calculates the degree of similarity between the TBB and the profile of the user. Semantic matching compares the positions of the concepts of the profile in relation to the concepts of the TBB. If both concepts are equal, or if the interest of the tourist is more general than the one of the TBB, an interest matching point is received; if the interest of the tourist is more specific, a partial point is received, depending on the distance between the concepts in the tree. Furthermore, a TBB can be a partial member of a concept class, or it can be member of several concept classes; a concept class can also be synonymous, with another. After the assignment of interest matching points, an algorithm tries to construct a tour by maximizing the sum of the interest matching points within a given time frame. The algorithm keeps a list of candidate TBBs, sorted by use of gain, i.e., the interest matching point divided by the cost needed to visit the TBB. From the list, the TBBs are removed and inserted randomly in the tour until the available time runs out. It appears that a large amount of research effort is still required in order to devise efficient decision support techniques acceptable for tourists.

Based on these developments, it can be concluded that providing automated attraction selection and routing is an upcoming trend in mobile tourist applications.

PROBLEM DESCRIPTION AND APPROACH

In this article, the orienteering problem (Golden, Levy, and Vohra 1987) is the starting point for modelling the TTDP. In the orienteering

problem, a collection of locations is given, each with a score S_i that denotes its attractiveness, as illustrated Figure 1. The time $t_{travel,\ ij}$ needed for travelling from location i to j, is known for all locations. The goal of the problem is to maximize the sum of the scores of the selected locations, keeping the total time of the route between these locations under a given time budget T_{max} or the total distance of the route under a given distance budget D_{max}. The orienteering problem can be extended by introducing $t_{visit,i}$ for each location i, to take into account the time a tourist needs to visit the location.

The orienteering problem shows similarities to the binary knapsack problem, in which a selection from a set of items has to be made. Each item has a profit and a weight. The objective of the problem is to maximize the total profit of the selection. The total weight of the selection is bounded by the capacity of the knapsack (Martello and Toth 1990). Each location of the orienteering problem can be seen as an item, its score as the profit, and the available time budget as the capacity of the knapsack. The main difference is that the total weight in the knapsack problem is independent of the order of the selected items. In the orienteering problem, the total travelling time depends explicitly on the order of the selected locations, which increases the complexity of the problem significantly.

In order to solve the TTDP, the attractiveness (score) of each location should be determined first. In this approach, the attractiveness is derived from documents containing the full text descriptions corresponding with tourist attractions. Each tourist attraction is uniquely determined by its GPS coordinates and can have multiple documents associated with it. A tourist attraction can be visited in function of different fields of interest: e.g., one can visit a church from a religious or from a historical point of

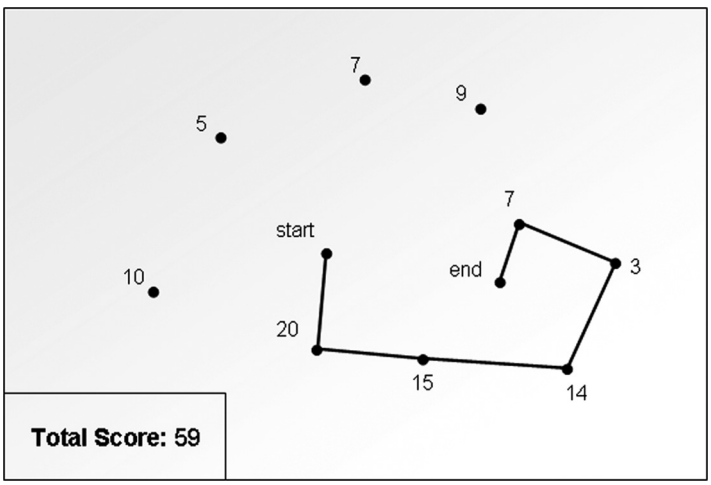

FIGURE 1 Orienteering problem.

view. For each field of interest of an attraction, a different text document is available. Based on the interests of the tourist and the description of the location, the attractiveness of the location is determined.

As this article does not focus on predicting personal visiting durations, zero visiting durations are assumed in what follows. Nevertheless, the proposed algorithm performs similarly if the visiting durations are not zero. The following section explains how the text documents are used to predict a personal score of the locations. A subsequent section describes a fast algorithm that integrates the selection of the attractions and the routing between them, to solve the TTDP.

PREDICTION OF INTEREST

This section describes how information retrieval techniques are applied to compute the score for each POI, based on the interests of the user. It is made possible by integrating a search engine in the tourist guide enabling the search for POIs. For that purpose, textual descriptions of the POIs are transformed to vectors by means of the so-called vector space model (Baeza-Yates and Ribeiro-Neto 1999). User interest is expressed by a number of keywords, which form a query that can be transformed into a vector in the same vector space of the POI vectors. Elementary linear algebra makes it possible to compare the query and the POI.

Document Preprocessing

All text descriptions of available POIs will be transformed to a vector space. This space is defined by the vocabulary used in the entire document corpus. Such a vocabulary can easily expand to several tens of thousands of words, resulting in a vector space with very high dimensionality. For computational, as well as qualitative reasons, the dimensionality is reduced in two steps.

The first step is to remove stopwords in the text. Stopwords are words such as “the,” “and,” “it” (in English). These words do not contribute to the essential content of a text, and thus cannot be used to represent that text. The second step involves stemming each individual word in the text. Stemming is the procedure in which a word is transformed to its base form. Several word forms are mapped to the same stem. The words “connect,” “connection,” “connected,” and “connecting,” for instance, all have the same basic meaning. A computer, however, does not know that. By mapping all listed words to their base form, the search engine will search for each of these terms when given one of them as input. Consequently, the search engine becomes more intelligent and is able to deliver qualitatively better results. Equally important is the vocabulary reduction of about 33%,

achieved by stemming. It reduces the vector space, and makes calculations in that space less demanding. The stemming process is commonly performed by the Porter stemming algorithm (Porter 1980).

Document Indexing

Once all the POI descriptions have been transformed to a vector with the stemmed vocabulary as dimensions, a weight will be assigned to each vector in each dimension. The higher the weight that is given to a dimension, the more important the stem is in describing the text contents.

The most widely recognized weighting scheme is TF-IDF (Salton 1989). It is composed of two parts:

- Term frequency (TF) tf_{ij} is the number of times term i occurs in text j.
- Inverse document frequency (IDF) incorporates the inverse of the number of texts in a corpus that a term i occurs in $(1/N_i)$.

Term frequency and IDF are integrated by the following function:

$$w_{ij} = \frac{tf_{ij}}{\|d_j\|_2} * \log\left(\frac{n}{N_i}\right)$$

with w_{ij} the weight given to stem i in text j, n the number of texts in the corpus, and d_j the document vector of which the elements are normalized to a value between 0 and 1 according to

$$\|d_j\|_2 = \sqrt{d_j \cdot d_j} = \sqrt{\sum_i w_{ij}^2}.$$

.This is to ensure that all the vectors have the same length, and can thus be compared on an equal basis. Using this scheme, a high weight is given to terms that appear frequently in only a few texts. Wide usage and extensive testing in the information retrieval research domain have shown that this presentation is favorable for information retrieval applications such as search engines.

Note that this part of the approach should be carried out before the tourist actually applies the tourist guide for decision support.

User Interest and Location Scores

As mentioned above, the user interest is formulated as a query, i.e. a series of keywords. The query is transformed to a vector in the vector space described by the POI vectors. Both stopword removal and stemming are

applied to the query as well. The dimensions used in the query vector are given a weight of 1, the others a weight of 0.

To calculate the interest of a user in a specific POI (personalised score of the location), the cosine similarity between the user query vector q and each POI vector j is calculated:

$$sim_{qj} = \cos(q, j) = \frac{q \cdot j}{\|q\| \cdot \|j\|}$$

Based on the calculated similarity, a list is produced that ranks POIs according to their correspondence to the user's query, and thus to his interests. Limiting this list to a prefixed number of POIs supplies the tourist guide with locations that may be of interest to him.

SELECTION AND ROUTING

The approach described in this section must be efficient and fast as it should provide the user with real-time decision support.

Instead of spending much time and memory looking for the optimal solution of a TTDP, metaheuristics can be used in order to find a good quality solution in a limited amount of time. The authors developed a guided local search (GLS) metaheuristic procedure to tackle the orienteering problem and some of its extensions (Souffriau, Vansteenwegen, Vanden Berghe, and Van Oudheusden 2006; Vansteenwegen, Souffriau, Vanden Berghe, and Van Oudheusden 2008). The algorithm applies a combination of different heuristics. Guided local search (GLS) is used to further improve the two basic heuristics that contribute most to achieving high quality results. GLS penalizes, based on a utility function, unwanted solution features during each local search iteration. Due the penalty, the probability for the algorithm to get stuck in a local optimum is decreased (Voudouris and Tsang 1999).

In the forthcoming paragraphs, the heuristics are briefly explained one after the other; it is shown how they are combined. The metaheuristic and its performance is explained in detail in Vansteenwegen et al. (2008).

Construct is a greedy construction heuristic that produces an initial tour in a short amount of time. It was first introduced by Chao, Golden, and Wasil (1996). Firstly, all locations that are outside the maximum radius of the trip are removed from the problem. Then, a fixed number of points (L) are selected, namely, those that are furthest from the start and end point. Each of the L tours is started by assigning one of these chosen points to the tour. Next, all the remaining points are inserted into these L tours by applying "cheapest insertion," which only looks at the distance, not at the score. If all

L tours exhaust the time budget while still unassigned points remain, new tours are constructed with these unassigned points (using cheapest insertion) until all the points are assigned. Finally the best tour, with respect to the score, is selected to start the remaining part of the algorithm.

TSP is a TSP heuristic using GLS. Finding the shortest route between the selected locations is not an explicit part of the orienteering problem objective. However, the shorter the tour, the more likely extra locations can be added to that tour. In order to find the shortest route "GLS with greedy local search" is implemented, which was first introduced by Voudouris and Tsang (1999). It is a GLS heuristic using a two-opt move, which is fast, easy to implement, and effective enough for this application.

Insert inserts an extra location in the tour. For every location not yet included, the heuristic determines the least time-consuming position in the tour to include it. If the time consumption of the location in that position is lower than the remaining time budget, the location is inserted in the tour. The order in which the locations are considered for insertion is not random. For every tour the locations are ranked in increasing order based on their Euclidian distance to the included nodes. If the remaining budget appears insufficient, the node is not included.

Replace is a location replacing heuristic which uses GLS. The replace location heuristic considers the nonincluded locations one by one. If the remaining budget is insufficient to include the considered location in a tour, all included locations with a lower score are considered for exclusion. If excluding a location from the tour creates enough time to insert the nonincluded location under consideration, the included node will be excluded, and the nonincluded node inserted. If no more replacements are possible, the heuristic reaches a local optimum.

In order to escape from this local optimum, two GLS penalties are used: increase the augmented score of the nonincluded location with the highest score and decrease the augmented score of an included location with a low score.

The utilities U_i of all nonincluded locations are calculated and the location with the highest utility is rewarded by increasing its score with p_R. In the next iteration, the probability that this location will be included is increased. The utility is calculated by

$$U_i = \frac{S'_i}{1 + nr_i}$$

with S'_i the augmented score of location i: $S'_i = S_i + nr_i * p_R$; nr_i the number of times this location was "rewarded" and p_R the magnitude of the reward used in the replace heuristic. The utility will be higher for locations with a higher augmented score. The term $1 + nr_i$ is introduced to prevent

the scheme from being completely biased towards rewarding high scoring features. If a feature is rewarded many times, this term will prevent other features from not receiving a reward.

The disutilities DU_i of all included locations are calculated and the location with the highest disutility is penalized by decreasing its score with p_P and removing it from the tour. In the next iteration, the probability that the location will be included again is decreased and thus more time budget becomes available for nonincluded locations. The disutility is calculated by

$$DU_i = \frac{1}{S'_i * (1 + np_i)}$$

with S'_i the augmented score of location i: $S'_i = S_i - np_i * p_R$; np_i the number of times this location was "penalized" and p_R the magnitude of the penalty used in the replace heuristic. In this case, since S'_i is part of the denominator, the disutility will be higher for locations with a lower score. The term $1 + np_i$ is used in the same way as above. Having applied these penalties and rewards, the replace location heuristic is used again during the next iteration step, but always with the augmented scores. The exact magnitude of the penalties depends on the average score of all the locations.

```
Construct;
Loop2 = 0;
while Loop2 < MaxLoop2 do
    Loop2++;
    Loop1 = 0;
    while Loop1 < MaxLoop1 and Solution improved do
        Loop1++;
        TSP;
        Insert;
        Replace;
    end
    if Solution better than BestFound then
        BestFound = Solution;
    else
        if Solution = BestFound then
            if Not Disturbed Before then
                Disturb;
            end
        end
    end
    if Loop2 = MaxLoop2 / 2 then
        Disturb;
    end
end
Return BestFound;
```

ALGORITHM 1 The GLS algorithm.

Disturb the current solution. In order to better explore the whole solution space, many metaheuristics implement diversification strategies. The disturb algorithm removes a part of each tour, in order to create an opportunity for other locations to enter the tour or to move from one tour to another.

Algorithm Structure: pseudo code 1 illustrates how the different heuristics are combined into one algorithm. The heuristics are implemented in two loops and disturb will always be used in this algorithm at least once, to guarantee a diversification of the search in the solution space.

VALIDATION OF THE APPROACH

This section describes a practical implementation of the system for the city of Ghent. Touristic data was provided by the company RouteYou[2] for 223 POIs. Each POI has one or more descriptions. In total, the collection contains 649 documents written in Dutch. Each description has a corresponding category label (e.g., restaurant, garden, station, etc.). The document collection has 112 different category labels in total. The company has categorized each category label into a tourist interest (e.g., culture, history, architecture, etc.). We identified 14 different interests in total.

This section illustrates the three steps required for the design of a personalized tourist trip. First, the details of the indexing process are described. Next, the resulting index is queried and a number of TTDPs for the city of Ghent are created. Finally, these TTDPs are solved by both the GLS and the DTG algorithm presented in ten Hagen et al. (2005). A detailed comparison between the two approaches is presented as well. All experiments have been performed on an Intel Pentium processor with a clock speed of 2 GHz and 1 GB of RAM, Ubuntu 7.04 as operating system (http://www.ubuntu.com), and JAVA 1.6 as programming environment (sun microsystems http://java.sun.com).

Indexing the Document Collection

The total number of words in the descriptions is 61,767, total size of the descriptions is 386,810 bytes. Together with the extra information, such as the corresponding longitude, latitude, name, category and interest, the total size of the data is 432,361 bytes. The longitude, latitude, interest, category, and text description properties will be indexed for searching. The longitude and latitude are numeric fields that need no preprocessing. But, interests, categories, and text descriptions are text fields that do require preprocessing. As preprocessing is an irreversible operation, the original text fields also need to be stored.

TABLE 1 Tourist Trip Design Problems-1

Longitude	Latitude	Score	Longitude	Latitude	Score
	Museum			Garden	
3.726605	51.053052	9	3.727153	51.042002	26
3.723504	51.057517	50	3.721677	51.049708	21
3.729534	51.059537	11	3.721280	51.056653	23
3.722212	51.056779	50	3.724479	51.053565	22
3.727639	51.056505	12	3.723504	51.057517	26
3.724702	51.038453	12	3.715673	51.059271	28
3.697937	51.051752	12	3.726668	51.041684	26
3.725564	51.053617	9	3.719539	51.057020	32
3.726616	51.053881	9	3.725977	51.042422	28
3.725977	51.042422	10	3.724743	51.055936	34
3.720015	51.056115	100	3.724984	51.055926	34
3.721339	51.056912	9	3.728708	51.060467	32
3.720718	51.054236	50			
3.722726	51.038081	12		Restaurant	
3.727153	51.042002	12	3.723388	51.049856	11
3.723051	51.053939	9	3.721341	51.041054	13
3.721280	51.056653	9	3.728547	51.055607	10
3.726668	51.041684	12	3.722212	51.056779	11
3.710378	51.035843	9	3.723490	51.057944	11
3.720171	51.053341	11	3.721909	51.057814	11
			3.719539	51.057020	100
	Factory		3.712995	51.060430	10
3.720943	51.053245	9	3.722082	51.056786	9
3.714485	51.059914	100	3.727369	51.041165	12
3.729534	51.059537	50	3.726089	51.050685	11
3.758528	51.028062	97	3.720289	51.055336	46
3.721339	51.056912	11	3.723093	51.057816	11
3.723803	51.057847	10	3.721791	51.054362	13
3.728458	51.051449	9	3.725688	51.057977	12
3.722130	51.057779	10	3.721522	51.057511	10
3.722436	51.055729	14	3.722047	51.057682	11
3.715673	51.059271	9	3.723752	51.057791	9
3.725393	51.057586	9	3.722076	51.056137	11
3.729202	51.048140	11			
3.725524	51.056837	10		Monastery	
			3.715588	51.059240	21
	Statue		3.727733	51.057995	24
3.715588	51.059240	33	3.723519	51.052165	21
3.726517	51.054305	23	3.728717	51.057201	24
3.723322	51.050209	23	3.724923	51.053651	24
3.726616	51.053881	23	3.725977	51.042422	24
3.722082	51.056786	29	3.719071	51.059264	27
3.727286	51.048883	25	3.721339	51.056912	22
3.710345	51.036801	30	3.727153	51.042002	26
3.726881	51.042541	25	3.715673	51.059271	21
3.721280	51.056653	23	3.728904	51.055523	22
3.724479	51.053565	28	3.728334	51.051944	24
3.715673	51.059271	24	3.736871	51.053535	26
3.725407	51.053424	24	3.726165	51.057419	24

(*Continued*)

TABLE 1 Continued

Longitude	Latitude	Score	Longitude	Latitude	Score
3.723136	51.053655	23	3.723697	51.057914	29
3.721047	51.056475	26	3.721929	51.056008	26
3.722790	51.058245	22	3.720171	51.053341	22
3.721268	51.054157	26			
3.727356	51.056746	24			

The indexer program makes use of Lucene,[3] which is an open-source text search engine. As the texts are available in Dutch, a Dutch version of the Porter Stemmer algorithm is used. The indexing took 8,300 ms and used 8 MB of working memory. The size of the resulting index file is 890,510 bytes, or about twice the size of the original data.

Note that it is not necessary to include the full text in the index file. The original text can be stored in the system's database and only the primary key has to be stored in the index file. By implementing the system in that manner, a database query has to be executed after the search in order to retrieve the original data corresponding to the points of interest. An alternative solution could involve storing a compressed version of the original text.

The indexing process is performed off-line and does not influence the speed of the tourist trip design algorithm. Also, new documents can easily be added to an existing index without the need for reindexing the complete set of documents.

In the DTG tour composition approach (Kramer et al. 2006) POIs have to be categorized manually. The solution is to define a common knowledge base, containing all possible terms, arranging relations like synonyms, and defining attributes. All existing sights of a city need to be assigned to main categories of interests, which have little in common. This puts an enormous workload on the administrator of the system and only terms belonging to the knowledge base can be used for querying. The automated indexing approach of this article allows any term to be used for querying and eliminates the need for categorizing the POIs; however, both can be used in a complementary way.

Generation of TTDPs

A number of TTDPs are generated by combining a set of queries, a set of cost budgets, and a set of start and end-points. For the experiments described in the following paragraphs, the index is queried with the

TABLE 2 Tourist Trip Design Problems-2

Longitude	Latitude	Score	Longitude	Latitude	Score
3.726196	51.050593	63	3.718787	51.029956	57
3.718787	51.029956	63	3.723504	51.057517	57
3.723933	51.053622	63	3.724923	51.053651	57
3.723504	51.057517	17	3.728918	51.051748	57
3.723322	51.050209	63	3.720115	51.056679	57
3.724923	51.053651	17	3.728500	51.057519	57
3.720558	51.053854	14	3.727865	51.055434	57
3.726616	51.053881	11	3.725759	51.054601	57
3.727559	51.052515	100	3.723136	51.053655	57
3.725407	51.053424	63	3.726756	51.054237	57
3.725759	51.054601	17	3.730669	51.054949	57
3.726756	51.054237	17	3.750000	51.040000	57
3.715588	51.059240	63	3.720975	51.054969	57
3.720491	51.041064	10	3.727286	51.048883	57
3.720975	51.054969	63	3.717787	51.030296	57
3.720718	51.054236	63	3.720815	51.054738	57
3.729111	51.052878	100	3.725939	51.041082	57
3.715673	51.059271	63	3.727418	51.049312	57
3.726605	51.053052	63	3.727733	51.057995	57
3.721677	51.049708	63	3.726605	51.053052	57
3.723519	51.052165	17	3.723519	51.052165	57
3.726517	51.054305	14	3.727153	51.057331	57
3.725564	51.053617	11	3.729520	51.041478	57
3.725753	51.041581	17	3.719989	51.031683	57
3.726243	51.057144	17	3.715559	51.033140	57
3.723051	51.053939	63	3.725564	51.053617	57
3.721280	51.056653	63	3.725753	51.041581	57
3.736871	51.053535	11	3.724984	51.055926	57
3.719543	51.053652	63	3.726243	51.057144	57
3.727977	51.050495	63	3.716228	51.031859	57
3.723636	51.036512	63	3.725471	51.056629	57
3.727725	51.052878	63	3.725976	51.056417	57
3.721047	51.056475	17	3.721047	51.056475	57
3.725976	51.056417	17	3.729776	51.056160	57
3.726406	51.053710	63	3.721268	51.054157	35
3.722790	51.058245	63	3.726406	51.053710	57
3.721929	51.056008	17	3.721929	51.056008	57
3.710378	51.035843	63	3.720696	51.049723	57
3.731532	51.038838	63	3.720545	51.052606	57
3.725366	51.057442	17	3.725366	51.057442	57
3.726590	51.049459	63	3.720913	51.054885	57
3.728145	51.050239	63	3.721699	51.055796	57
3.721699	51.055796	17	3.724743	51.055936	57
3.731435	51.048402	63	3.725532	51.044753	35
3.724743	51.055936	63	3.721339	51.056912	57
3.725532	51.044753	63	3.730075	51.056106	57
3.721339	51.056912	63	3.727381	51.047634	57
3.728458	51.051449	63	3.724479	51.053565	57

(*Continued*)

TABLE 2 Continued

Longitude	Latitude	Score	Longitude	Latitude	Score
3.727381	51.047634	17	3.726165	51.057419	57
3.725722	51.056998	26	3.726580	51.054281	57
3.724479	51.053565	17			
3.725524	51.056837	63			
3.725571	51.051643	26			
3.718669	51.057902	12			
3.722076	51.056137	63			

Dutch equivalent of the following keywords: "garden," "factory," "statue," "monastery," "restaurant," "museum," "architecture," and "historical." Each query takes 70 ms on average, including opening the index for reading. To improve the speed of retrieval, a number of index readers can be opened beforehand and maintained in a pool, so that they can be selected whenever a query is executed. In this way, the retrieval time is reduced to 50 ms on average. Preprocessing the query takes 15 ms on average.

Possible distance budgets are 8000, 10,000 and 12,000 meters. POIs used as starting and end-points are the railway station (longitude 3.710378, latitude 51.035843), a hotel (longitude 3.729866, latitude 51.050471), and a public car park (longitude 3.725977, latitude 51.042422). Four different combinations of these points are used: starting and ending at the railway station, starting and ending at the hotel, starting and ending at the public park, and starting at the railway station and ending at the hotel.

The generated tourist trip design problems are included in Tables 1 and 2. Each problem consists of a number of POIs that can be visited. Each POI is described by its longitude, latitude, and the score. The keywords that describe each problem are transformed into a query vector. The similarity between this query vector and each document vector in the index results in the score of the corresponding POI. This score is the result of the TF-IDF vector weighing scheme. Points of interest with a higher score are considered more relevant than POIs with lower scores. Points of interest with a score equal to zero are assumed not to be relevant and are discarded from the problem. A POI can be relevant for different queries and thus may appear in different problems. In this case the user's interest is defined by multiple keywords, the query vector will contain multiple dimensions, and more POIs will be relevant

Comparison GLS-DTG Algorithm

The generated TTDPs are used to evaluate the performance of the GLS metaheuristic procedure. For comparison, the DTG algorithm that

TABLE 3 Results Garden

		Score (–)			Distance (m)		
Start–End Location	Budget (m)	DTG	GLS	Improvement (%)	DTG	GLS	Improvement (%)
station–station	8 000	193	192	−0.52	7 942	7 607	–
	10 000	304	304	0	9 685	9 637	0.50
	12 000	322	322	0	10 645	10 420	2.11
station–hotel	8 000	304	304	0	7 550	7 433	1.55
	10 000	322	322	0	8 431	8 389	0.50
	12 000	322	322	0	8 431	8 389	0.50
hotel–hotel	8 000	332	332	0	7 937	7 889	0.60
	10 000	332	332	0	7 937	7 889	0.60
	12 000	332	332	0	7 937	7 889	0.60
parking–parking	8 000	332	332	0	7 669	7 655	0.18
	10 000	332	322	3.11	7 669	7 655	–
	12 000	332	322	3.11	7 669	7 655	–

appeared in ten Hagen et al. (2005) was also implemented. The DTG algorithm uses Microsoft MapPoint to calculate the distance between two locations using a graph of streets. This experiment uses the Manhattan distance as a fast approximation of the length of the route between two locations, for both algorithms. The resulting sequence of locations is then used as input to a point-to-point routing algorithm, which calculates the route on a graph of streets. In case the real distance would exceed the given distance constraint, the locations with the lowest score will be dynamically removed from the tour, in order to reduce the length.

TABLE 4 Results Factory

		Score (–)			Distance (m)		
Start–End Location	Budget (m)	DTG	GLS	Improvement (%)	DTG	GLS	Improvement (%)
station–station	8 000	118	109	−7.63	7 981	7 603	–
	10 000	223	232	4.04	9 812	9 812	–
	12 000	252	252	0	11 079	10 626	4.09
station–hotel	8 000	241	241	0	7 991	7 653	4.23
	10 000	252	252	0	8 675	8 336	3.91
	12 000	252	252	0	8 675	8 336	3.91
hotel–hotel	8 000	252	252	0	6 373	6 256	1.84
	10 000	252	252	0	6 373	6 256	1.84
	12 000	252	252	0	6 373	6 256	1.84
parking–parking	8 000	242	252	4.13	7 869	7 767	–
	10 000	252	252	0	8 107	7 767	4.19
	12 000	252	252	0	8 107	7 767	4.19

TABLE 5 Results Statue

		Score (–)			Distance (m)		
Start-End Location	Budget (m)	DTG	GLS	Improvement (%)	DTG	GLS	Improvement (%)
station–station	8 000	280	280	0	7 901	7 901	0
	10 000	407	409	0.49	9 948	9 992	–
	12 000	431	431	0	10 602	10 148	4.28
station–hotel	8 000	385	431	11.95	7 769	7 992	–
	10 000	431	431	0	8 693	7 920	8.89
	12 000	431	431	0	8 693	7 920	8.89
hotel–hotel	8 000	401	401	0	7 518	7 153	4.86
	10 000	407	407	0	9 918	9 918	0
	12 000	431	431	0	10 571	10 129	4.18
parking–parking	8 000	401	401	0	7 413	6 959	6.12
	10 000	431	407	5.90	9 730	9 941	–
	12 000	431	431	0	10 383	9 929	4.37%

The results of the two different algorithms for the entire set of problems are included in Tables 3 to 10. For each problem the scores resulting from the two algorithms are given and the improvement percentage of the GLS over the DTG algorithm is calculated. Also, the distances obtained by the two algorithms are presented. The distance improvement is only computed when the score remained unchanged. Table 11 presents statistics of both algorithms for each of the tourist interests. For each set of test problems, the average computational time is given for both algorithms ($time_{DTG,avg}$ and $time_{GLS,avg}$). Next, the average score and distance improvements are given for each set of test problems ($score_{imp,avg}$ and $distance_{imp,avg}$).

TABLE 6 Results Monastery

		Score (-)			Distance (m)		
Start–End Location	Budget (m)	DTG	GLS	Improvement (%)	DTG	GLS	Improvement (%)
station–station	8 000	170	170	0	7 960	7 960	0
	10 000	317	336	5.99	9 603	9 938	–
	12 000	385	407	5.71	11 342	11 790	–
station–hotel	8 000	359	360	0.28	7 684	7 983	–
	10 000	385	407	5.71	8 837	9 158	–
	12 000	407	407	0	10 341	9 158	11.44
hotel–hotel	8 000	361	381	5.54	7 944	7 542	–
	10 000	407	407	0	9 759	8 626	11.61
	12 000	407	407	0	9 759	8 626	11.61
parking–parking	8 000	359	381	6.13	7 034	7 475	–
	10 000	407	407	0	9 610	8 542	11.11
	12 000	407	407	0	9 610	8 542	11.11%

TABLE 7 Results Restaurant

		Score (–)			Distance (m)		
Start–End Location	Budget (m)	DTG	GLS	Improvement (%)	DTG	GLS	Improvement (%)
station–station	8 000	264	266	0.76	7 905	7 963	–
	10 000	312	322	3.21	9 376	9 398	–
	12 000	332	332	0	11 352	10 974	3.33
station–hotel	8 000	322	322	0	7 938	7 086	10.73
	10 000	332	332	0	9 514	8 720	8.35
	12 000	332	332	0	9 514	8 720	8.35
hotel–hotel	8 000	322	322	0	7 583	7 190	5.18
	10 000	332	332	0	9 158	8 819	3.70
	12 000	332	332	0	9 158	8 819	3.70
parking–parking	8 000	322	322	0	7 565	6 953	8.09
	10 000	332	332	0	9 140	8 581	6.12
	12 000	332	332	0	9 140	8 581	6.12%

Finally, the average computational time, score, and distance improvements are presented for the total set of test problems.

The value n represents the number of POIs that are considered interesting and ranges from 12 to 55. The scores of the GLS algorithm are on average 4.38% higher than the scores of the DTG algorithm. When both algorithms find an equal score, the distance of the GLS tour is on average 4.97% lower than the distance of the corresponding DTG tour. As described in ten Hagen et al. (2005), the DTG algorithm is stopped after 5000 ms. On average, the GLS algorithm is about 10 times faster than the DTG algorithm and provides higher quality solutions.

TABLE 8 Results Museum

		Score (–)			Distance (m)		
Start–End Location	Budget (m)	DTG	GLS	Improvement (%)	DTG	GLS	Improvement (%)
station–station	8 000	306	318	3.92	7 896	7 886	–
	10 000	394	405	2.79	9 651	9 869	–
	12 000	408	417	2.21	11 934	11 937	–
station–hotel	8 000	405	405	0	7 677	7 571	1.38
	10 000	405	405	0	7 677	7 571	1.38
	12 000	417	417	0	11 190	11 145	0.40
hotel–hotel	8 000	385	387	0.52	7 594	7 895	–
	10 000	396	396	0	8 336	8 269	0.80
	12 000	405	408	0.74	10 691	11 314	–
parking–parking	8 000	376	387	2.93	7 846	7 811	–
	10 000	396	405	2.27	8 817	9 869	–
	12 000	405	417	2.96	11 172	11 937	–

TABLE 9 Results Architecture

		Score (–)			Distance (m)		
Start–End Location	Budget (m)	DTG	GLS	Improvement (%)	DTG	GLS	Improvement (%)
station–station	8 000	1 004	1 232	22.71	7 954	7 984	–
	10 000	1 346	2 122	57.65	9 911	9 923	–
	12 000	2 201	2 635	19.72	11 992	11 982	–
station–hotel	8 000	1 951	2 236	14.61	7 912	7 915	–
	10 000	2 486	2 635	5.99	9 958	9 906	–
	12 000	2 749	2 749	0	11 723	10 896	7.05
hotel–hotel	8 000	2 258	2 293	1.55	7 514	7 231	–
	10 000	2 464	2 464	0	9 896	8 891	10.16
	12 000	2 464	2 692	9.25	9 896	11 987	–
parking–parking	8 000	2 008	2 350	17.03	7 948	7 927	–
	10 000	2 464	2 464	0	9 834	8 822	10.29
	12 000	2 464	2 600	5.52	9 834	11 978	–

For the two simplest test problems, "garden" ($n = 12$) and "factory" ($n = 13$), the two algorithms perform equally well with respect to execution time, score, and distance. For the problems containing up to 20 locations, the GLS algorithm performs slightly better than the DTG algorithm. An increase in average solving time is observed for the DTG algorithm. This can be explained by the exponential increase of the the size of the solution tree. The "statue" and "restaurant" test problems are considered fairly easy. In almost all the settings, all locations can be visited within the distance budget and thus the problem is reduced to minimizing the distance and solving a traditional travelling salesperson problem. The GLS algorithm

TABLE 10 Results Historical

		Score (–)			Distance (m)		
Start–End Location	Budget (m)	DTG	GLS	Improvement (%)	DTG	GLS	Improvement (%)
station–station	8 000	936	1 305	39.42	7 988	7 926	–
	10 000	1 162	2 028	74.53	9 992	9 933	–
	12 000	1 816	2 184	20.26	11 996	11 999	–
station–hotel	8 000	1 888	2 001	5.99	7 973	7 998	–
	10 000	2 166	2 190	1.11	9 999	9 973	–
	12 000	2 326	2 379	2.28	11 829	11 316	–
hotel–hotel	8 000	2 059	2 110	2.48	7 903	7 591	–
	10 000	2 173	2 190	0.78	9 493	9 495	–
	12 000	2 265	2 326	2.69	11 945	11 873	–
parking–parking	8 000	1 556	2 018	29.69	7 970	7 878	–
	10 000	2 030	2 225	9.61	9 968	9 990	–
	12 000	2 229	2 326	1.17	11 874	11 990	–

TABLE 11 Comparison of Results

Theme	Garden	Factory	Statue	Monastery	Restaurant	Museum	Architecture	Historical	Total
n	12	13	17	17	19	20	50	55	–
$time_{DTG,avg}$ (ms)	27	28	835	811	3 608	4 032	5 000	5 000	2 418
$time_{GLS,avg}$ (ms)	26	25	31	28	35	38	452	470	235
$score_{imp,avg}$ (–)	0.47%	0.05%	1.53%	2.45%	0.33%	1.53%	12.84%	15.83%	4.38%
$distance_{imp,avg}$ (m)	0.79%	3.34%	4.62%	9.48%	6.37%	0.99%	9.17%	–	4.97%

performs better in solving this problem (4.62% and 6.37%, respectively) due to the use of the 2-Opt move and the pruning strategy of the DTG algorithm. For the most complex test problems, "architecture" ($n=50$) and "history" ($n=55$), the solution tree of the DTG algorithm cannot be traversed anymore within 5 seconds. This results in a clearly better performance of the GLS algorithm: 12.84% and 15.83% is the total score improvement respectively.

CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

This article presents a new approach to calculate tourist routes in a dynamic context by combining techniques from the field of information retrieval with techniques from operational research. Documents related to physical locations are indexed using the vector space model. The index is queried afterwards, by calculating the similarities between the tourist's interests and the documents. The resulting locations and their personalized interest scores form the basis for formulating a tourist Trip design problem, in which the total score has to be maximized without exceeding a given distance or time budget. The authors also developed a metaheuristic procedure, based on guided local search, to tackle this optimization problem and to present the tourist instantly with a very high quality tour.

The approach has been tested on real tourist data of the city of Ghent. The test set was used to compare the authors' GLS-based meta-heuristic procedure with the DTG algorithm. In problems with only a few locations, both DTG and GLS approaches perform equally well. As the number of locations increases up to 20, the GLS algorithm performs slightly better with respect to the solution quality. For the most complex problems, with over 50 locations, the GLS algorithm clearly outperforms the DTG algorithm with respect to both solution quality and computation speed. Thus, the new procedure represents an important advancement towards

operational personalized tourist guides. The indexing process is automated and any term can be used for searching interesting POIs.

Future research will involve adding an integrated user profile to the system, exploring state-of-art information retrieval techniques, extending the routing and selection algorithm with time windows, and predicting the visiting duration of the POIs.

REFERENCES

Abowd, D., G. Atkeson, J. Hong, S. Long, R. Kooper, and M. Pinkerton. 1997. Cyberguide: A mobile context-aware tour guide. *ACM Wireless Networks* 3:421–433.

Baeza-Yates, R. and R. Ribeiro-Neto. 1999. *Modern Information Retrieval*. New York: Addison-Wesley.

Chao, I.-M., B. Golden, and E. Wasil. 1996. A fast and effective Leuristic for the orienteering problem. *European Journal of Operational Research* 38(3):475–489.

Cheverst, K., N. Davies, and K. Mitchell. 2000. The role of adaptive hypermedia in a context-aware tourist guide. *Communications of the ACM: Special Issue on Adaptive Web-Based Systems and Adaptive Hypermedia* 45:47–51.

Cheverst, K., N. Davies, K. Mitchell, A. Friday, and C. Efstratiou. 2000. Developing a context-aware electronic tourist guide: Some issues and experiences. In: *Proceedings of ACM CHI Conference on Human Factors in Computer Systems*, The Hague, The Netherlands.

Davies, N., K. Cheverst, K. Mitchell, and A. Efrat. 2001. Using and determining location in a context-sensitive tour guide. *IEEE Computer: Special Issues on Location Aware Computing* 34:35–41.

Dunlop, M., P. Ptasinski, A. Morrison, S. McCallum, C. Risbey, and F. Stewart. 2004. Design and development of Taeneb city guide - from paper maps and guidebooks to electronic guides. Technical report www.cs.strath.ac.uk/~mdd/research/publications/04dunlop_enter.pdf. last accessed 9/25/2008

Golden, B., L. Levy, and R. Vohra. l987. The orienteering problem. *Naval Research Logistics* 34:307–318.

Hermann, F. and F. Heidmann. 2002. User requirement analysis and interface conception for a mobile location-based fair guide. In: *Human Computer Interaction with Mobile Devices, Proceedings of the 4th Internationl Symposium Mobile HCI*, ed. F. Paterno, pp. 388–392. Berlin: Springer. Springer Verlag, London UK.

Keyson, D. 2004. An electronic mobile guide for Artis zoo. Technical Report, Intelligence in Products Group, Faculty of Industrial Design, Delft University of Technology, The Netherlands.

Kramer, R., M. Modsching, and K. ten Hagen. 2006. A city guide agent creating and adapting individual sightseeing tours based on field trial results. *International Journal of Computational Intelligence Research* 2(2):191–206.

Krüger, A. and R. Malaka. 2004. Artificial intelligence goes mobile. *Applied Artificial Intelligence* 18: 469–476.

Malaka, R. and A. Zipf. 2000. Deep Map - challenging it research in the framework of a tourist information system. In *ENTER*, Barcelona, Spain.

Martello, S. and P. Toth. 1990. *Knapsack Problems - Algorithm and Computer Implementations*. John Wiley & Sons. New York, NY, USA.

O'Hare, G. and M. O'Grady. 2002. Genie: A multi-agent system for ubiquitous and intelligent content delivery. *Computer Communications* 26:1177–1187.

Oppermann, R. and M. Specht. 1999. A nomadic information system for adaptive exhibition guidance. *Archives & Museum Informatics* 13:127–138.

Petrelli, D., E. Not, M. Sarini, O. Stock, C. Strapparava, and M. Zancanaro. 1999. Hyperaudio: Location-awareness + adaptivity. In: *The Extended Abstract of CHI99*, Pittsburgh, PA.

Porter, M. 1980. An algorithm for suffix stripping. *Program* 14(3):130–137.

Poslad, S., H. Laamanen, R. Malaka, A. Nick, P. Buckle, and A. Zipf. 2001. Crumpet: Creation of user-friendly mobile services personalized for tourism. In: *3G*. London. In Proc. Conference on 3E Mobile Communications technologies pages 26–29, 2001.

Salton, G. 1989. *Automatic Text Processing*. Boston: Addison-Wesley.

Souffriau, W., P. Vansteenwegen, G. Vanden Berghe, and D. Van Oudheusden. 2006, November. Multilevel Metaheuristics for the orienteering problem. In: *7th UE/Meeting on Adaptive, Self-Adaptive, and Multi-Level Metaheuristics*, University of Malaga, Spain.

ten Hagen, K., R. Kramer, M. Hermkes, B. Schumann, and P. Mueller. 2005. Semantic matching and heuristic search for a dynamic tour guide. In: *Information and Communication Technologies in Tourism*, eds. A. J. Frew, M. Hitz, and P. O'Connor, Vienna: Springer.

Vansteenwegen, P. and D. Van Oudheusden. 2007. The mobile tourist guide: An OR opportunity. *OR Insight* 20(3):21–27.

Vansteenwegen, P., W. Souffriau, G. Vanden Berghe, am D. Van Oudheusden. 2008. A guided local search metaheuristic for the team orienteering problem. *European Journal of Operational Research in Press*, doi:10.1016/j.ejor.2008.02.037.

Voudouris, C. and E. Tsang. 1999. Guided local search and its application to the traveling salesman problem. *European Journal of Operational Research* 113(2):469–499.

NOTES

1. http://www.enarro.com
2. http://www.routeyou.com
3. http://lucene.apache.org