Solving Travel Problems by Integrating Web Information with Planning 

David Camacho, José M. Molina, Daniel Borrajo, and Ricardo Aler

Universidad Carlos III de Madrid, Computer Science Department, Avenida de la Universidad n° 30, CP 28911, Leganés, Madrid, Spain
{dcamacho, molina, dborrajo}@ia.uc3m.es, aler@inf.uc3m.es

Abstract. The evolution of the Web has encouraged the development of new information gathering techniques. In order to retrieve Web information, it is necessary to integrate different sources. Planning techniques have been used for this purpose in the field of information gathering. A plan for information gathering is the sequence of actions that specify what information sources should be accessed so that some characteristicts, like access efficiency, are optimised. MAPWeb is a multiagent framework that integrates Planning Agents and Web Information Retrieval Agents. In MAPWeb, planning is not only used to integrate and to select information sources, but also to solve actual planning problems with information gathered from the Web. For instance, in an travel assistant domain, plans represent the sequence of actions an user has to follow to perform his/her trip. But also, each step in the plan informs the WebAgents which information sources should be accessed. In this paper we describe MAPWeb and study experimentally two information retrieval characteristics: the average number of solutions retrieved depending on the WebAgents used and the allocated time limit, and the number of problems solved (those travel assistant problems for which at least one solution was retrieved).

1 Introduction

Traditional information retrieval tries to extract documents from a database related to a user query [10]. However, the evolution of the Web has encouraged the development of new information gathering techniques. Information in the Web is usually more structured than a simple document collection. Not only documents in the Web display more structure, but different information sources contain different kinds of information. Therefore, in order to solve user queries, it is necessary to integrate the different information sources. In other words, information gathering intends to integrate a set of different information sources with the aim of querying them as if they were a single information source [6, 7]. Also, because of the amount of information, efficiency considerations become very important. For instance, it is very useful to select which information sources will be queried. In order to both integrate and select the relevant information sources different techniques can be used. A frequently used technique is planning [1,3]. In that case, every step in the plan represents an action to query an

information source. Figure 1 summarizes the discussion above about traditional information retrieval and the new WEB information gathering techniques.

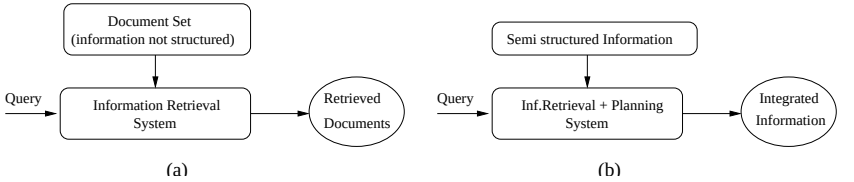

Fig. 1. (a) Traditional information retrieval and (b) plan based WEB information gathering.

There exist different actual systems for WEB information retrieval, the most close to the multi-agent system (MAS) presented in this paper is Heracles [9], in this case, instead of classical planning, a dynamic, hierarchical constraint propagation network is used the integration of the different information sources. Two assistant systems have been implemented: *The Travel Planning Assistant* (specialized in assisting tourists to plan their trips) and *The WorldInfo Assistant* (for a user-specified location, it integrates information from different information sources like weather, news, holidays, maps, airports, ...).

However, the information distributed in the WEB is heterogeneous in both content and format. This makes difficult to integrate different information sources to answer user queries. It would be better if all sources shared the same language so that standard techniques like planning could be directly applied. In order to solve this problem, wrappers are commonly used [2]. Furthermore, flexibility can be improved by having agents which are specialized in particular information sources [8].

In this paper we describe and study empirically MAPWEB [5] from the point of view of its information gathering skills. MAPWEB is a MAS which combines different kinds of agents that cooperate to solve problems. In this paper, the MAPWEB framework is applied to a travel planning assistant domain (e-tourism [4]),[1] where the user needs to find a plan to travel among several places. Each plan not only determines what steps the user should perform, but also which information sources should be accessed. For instance, if a step is to go from A to B by a plane of a given airline, then it is also known that the WEB server of that airline has to be accessed for further flight information.

MAPWEB will be analyzed empirically by taking into account its information retrieval abilities. Basically, two aspects will be studied: the number of solutions recalled depending on the time allocated to the system and agent topology, and the number of actual problems solved using these retrieved solutions.

This paper is structured as follows. First, MAPWEB architecture will be described. Three main points will be addressed: MAPWEB MAS based archi-

[1] This domain is a modified version of the Logistics domain.

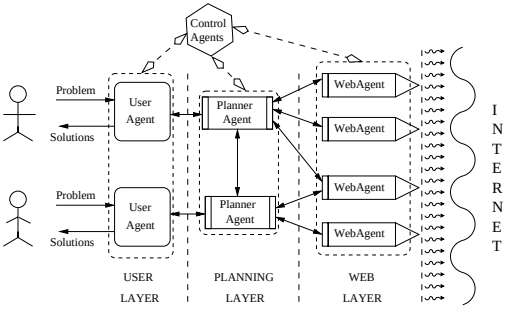

Fig. 2. MAPWeb three-layer architecture.

tecture, how planning is used by MAPWeb to gather information and to solve problems, and the role of the specialized WebAgents. Next, the experimental results about recall will be shown. Finally, conclusions will be summarized.

2 MAPWeb System Architecture

As Figure 2 shows, MAPWeb is structured into three layers: the *user layer* (which contains UserAgents), the *planning layer* (with the PlannerAgents), and the Web *access layer* (made of specialized WebAgents). Next, each one of the three types of agents will be described:

- **UserAgents**: They pay attention to user queries and display to users the solution(s) found by the system. When an UserAgent receives problem queries from the users, it passes them to the PlannerAgents and when they answer back with the plans, they provide them to the user.
- **PlannerAgents**: They receive an user query, build an abstract representation of it, and solve it by means of planning. Then, the PlannerAgents fill in the information details by querying the WebAgents. The planner that has been used for this work by the PlannerAgents is PRODIGY4.0 [11].
- **WebAgents**: Their main goal is to fill in the details of the abstract plans obtained by the PlannerAgents. They obtain that information from the Web.

In addition, MAPWeb contains **ControlAgents** to handle several control functions like the insertion and deletion of agents in the system and communication management.

3 The PlannerAgents

The goal of MAPWeb is to deal with problems that require planning to be solved and access to Web information to validate and complete the plans obtained previously. For instance, as described in Section 1, in the e-tourism domain [4] if a step in a plan is to go from A to B by plane, then it is also known

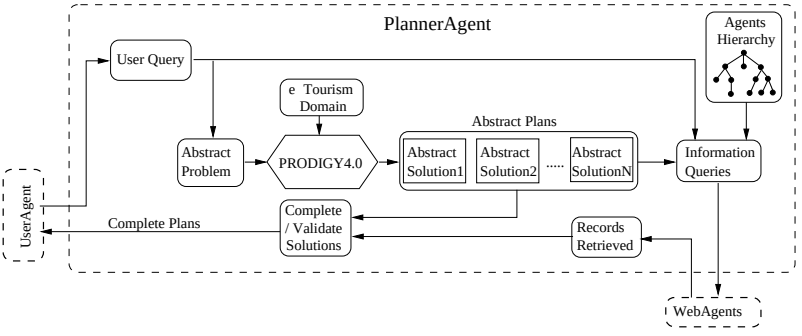

Fig. 3. Planning Process developed by the PlannerAgent.

that a WEB server with flight information must be accessed to complete the details for that step.

MAPWEB decouples the problem solving task by splitting it between the PlannerAgents and the WebAgents. Figure 3 displays the inner workings of a PlannerAgent.[2] The planning process works as follows. First, the PlannerAgent receives a query from any UserAgent. This query is analyzed and translated into an abstract planning problem by removing some details from the user query. Second, the PlannerAgent uses a planner (PRODIGY4.0 [11]) and tries to solve it. If the solving process is successful, the PlannerAgent generates a set of abstract solutions (abstract plans). However, the solutions obtained (abstract plans) lack the details. Therefore, each step in the plans has to be completed. To do so, the PlannerAgent builds a set of information queries for every step in every abstract plan. In order to build the queries, some of the details supplied by the user that were removed previously are also required. Then, each query is sent to the most appropriate WebAgent according to hierarchy. This hierarchy classifies the different WebAgents into categories, depending on the kind of information sources they are able to access. Finally, the PlannerAgent integrates all the specific information with the abstract solutions to generate the final solutions that will be sent to the UserAgent.

To illustrate the previous process, let us suppose that an user wants to travel from Madrid (airport) to Barcelona (train station). After removing unnecessary details, an abstract problem would be generated and subsequently solved. One of the possible abstract plans is shown in Figure 4.

Each step in the plan of Figure 4 would generate a set of information queries. The set corresponding to the `travel-by-airplane` would be sent to WebAgents that can retrieve flight information. Likewise for the `move-by-local-transport` set of queries.

[2] A more detailed description of this process can be found in [5].

```
Abstract plan:

<travel-by-airplane user1 plane0 airport0 airport1>
<move-by-local-transport user1 lbus0 bustop1 trainstat1 city1>
```

Fig. 4. Abstract solutions generated by PRODIGY4.0 for Leg 1 with 0-Transfers.

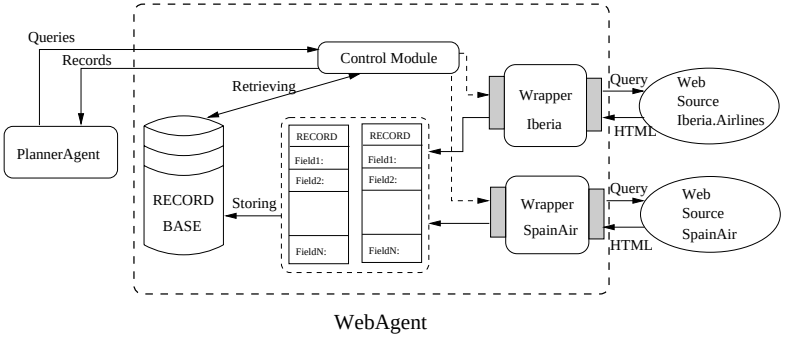

Fig. 5. WebAgents Architecture.

4 The WebAgents

The purpose of the WebAgents is to receive queries from the PlannerAgents, transform them into actual WEB queries, query the WEB source associated with the WebAgent, and return the information to the PlannerAgent in a standard format. The translation of the information in the WEB, which can be stored in many different formats, into a common format is performed by Wrappers [2].

Figure 5 displays the WebAgent architecture. A WebAgent is made of three main components: a control module, a record database, and one or several Wrappers. When a WebAgent receives a query from a PlannerAgent, its control module tries to fulfill the query by retrieving the appropriate records from the database. If there is no such record, the WebAgent access its WEB sources through the Wrappers. The Wrappers return a set of records which are first stored in the database, and then returned to the PlannerAgent.

For instance, the queries in the previous subsection would be answered by the WebAgents with the three records shown in Table 1.

Finally, the PlannerAgent receives all the retrieved information from the different WebAgents and uses these templates to instantiate the abstract plans. Those plans in which one or several steps received either no answer or an empty answer are rejected. Therefore, only fulfillable plans are sent back to the UserAgent. Every abstract plan will be instantiated into many different actual plans.

Table 1. Retrieved WebAgent Information.

Information-Flights	flight1	flight2	flight3
air-company	Iberia	Iberia	Spanair
http-address	www.iberia.es	www.iberia.es	www.spanair.com
flight-id	323	450	null
ticket-fare	38200	21850	43700
currency	ESP	ESP	ESP
flight-duration	null	null	null
airport-departure-city	MAD	MAD	MAD
departure-date	03-06-01	03-06-01	03-06-01
airport-arrival-city	BCN	BCN	BCN
return-date	06-06-01	06-06-01	06-06-01
number-of-passengers	1	1	1

5 Experimental Setup and Results

The aim of this section is to carry out several experiments with MAPWeb to evaluate its performance. The main aspect we want to evaluate in this paper is the number of plans (solutions) retrieved depending on the allowed time limit and the set of WebAgents used. To achieve this goal we followed the following steps:

- Three categories of problems were considered: national trips (within Spain), European trips, and Intercontinental trips. 10 planning problems were randomly generated for each category.
- Then, all the possible combinations of WebAgents were generated and tested by considering four specialized WebAgents[3]: Amadeus-Flights (AMF), 4airlines (4AL), Iberia (IBE), and Avianca (AVI). Amadeus-Flights and 4airlines are metasearchers: they can search information from many airplane companies. Iberia and Avianca can only search information about their own flights.
- Finally, we have plotted the number of solutions retrieved depending on the allocated time limit for the combination that retrieves the largest number of plans.

Table 2 shows the number of problems solved for the topologies that use only an isolated WebAgent, and the problems solved for the best topology tested for the different possible WebAgents combinations. As it is shown in Table 2 the best isolated WebAgents are the metasearcher engines (AMF and 4AL) given that these agents are able to retrieve information from different companies.

Table 2 shows that using different specialized WebAgents is useful to solve more problems: the best single agent configuration (AMF) solves 15 problems of 0 transfers and 25 of 1 transfer, whereas the four agent configuration (AMF-4AL-IBE-AVI) manages to solve 29 out of the 30 problems. However, adding more agents (3 and 4 agent topologies) is not always beneficial with respect to solving problems: a simple 2 agent configuration already fulfills 17/28 problems.

[3] *Amadeus:* http://www.amadeus.net, *4airlines:* http://www.4airlines.com, *Iberia:* http://www.iberia.com *Avianca:* http://www.avianca.com.

Table 2. Number of solved problems (out of 30) using different topologies.

Topology type	Selected topology	Problem type 0 Transfers	1 Transfer
1 WebAgent	AMF	15	25
	4AL	11	24
	IBE	5	14
	AVI	2	2
2 WebAgents	AMF-4AL	17	28
	AMF-IBE	17	28
3 WebAgents	AMF-4AL-IBE	17	28
4 WebAgents	AMF-4AL-IBE-AVI	18	29

But larger topologies usually find more solutions per problem, which can be useful if the user wants to choose a solution from a set of many possible plans according to some personal preferences (like travel time, cost, etc.). This can be seen in Figure 6, which shows the average number of plans per problem found for each of the three categories described in the experimental setup.

6 Discussion and Conclusions

Three main points will be addressed: the total number of solutions found, the time required to obtain them, and finally the effect of integrating complementary information sources.

Number of solutions: Figure 6 shows that there is a large difference between the 0-transfer and 1-transfer problems: the later always returns many more different solutions. This is due to existing many more indirect flights than direct flights.

Time required to obtain the maximum number of solutions (transient time): it can be seen that for the three categories (national, European, and international) of 0-transfer problems, this time is usually small (less than 3 minutes). The reason is that when only 0 transfers are considered, the number of queries and the number of retrieved solutions is small. On the other hand, 1-transfer problems require at least 5 minutes to retrieve all the solutions. This is because many more solutions are found. Also, this time increases from the national problems (5 minutes) and the European problems (10 minutes), to the international problems (15 minutes). In this case, the cause is not the higher number of solutions (there are fewer solutions for international problems) but the high number of Web queries. WebAgents send more queries for international problems because many more cities are involved. Fewer solutions for international flights are found because some queries return no solution, or redundant solutions are found.

Finally, the more WebAgents there are in the system, the more solutions are found. It is remarkable that when two 2-WebAgent configurations are combined into a 3-WebAgent configuration (i.e. from AMF-4AL and AMF-IBE to AMF-4AL-IBE), the number of solutions of the 3-WebAgent configuration (AMF-4AL-IBE) is usually higher than the sum of solutions of the two 2-WebAgent configurations (AMF-4AL and AMF-IBE), even though many solutions found

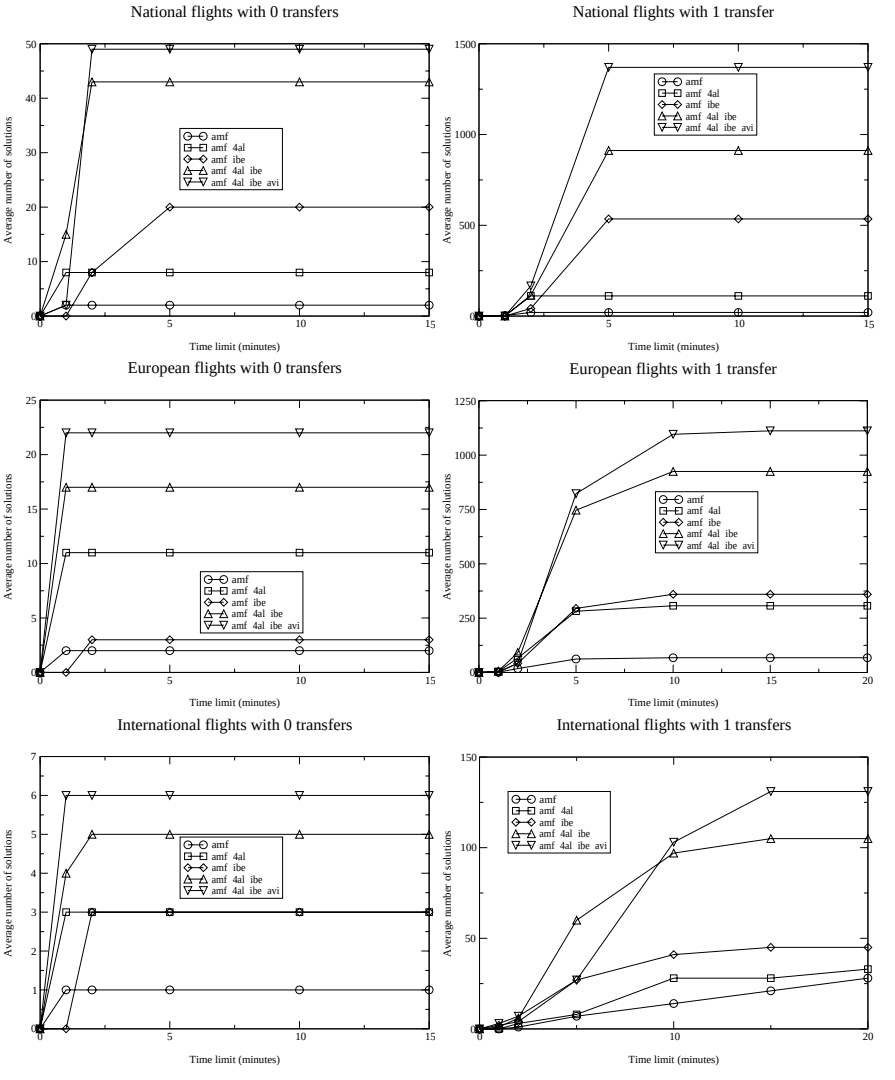

Fig. 6. Average number of solutions for 0 and 1 transfers.

by AMF-4AL and AMF-IBE might be the same. This is because new solutions are found by integrating the three information sources. Also, introducing new WebAgents in the configuration does not significantly increase response time even though there is a single PlannerAgent to process all the solutions found by all the WebAgents.

So far, we have only used WebAgents for airplane companies. However, MAPWeb is very well suited for integrating information coming from heterogeneous Web sites (like taxi, bus, trains, etc.). In the future we plan to integrate

these kind of sources, so that new solutions are achieved, which are not usually obtained by traditional travel WEB applications.

Acknowledgements. The research reported here was carried out as part of the research project funded by CICYT TAP-99-0535-C02.

References

1. Ambite, J.L., Knoblock, C.A.: Flexible and scalable query planning in distributed and heterogeneous environments. Proceedings of the Fourth International Conference on Artificial Intelligence Planning Systems. Pittsburgh, Pennsylvania (1998).
2. Ashish, N., Knoblock, C.A.: Semi-automatic Wrapper Generation for Internet Information Sources. Second IFCIS Conference on Cooperative Information Systems (CoopIS).Charleston, South Carolina.1997.
3. Bergmann, R., Wilke, W.: PARIS: Flexible plan adaptation by abstraction and refinement. ECAI (1996) Workshop on Adaptation in Case-Based Reasoning. Ed. by A. Voss et al. Published by John Wiley & Sons, Ltd.
4. Camacho, D., Borrajo, D., Molina, J.M.: Intelligent Travel Planning: A MultiAgent Planning System to Solve Web Problems in the e-Tourism Domain. International Journal on Autonomous Agents and Multiagent Systems. Vol. 4, num. 4, pp. 385-390, 2001.
5. Camacho, D., Borrajo, D., Molina, J.M., Aler,R.: Flexible Integration of Planning and Information Gathering. European Conference on Planning (ECP-01). September, 2001. pp. 73-84. Springer-Verlag. Toledo (Spain).
6. Fan, Y., Gauch, S.:Adaptive Agents for Information Gathering from Multiple, Distributed Information Sources. Proceedings of 1999 AAAI Symposium on Intelligent Agents in Cyberspace, Stanford University, March 1999.
7. Lambrecht, E., Kambhampati, S.: Planning for Information Gathering: A tutorial Survey. ASU CSE Techincal Report 96-017. May (1997).
8. Knoblock, C.A., Ambite, J.L.: Agents for Information Gathering. In Software Agents. Ed. by AAAI/MIT Press. Menlo Park, CA. (1997).
9. Knoblock, C.A., Minton, S., Ambite, J.L., Muslea, M., Oh, J., Frank, M.: Mixed-Initiative, Multi-source Information Assistants. The Tenth International World Wide Web Conference (WWW10). ACM Press. May 1-5. (2001).
10. Salton, G., and McGill, M. J.: Introduction to Modern Information Retrieval. McGraw-Hill Computer Science Series. New York: McGraw-Hill.1983.
11. Veloso, M., Carbonell, J., Perez, A. Borrajo, D., Fink, E., Blythe, J.: Integrating planning and learning: The Prodigy architecture. Journal of Experimental and Theoretical AI. Volume 7 (1995) 81–120.