Annales UMCS Informatica AI 4 (2006) 266-278

Annales UMCS
Informatica
Lublin-Polonia
Sectio AI

http://www.annales.umcs.lublin.pl/

Automated travel planning

Piotr Nagrodkiewicz[1], Marcin Paprzycki[2*]

[1] *Department of Mathematics and Information Science, Warsaw University of Technology, Plac Politechniki 1, 00-661 Warszawa, Poland*
[2] *Computer Science Institute, SWPS, Chodakowska 19/31, 03-815 Warszawa, Poland*

Abstract

This paper summarizes the current state of art in the domain of automated travel planning. Requirements for planning systems are identified taking into account both functionality and personalization aspects of such systems. A new algorithm that allows planning routes between any two locations and that utilizes combination of various means of transportation is discussed.

1. Introduction

One of the areas served by information technology is the travel domain. Within this domain, much effort is being paid to the problem of integrating Internet available information. For instance, finding transportation between travel origin and destination is nowadays expected to be combined with locating suitable hotels, restaurants, etc. To achieve this goal information already available on the Internet must be integrated into a homogenous system that allows its robust exploitation. In this paper we address only one of aspects of developing an integrated *Travel Support System* – preparation of travel plans that combine multiple means of transportation. For the up-to-date description of a travel support system that attempts at a complete information integration, see [1].

Currently, within the Internet, there exists a number of systems that provide access to timetables and/or allow performing queries for routes between two given locations. However, in most cases such systems are limited to only one specific mean of transportation (airplane, train, bus etc.). A person who plans a trip involving an airplane and a train must manually search within multiple systems to find connections that together constitute a sensible travel plan. Within such a system proposed connection must satisfy user-defined and interrelation-based constraints. Separately, there exists software that supports traveling by

[*]Corresponding author: *e-mail address*: mpaprzycki@swps.edu.pl

combining GPS technology and digital maps. This software is capable of searching for the shortest routes and can help while traveling [2]. Unfortunately, this software is of value only for "individual" travels that utilize means of transportation such as cars, motorcycles etc. In other words, while it may be fun to watch on a GPS based display how the train is moving across France; such a system will not help us find connection with another train or a bus that we have to catch in Dijon. Obviously, combining travel from a "generic" origin to a "generic" destination (defined e.g. as a city name or a zip code) becomes more complicated when it involves particular targets like a detailed home address or a hotel. In this case the complete trip may involve one or more bus connections from home to the airport, flight to a specific destination, a train to the center of the city and a tram to the hotel. This is exactly the type of travel scenario that we are interested in.

In the paper we proceed as follows. In the next section we summarize information about most important travel planning systems. We follow with a description of our proposed approach (in Sections 3-7). In Section 8 we discuss some open research issues that have to be addressed for the proposed system to become truly robust.

2. Travel Planning

2.1.Current research projects

While there exists a number of travel planning projects [3-14], two of them are of particular interest in the context of this paper. Both are a part of research conducted by Craig Knoblock and his collaborators at the University of Southern California. These projects are: *Heracles* and *Theseus* [13,14]. *Heracles* is a framework designed to support creation of information agents that provide means of gathering and integration of data for a particular domain. An example of applying this framework (described on project web pages) is a system called *Travel Assistant* that supports planning of business trips [10]. Provided with the origin and the destination of the proposed travel, it is able to recommend best suiting means of transportation (taxi, private car or airplane – note that this project is US-based and thus focuses on means of transportation most typical of US travelers). If the airplane is chosen, the system advises whether it is cheaper to go to the airport by a private car and to pay for parking (for the time of being away on the trip) or to take the taxi. Additionally, *Travel Assistant* presents the weather forecast for the time of arriving at the destination and selects the closest hotel to the destination airport. *Travel Assistant* is able to optimize the plan of travel against time or price. The *Theseus* project, on the other hand, allows mining information from sources available on the Internet – WWW pages, in particular. It is also capable of monitoring over time the state of a task and it detects and reacts to changes. *Theseus* is based on flow charts, so it can execute

queries in parallel and can also predict queries it will receive in the close future so it can start their execution in advance, thus increasing the speed of the whole system [4,5]. Mining of information is achieved through wrappers dedicated to each source of information.

2.2. Hierarchical task networks

When speaking about planning it is difficult not to mention *Hierarchical Task Networks* [15,16]. Here, the idea how to solve the travel problem is to divide getting from place A to place B into two or more tasks of getting from A to B through some intermediate locations. For instance, let us consider a person that wants to get from his or her home to work. This task might be scheduled into smaller subtasks: getting from home to the bus stop, getting from that bus stop to the bus stop next to work and finally from the last bus stop to the place of work. This is a very simple example, however often a very complicated net of (sub)tasks could be obtained. Generally, schedule is achieved by applying one of the predefined operators which involve conditions that specify precisely when they can be applied.

3. Planning algorithm – the idea

When planning a trip one must take into account a lot of information to achieve the desired goal. Since the space of possible solutions is extremely large, algorithm that is to be used to solve this problem has to proceed in such a way to reduce maximally the search space. We can start from an observation as to how people act when planning a trip. First, they try to find a 'general' route to the destination and once this "step" is completed they search for connections that satisfy the route that they have established. The proposed algorithm works in a similar manner (which also matches the general approach of *Hierarchical Task Networks*). In the *first step*, it finds out divisions of the travel into stages (where each of them has defined both the starting and end locations and a mean of transportation between these two locations.) In the *second step*, for each division algorithm searches for possible chains of real connections. (Let us clarify, that when we talk about real connections we mean those involving real means of transportation, e.g. a train or a bus; however, currently the system is not connected with any real-world travel resources; instead, a simulated world of travel has been created to test the proposed algorithm). Resulting proposals are presented to the user. Algorithm is executed once and finds as many proposals as possible (at this stage no optimization is performed). Choosing the cheapest offer (or the fastest way) will be made by sorting all proposals against the desired criteria. Finding out all solutions will take more time than a search devoted to optimize a single criterion, but will be generally faster than

performing multiple searches to satisfy multiple requests that the user is likely to make. Let us now describe the proposed algorithm in more detail.

4. Planning algorithm – step I

4.1. Conditions of dividing the trip into stages

As described above, the task of the first step of the algorithm is to prepare a set of divisions (into stages) of the complete travel. We require that each stage must have defined (1) both start and end locations, and (2) mean of transportation used between these two locations. What is more, we require that all stages must end where the next stage in the division begins. For the first stage in the division we require that it starts as close as possible to the actual beginning of the travel and for the last stage in the division to end as close as possible to the actual end of travel (in both cases we call distances between these locations: *unplanned distances*).

4.2. Generation of divisions

Let us now describe how divisions of travel into stages are generated. Here, we utilize the idea of each travel having one main stage, for example, a flight or travel by train, and apply the *Hierarchical Task Networks* approach. We create a division by finding out the main part first, and solving two subtasks recursively (from the start of the stage to the start of the main part; and from the end of the main part to the end of the stage). By doing so, we eventually obtain a complete set of divisions. We will construct more than one division, because there are many possibilities to choose the main stage (it is described in detail below). Recursive division ends at some level when the main part covers the whole distance being divided or when there is no possibility of further division. Having solved both subtasks it is possible to construct divisions for a given level – it is achieved by merging each of the divisions for the first subtask, the main stage and each of the divisions of the second subtask. Merging is possible only if (1) all stages satisfy conditions described before (geographic continuity of stages) and (2) the sum of unplanned distances on both ends of the result chain of stages is not greater than the distance between the start and end locations of the part being divided (since the division would increase the overall distance the plan is needed for).

4.3. Selecting accepted paths

When all divisions are ready it is possible to remove some of them. First of all, it may turn out, that some divisions are identical (this is a result of recursion and many possibilities at each level of it). Secondly, we remove all divisions that imply much longer distance than the rest of them. This is done by the means of

statistics. Each division implies a distance at least as long as the sum of lines joining ends of each stage (since the straight lines are the shortest possible routes). Divisions with the implied distance, for example, twice as long as the average one might be removed as very often they are of lesser quality (they may be zigzagged). Furthermore, divisions that start and/or end not exactly where the travel does (*unplanned distances*) should be removed if there are divisions that start and/or end exactly where the travel does (those without *unplanned distances*). We assume here that it is better for a user, if possible, to obtain only plans that start and end where he or she wants to. This assumption may need to be relaxed though, if we take into account discount airlines that fly to somewhat remote places. If plans do not start or end exactly where the travel does (for example, if there are no stations known to the system in the area), it would be an improvement to remove these divisions that have *unplanned distances* much longer than the other ones, once again this is achieved by means of statistics. Whenever removing possible paths it is assumed that it is done only if some arbitrary number of them will be left (e.g. 10) so we never reduce the number of possibilities too far − a relatively wide variety of proposals is guaranteed to be ready for the processing in the second step of our planning algorithm.

4.4. Establishing travel details

Let us now describe how the main stages are found. At first, proper means of transportation are chosen using dedicated rules. These rules may decide that it is, for example, too long to walk, just enough to take a taxi or too close to fly (but in another situation flying might be just fine, if there is an obstacle in the way that makes other possibilities impossible.) Generally, the rules decide which means of transportation would work in a given situation and they should tend to exclude means of transportation that would not produce any possibilities in the second stage of the planning algorithm that is described later. Subsequently, for each of the chosen means of transportation the start and the end of the stage are set. If the means of transportation has no stations (like walk or taxi) the stage covers the whole distance. If there are stations, at both ends we search for N stations lying closest to the start (or end). Once they are identified, the main stages are created between every possible pair of start-end stations (under condition that such a pair is not against the rules used to choose proper means of transportation; before rules where applied to a different pair of locations – it can turn out, for example, that one of the chosen closest stations lies on another island and hence it is not a wise choice any more.)

Rules applied to select specific means of transportation can be based purely on distance between the two given locations – different means of transportation are better for different distances. However, if some extra information is added to the coordinates of those two locations, more sophisticated rules may be

constructed. For instance, information on a continent and an island/continental part on which the given location lies, may allow bringing better rules in – normally an airplane would not be used for short distances, unless there is an obstacle in the way (such as a sea). Such extended information may be gathered automatically for most stations and towns [17].

Use of rules causes the system to search for connections that lead in a rather direct way to the target location. This is generally a recommended behavior, however, sometimes it might be undesirable. There might be only one complex solution that leads in a curve-like way to the target or there might be a very attractive price offer requiring such a route. Unfortunately, the planning system will not find it, as it will not even consider it. However, the presented algorithm allows receiving feedback that may be used to mediate such problems – it allows extending the list of stations searched for a given means of transportation in the first step of algorithm by some additional pairs of stations. These new stations are marked to be coupled only with each other, not with any other station. Such additional pairs of stations allow making the algorithm consider some cases it would have normally not investigated. In this way, for instance, we address the problem of discount airlines flying from Frankfurt Hahn rather than Frankfurt's main airport.

5. Planning algorithm – step II

In the first step of the planning algorithm divisions of travel into stages have been prepared. In the second step these divisions are being realized by real connections.

5.1. Implementation of divisions

Stages of a single division are completed one by one. We start from the beginning of the trip and conditions at the end of each stage specify entry conditions for the next stage. For instance, arrival time at the station establishes a set of possible trains. Of course, the first stage requires a special treatment, since there are no connections leading to it. Therefore a number of connections (e.g. 5) leaving from the start location is chosen as the initial search connections (each with different next station in its schedule; and for each of them the earliest one is chosen). Then the search is made until the connection arriving at the end location of the stage is found. Search is performed by simulating travel and checking possible changes at each visited station. Not all changes are checked as that would lead to browsing of far too many possibilities to perform the search in the acceptable time. Once again a set of dedicated rules is used to discard some of the potential changes to reduce the time spent on browsing of the search space. Rules are deciding whether a particular change should be ignored or not. They can do so taking into account various aspects: (1) These rules may easily

discard connections that definitively do not lead in the right direction. (2) They may also enforce keeping of direct connections (so no unnecessary search is performed if a connection leading to the destination is found.) (3) Another rule may state that it is not wise to change to the connection that have the same route but is a later one, since this gains nothing. (4) It is also pointless to change to a local train if the current location is very far away from the target destination. Other rules may be devised as well. It is obvious that the more sophisticated rules will be, the faster the search will be performed (as they will eliminate a larger number of spurious connections).

We have to acknowledge that not all means of transportation do have stations and hence they have no timetables (e.g. taxi). As a result, when stage with such a means of transportation is encountered we need to reserve a minimal required time for that stage, so that the start time for the next stage can be established. An exact duration of that stage will be known precisely when connections for the next stage will be found – it may turn out then, that there will be more time for the stage then previously reserved. It is crucial to reserve always ample amount of time for such stages, otherwise it may turn out that the whole plan will not be executable.

6. Planning algorithm – architecture

In Figure 1 we present architecture of the proposed planning system. The darkest arrows indicate a typical flow of information in the system: at first, for given locations, divisions of travel into stages are generated, subsequently these divisions are implemented with the real connections and eventually all the resulting proposals are being sorted according to expected user satisfaction (personalization).

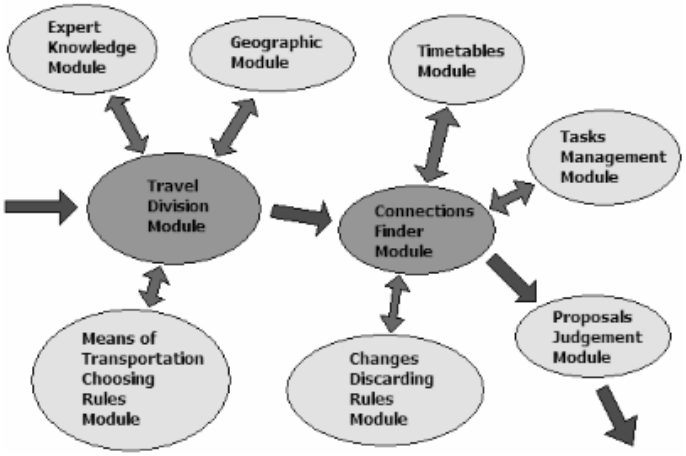

Fig. 1. Architecture of a travel planning system

Each of system modules might be encapsulated by a separate agent. These agents might be divided into groups that can operate on different computers, if there is such a need, for instance, for performance reasons. These groups are: agents working on travel divisions, agents finding connections and agents judging the prepared travel proposals. These groups should not be split as agents belonging to them exchange large amount of information (thus possibly generating a lot of network traffic).

6.1. Travel division and finding connections

Divisions of travel into stages are generated with utilization of information provided by the following modules: geographic module (extended location information and finding of stations situated close to the given location), means of transportation selection module (a rule based system that provides list of appropriate means of transportation) and expert knowledge module (responsible for providing feedback to the first step of the algorithm, as described above). The test implementation of the planning system utilizes JESS (*Java Expert System Shell*) as the rule system [18]. This allows the dynamic edition of rules without making any changes to the remaining code of the system and thus experimenting with different sets of rules. Prepared divisions of travel into stages are passed to the connection finder module for realization using the real connections.

Realization of each proposed path is achieved with help of the following modules: timetables module (provides connections passing through the given station; it might also manage a database of such connections), tasks management module (decides in what order changes are investigated) and changes discarding rules module (a rule-based system deciding whether to ignore the given change, also JESS-based). Realizations of all paths constitute proposals of travel routes prepared for the user. They are passed to the judging module and sorted so that the best of them (according to the system knowledge of user preferences) are going to be presented first.

6.2. Proposals judgment module

This is the last of main modules of the planning system. Its task is to judge already prepared proposals and sort them based on likelihood of being attractive to the user. Judging proposals might be done by using case-based logic. Each travel proposal is described as a single case containing its most important characteristics [19,20] (the more of them, the better). Cases, in turn, are stored in the *Case Retrieval Net* that allows fast retrieving cases similar to the given one. Furthermore, each case does not have to have the same descriptors as the other ones (e.g. one case can contain more visited stations while another case may contain details of a flight while yet another one may involve details of travel by

a train); the net will still work. With each case there is also associated a decision made by the user on the proposal it describes: it can be either acceptance (the user has chosen this proposal as his or her final plan) or the direct statement that a proposal is unacceptable (user has indicated that the given proposal is totally unacceptable to him or her). Each proposal being judged is transformed into a case, which, in turn, is compared against cases already stored in the net. The most similar case from the net is chosen. The user may influence the process of computing similarity between cases by defining the importance levels of each descriptor – this gives him or her opportunity to tune the mechanism, so it will take into account only criteria important to the user (with proper weights). Similarity of the most similar case to the judged one is taken as the judgment value (with a negative sign, if the decision associated with the most similar case was a rejection). In this way, each of the proposals is judged and then all of them sorted based on that judgment. Subsequently, the user browsing the list of proposals may reject some of them or choose one of them as a plan to be executed. In both cases an additional case is created and stored in the net along with the decisions made about it–this will extend the knowledge gathered in the net, hence over time the judging mechanism is able to learn preferences of the user–even such that would be very difficult to learn otherwise, like that in some regions the user prefers trains and in other buses.

A screen-shot of the system as it has been implemented is presented in Fig. 2. Instead of a GIS module (which was not available) we have implemented our own "world editor" that was used to generate worlds characterized by features that the proposed algorithm had to deal with (e.g. islands, and peninsulas introduce special complexities to the system).

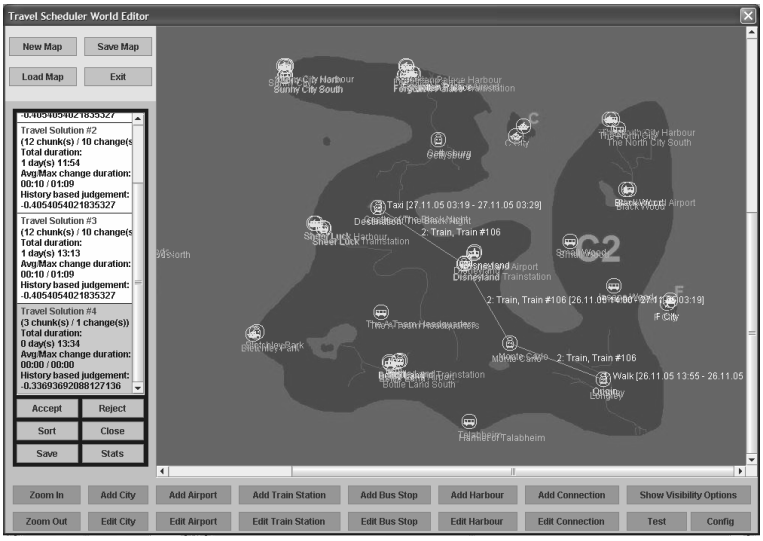

Fig. 2. Screen-shot of the system as it is currently implemented

7. Planning algorithm – additional issues

The optimal solution. In both steps of the algorithm the rule-based systems are used to reduce the search space as far as possible. They prune the search tree considerably, however, it may turn out that in some cases they prevent the algorithm from finding the most optimal solutions. This is an unavoidable trade off, since deciding not to use the rules would mean searching through an extremely large number of possibilities what would in turn made the planning process cumbersome, if not prevent it from running at all in any sensible time.

Planning with date of departure and date of arrival. As every trip planning system, algorithm presented here allows the user to issue travel queries that may be either given date of departure or date on which he or she would like to reach the destination location. From the technical point of view, there are only small differences in the algorithm in both cases – only time definitions of 'next' and 'previous' stations are being swapped. In the case of planning to arrive at the given date, the planning takes place from the end toward the beginning – stages of travel divisions are being implemented from the last to the first.

Parallel Computing. It is worth mentioning that the proposed algorithm can be easily parallelized. Even though applied rules reduce greatly the search space, it still may turn out that there will be a massive number of possibilities to check. Hence this feature might be important in a production environment. Possibilities of parallel computation exist in both steps of algorithm. In the first one, subtask divisions at all levels might be found in parallel as these are independent divisions. In the second step each change might be investigated independently and thus in parallel.

8. Future work

Let us now discuss directions that the existing algorithm (that has been implemented – see Fig. 2 for the screen-shot of its current interface – and is available from http://agentlab.swps.edu.pl) can be extended.

Planning trips between multiple destinations. The presented algorithm allows planning of a route between any two random points. However, sometimes it might not be enough. For example, a person may be interested in visiting more than just two places. In this case, the proposed algorithm could be used to find routes between any two of these places. The problem is how to establish order of visiting these places (assuming that it is up to the system to make such a suggestion). It is not a trivial task as it is a case of the Traveling Salesman Problem (TSP). A simple solution is to ask the user in what particular order he or she wants to visit all locations. The other one is to use one of the approximation algorithms known for the TSP.

Problems of Uncertainty in Time Reservations. When describing the second step of the planning algorithm it was mentioned that there is a need to reserve

some time when encountering stage(s) with no timetables (like a taxi) as it is not known how long these stages could last. While the idea to reserve some time for these stages is not so bad, the problem is that if the amount of time being reserved will turn out to be too short, the whole plan will become impossible to complete. On the other hand, reserving too much time would cause the plan to have unnecessary layovers. A simple solution to these problems is to utilize a distance measure and establish a "benchmark time" e.g. for a taxi use 15 minutes for 10 kilometers and then scale it linearly. Unfortunately, this approach has some serious disadvantages. First of all, how to establish the benchmark time (city or highway? day or night? etc.)? Second, when scaling, how to be sure that the real distance between the start and the end of the stage is the same as the length of the straight line between these two locations? It might be possible that between these two locations there is a river and the bridge is in some distance. Finally, how to take into account the fact that every day on a given street there is a traffic jam between 4pm and 6:30 pm? A better solution to that problem must be devised.

Planning without positions of stations. To join different means of transportation there is a need to know if stations are close to each other (can we switch from the bus to the train easily? do we need to change train stations – like in Paris or Vienna?). This is easy to determine only if geographic locations of stations are available. Without explicit joints between stations used in subsequent stages it is hardly possible to complete the plan. Assuming these joints could be somehow available, there still would be an unavoidable increase in the number of possibilities to check while searching, since only a smaller number of rules could be used. Of course, some simple geographic information could be implicitly gathered – for instance, the continent. But the name of island would still be a problem. Generally, geographic locations are crucial for the first step of the algorithm. They also influence the second step, but in theory it could work without them. Since geographic locations of all stations are not easily available at the moment, some research should be undertaken to try to devise some other means of finding out divisions of a travel into stages.

Increasing user interaction. Another interesting aspect of planning – connected to travel personalization – is increasing users' interactions with the system. This could be achieved either by letting them to modify the divisions prepared in the first step of the algorithm or by allowing modification of prepared proposals. Modifications of the first step of algorithm would allow the user, for example, to have a route leading through his favorite city (possibly to meet friends). From the algorithm point of view such modifications would be transparent. The only problem is a need to develop an intuitive graphical interface that would allow the user performing such modifications. Modifications to plans generated in the second step of the algorithm could allow the user to enforce the use of another connection, for example, if she decides the

algorithm gaives her not enough (or too much) time for some stages due to the time reservations or because she would like to have more time to have a stroll in the city. Modifications made to the prepared plans would, however, require re-planning of some parts of the route.

9. Conclusions

In this paper current the state of art concerning the automated planning of a travel has been described and some of their weaknesses have been specified. In response we have proposed an algorithm that allows preparing travel plans utilizing multiple means of transportation. This algorithm is not limited by distances between locations it is to plan the route for. Architecture of the solution allows easy modification and replacing modules responsible for different aspects of travel planning. Use of *Java Expert System Shell* allows modification and conducting experiments with any set of rules used to choose appropriate means of transportation when dividing the travel into stages, and to discard some solutions proposed when realizing these divisions with real connections. The proposed algorithm has been implemented and readers are invited to download it and experiment with. In the near future we plan to utilize it within the context of a travel support system described in [1].

References

[1] Gordon M., Paprzycki M., *Designing Agent Based Travel Support System* [accessed: 25 September 2005 http://www.cs.okstate.edu/~marcin/mp/cvr/research/ISPDC_2005.pdf
[2] *Automapa* http://www.automapa.com.pl/ [accessed: 20 July 2005]
[3] Strahan R., Muldoon C., O'Hare G.M.P., Bertolotto M., Collier R.W., *An Agent-Based Architecture for Wireless Bus Travel Assistants*. http://emc2.ucd.ie/publications/WIS03.pdf [accessed: 15 October 2004]
[4] Barish G., Knoblock C.A., *Learning value predictors for the speculative execution of information gathering plans*. http://www.isi.edu/info-agents/papers/barish03-ijcai.pdf [accessed: 10 September 2004]:
[5] Barish G., Knoblock C.A., *Speculative execution for information gathering plans* [accessed: 10 September 2004] http://www.isi.edu/info-agents/papers/barish02-aips.pdf
[6] Dillenburg J.F., Wolfson O., Nelson P.C., *The Intelligent Travel Assistant,* [accessed: 12 September 2004] http://www.cs.uic.edu/~wolfson/mobile_ps/ita02.pdf
[7] Dillenburg J.F., Wolfson O., Nelson P.C., *The Intelligent Travel Assistant* [accessed: 12 September 2004] http://www.cs.uic.edu/~wolfson/mobile_ps/ita02.pdf
[8] Kay M.G., Jain A., *Issues in Agent-based Coordination of Public Logistics Networks*. [accessed: 24 September 2004] http://www.ie.ncsu.edu/kay/pln/IETR02-01.pdf
[9] Kumar Praveen; Reddy Dhanunjaya; Singh Varun, *Intelligent transport system using GIS* [accessed: 25 September 2004] http://www.gisdevelopment.net/application/Utility/transport/pdf/164.pdf
[10] Knoblock C.A., Minton S., Ambite J.L., Muslea M., Oh J., Frank M., *Mixed-initiative, multi-source information assistants*. http://www.isi.edu/info-agents/papers/knoblock01-www.pdf [accessed: 10 September 2004]
[11] Stallard D., *Talk'n'Travel: A Conversational System for Air Travel Planning*. [accessed: 29 August 2004] http://acl.ldc.upenn.edu/A/A00/A00-1010.pdf

[12] Homb A., Mundhe M., Kimsen S., Sen S., *Trip-planner: An Agent Framework for Collaborative Trip Planning*. http://www.cs.wright.edu/people/faculty/mcox/mii/papers/manisha.pdf [accessed: 1 September 2004]
[13] *Heracles: Constrain-based Integration* http://www.isi.edu/info-agents/Heracles/ [accessed: 10 September 2004]
[14] *Theseus: Plan Execution* http://www.isi.edu/info-agents/Theseus/ [access: 10 September 2004]
[15] Nau D.S., *Automated Planning: Theory and Practice*. chapter 11: Hierarchical Task Network Planning [accessed: 1 November 2004] http://www.cs.umd.edu/~nau/cmsc722/notes-fall-2004/chapter11.pdf
[16] Nau D.S., *Ordered Task Decomposition: Theory and Practice [accessed: 1 November 2004]* http://prometeo.ing.unibs.it/sschool/slides/nau/nau1.ppt
[17] *Getty Thesaurus of Geographic Names* [accessed: 5 February 2005] http://www.getty.edu/research/conducting_research/vocabularies/tgn/
[18] *Java Expert System Shell* http://herzberg.ca.sandia.gov/jess/ [access: 5 September 2005]
[19] Peuret F., *Case-Based Travel Agent,* [access: 28 August 2004] http://www.cs.tcd.ie/publications/tech-reports/reports.99/TCD-CS-1999-69.pdf
[20] Waszkiewicz P., Cunningham P., Byrne C., *Case-based User Profiling in a Personal Travel Assistant*. [accessed: 29 August 2004] http://www.cs.usask.ca/UM99/Proc/short/WaszkiewiczP.pdf