ABSTRACT PLANNING IN DYNAMIC ENVIRONMENTS

DAVID CAMACHO, DANIEL BORRAJO, JOSÉ M. MOLINA, RICARDO ALER

Computer Science Department, Universidad Carlos III Madrid
Avda. Universidad nº 30, 28911 Leganés (Spain)
{dcamacho, dborrajo, molina}@ia.uc3m.es, aler@inf.uc3m.es

Abstract

Solving problems in dynamic and heterogeneous environments where information sources change its format representation and the data stored along the time is a very complex problem. In previous work we have presented a system called ***MAPWeb*** (***M**ulti**A**gent **P**lanning in the **W**eb*) that tries to solve those problems by integrating artificial intelligence planning techniques within the MultiAgent framework. Basically, *MAPWeb* allows cooperative work between planning agents and Web agents. The purpose of *MAPWeb* is to find solutions to travel problems. In order to give detailed solutions, *MAPWeb* uses information gathering techniques to retrieve travel information that is made available by many different companies. However, Web access to the information sources is quite time expensive. In this paper, we try to minimize the number of Web queries by using caching techniques based on relational databases. Experimental results show that the reduction in Web access time is quite important, while maintaining the number of solutions found.

Keywords
Multiagent Systems, Intelligent Agents, Planning, System Architecture.

1 Introduction

Currently, there is a vast (and increasingly growing) amount of information stored in Internet, especially since the development of the Web. This information is difficult to handle because it is heterogeneous, dynamic and distributed in nature. However, there are very few approaches that try to integrate a set of different and specialized information sources and reuse the data retrieved to solve problems [1, 2]. This is especially true if the goal of the system is to solve complex problem solving tasks, like finding complete travel plans by gathering information available on the Web.

In order to use Web information to solve complex problems, a framework called *MAPWeb* (***M**ulti**A**gent **P**lanning in the **W**eb*) has been developed and applied to solve travelling problems [6]. The key aspects of *MAPWeb* are:

- Solving complex problems requires using intelligent software components. *MAPWeb* integrates AI planning techniques with information gathering techniques. It actually divides the planning problem into two processes: abstract planning (performed by an AI planning system) and plan completion and validation (performed by Web gathering agents). This makes the planning problem more tractable, at least in some domains. On the other hand, using AI planning makes our approach very flexible.
- *MAPWeb* is a Multiagent System (MAS) [3, 9, 11]. This is very appropriate for our purposes because of MAS flexibility. First, it is easy to divide the task into different agents, each one containing the most appropriate skills. Thus, *MAPWeb* has PlannerAgents and WebAgents. Second, it faces the heterogeneity of Web sources by having different specialized Web agents. Third, the Web is a dynamic environment (Web sources are added/removed daily). To face this problem, *MAPWeb* takes advantage of the flexibility of MAS approaches for adding/removing agents to the agent society (besides the flexibility offered by classical AI planning)

However, *MAPWeb* (and any Web gathering system) must send many Web queries in order to access the desired information. This is usually costly in terms of time. Besides, in some cases a Web source might be temporally unavailable. In order to improve *MAPWeb* performance, learning techniques can be used in two ways: adding CBR skills to the PlannerAgents to store abstract plans [8] and adding caching skills to the WebAgents to store previous Web queries. In this paper we focus on the second aspect and show that the time cost can be decreased without significantly reducing the number of solutions found.

This paper is divided into 5 sections. Section 2 describes the *MAPWeb* architecture from a Multiagent perspective. Section 3 describes the caching skills of the WebAgents. Section 4 evaluates *MAPWeb* caching skills. And finally, section 5 summarizes the conclusions of the paper.

0-7803-7087-2/01/$10.00 © 2001 IEEE

2 MAPWeb Architecture

MAPWeb is structured into several layers whose purpose is to isolate the user from the details of problem solving and Web access. Each of these layers is implemented by a set of heterogeneous agents, which have different skills to solve the user problems. This multi-layer architecture can be seen in Fig. 1.

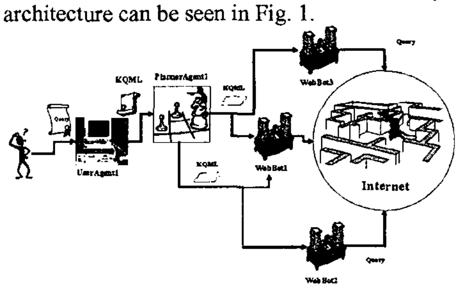

Fig. 1: MAPWeb system architecture.

MAPWeb deploys this architecture using a set of heterogeneous agents. Next, each of these types of agents will be described:

- **UserAgents**: They pay attention to user queries and display to the users the solution(s) found by the system. When an UserAgent receives problem queries from the users, it gives them to the PlannerAgent and when it answers back with the plans, the UserAgent provides them to the user.
- **PlannerAgents:** They receive a user query, build an abstract representation of it, and solve it by means of planning. Then, the PlannerAgent fills in the information details by querying the WebAgents. The planner that has been used by the PlannerAgent is Prodigy4.0 [10].
- **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.

The way these agents cooperate is as follows. First, the user interacts with the UserAgent to input his/her query. The query captures information like the departure and returns dates and cities, one way or return trip, maximum number of transfers, and some preference criteria. This information is sent to the PlannerAgent, which transforms it into a planning problem. This planning problem retains only those parts that are essential for the planning process, which is named the *abstract representation* of the user query. Prodigy4.0 provides several abstract solutions to the user query. The planning operators in the abstract solutions require to be completed with actual information that can be retrieved from the Web. To accomplish this, the PlannerAgent sends information queries to specialized WebAgents, which return several records for every information query. Then, the PlannerAgent integrates and validates the solutions and returns the data to the UserAgent, which in turn displays it to the user. MAPWeb agents use a subset of the KQML speech acts [7].

3 Caching techniques in the WebAgents

In this paper we have used caching techniques to reduce the number of actual Web queries. Caching has been implemented as a new skill for the WebAgents. MAPWeb caching follows the typical caching schema:

- Whenever an actual Web query is successful, a record is stored in the cache memory of the WebAgent. Cache memories have been implemented as local relational databases for each one of the WebAgents.
- Whenever a WebAgent is asked to carry out a query by the PlannerAgent, it firsts looks up its local database. If a record is found, then it is marked as if it were a new entry and it is returned to the PlannerAgent. If not successful, then it queries the Web directly.
- The cache memories have been limited in size. When the cache is full, old entries are removed by following a LRU policy (Least Recently Used: the oldest entries are deleted). As explained before, successful entries are automatically made young, This guarantees that useful entries will not be removed.

4 Experimental Evaluation

The purpose of this section is to compare the standard *MAPWeb* with a modified version where the WebAgents have extended caching skills.

Subsection 4.1 describes the domain that has been used to prove *MAPWeb*. Subsection 4.2 explains the topology that has been used in the experiments. Subsection 4.3 describes the experimental setup. And finally, Section 4.4 shows the actual results.

4.1 Problem Domain: e-Tourism

An e-tourism system must provide the user services such as:

- Inform how to go from the origin to the destination town using different means of transport.
- Lodging at destination.

- Informing about possibilities for visiting around town (renting a car, local transport, etc...).
- Returning to the initial (or other) town.

MAPWeb has the abilities enumerated above [4, 5]. However, in this paper, we will focus on the logistics problem of providing the user with plans to move from one place to another place.

Moving from place to place involves long range travels that can be achieved by means of airplanes, trains, or buses. It also involves taking local transport means (taxi, subway, bus, etc...) to move between airports, bus stations, or train stations. In order to represent and provide solutions to the user, we have defined an e-tourism domain that uses different planning operators like: [Travel-by-airplane, Travel-by-train, Travel-by-bus, Move-by-localbus, Book-hotel-room, etc...] .

4.2 *MAPWeb* Topology

Due to the heterogeneous nature of the agents that implement *MAPWeb*, it is possible to build different topologies. Those topologies could be used to study the performance system within a particular problem. In this paper a very simple topology was used (see Fig. 2).

. This topology includes a single PlannerAgent and four specialized WebAgents. Two of them offer information supplied by airplane companies (Iberia[1] and Avianca[2]) whereas the other two are *meta-searchers* (Amadeus[3] and 4Airlines[4]). A meta-searcher is a Web information source that is able to look in several information sources.

This topology employs only one reasoner agent that uses its planning skills to solve the problem given by the UserAgent and the WebAgents like softbots or searchers to validate and complete the solutions found. This topology should be analyzed like a *monolitic planning* application, which uses a set of distributed WebAgents in parallel.

[1] Iberia airlines: http://www.iberia.com
[2] Avianca airlines: http://www.avianca.com
[3] Amadeus: http://www.amadeus.net
[4] 4Airlines: http://www.4airlines.com

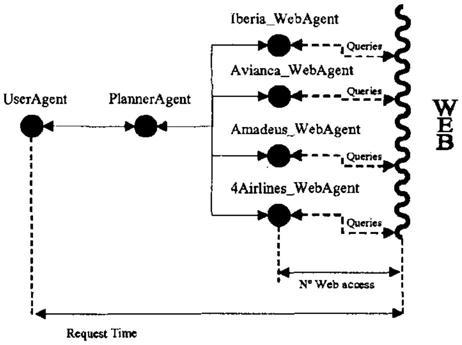

Fig. 2: Agents configuration and different characteristics measured in *MAPWeb*.

4.3 Experimental Setup

Two different configurations have been tested, with and without caching skills for the WebAgents. This caching skill allows us to evaluate how the modified system performs in three main aspects:

- *Number of Web queries*: this is the number of queries to the Web carried out by all the WebAgents to retrieve information for a user problem.
- *Time response*: this is the time between the user request for a problem and the system returning the solutions.
- *Number of solutions*: this is the total number of solutions found by the system. This is useful to show that the number of solutions with and without caching is about the same.

The later characteristics are displayed in Fig. 2. In order to evaluate the WebAgents performance (with and without caching), a set of 35 user queries was tested. Each one of them was tried with 0, 1, and 2 transfers.

4.4 Experimental Evaluation

Figs 3 and 4 display the average number of solutions (plans) per user query found for the non-caching and caching configurations, respectively. The number of abstract solutions is the number of abstract solutions found by the planner and validated by the WebAgents, whereas the number of specific solutions represent the number of actual solutions that could be validated and completed by the WebAgents. As it could be expected, when the number of transfers increases, the number of solutions increases as well.[5] Also, we can see

[5] A transfer is a point in the trip where the traveler can change plane, the means of transport, etc.

that the number of solutions for the non-caching and caching configurations is about the same (not many solutions are lost by using caching).

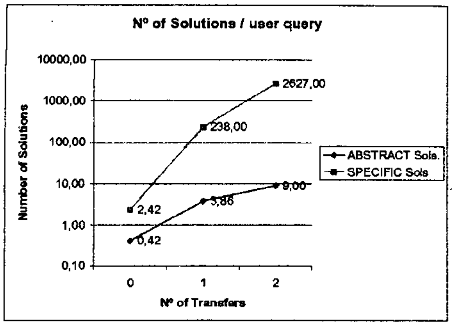

Fig. 3: Number of Abstract and Specific solutions found by *MAPWeb* without caching.

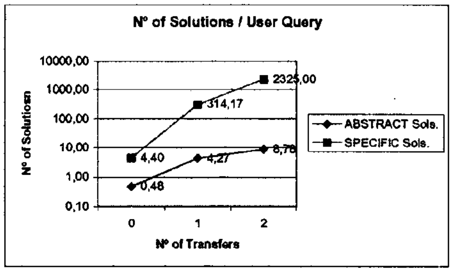

Fig. 4: Number of Abstract and Specific solutions found by *MAPWeb* with caching.

Fig. 5 shows the number of queries with and without caching. The number of queries is the same for the two experiments; the only difference between the two configurations is the number of Web accesses that the WebAgents (with caching skills) will finally perform.

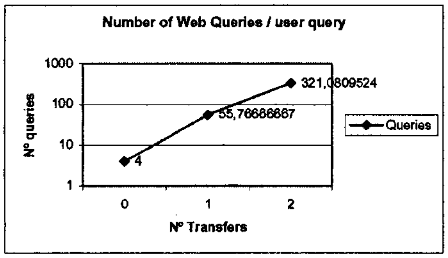

Fig. 5: Number of queries between the PlannerAgent and the WebAgents.

Figs 6 and 7 display the response time with and without caching, respectively. The reduction of the request time in *MAPWeb* is related with the successful queries that the WebAgents have stored in their local databases. By comparing Fig. 7 with Fig. 6, we can see that request time grows more slowly with caching. This is because it is possible to find more stored information in the agent.

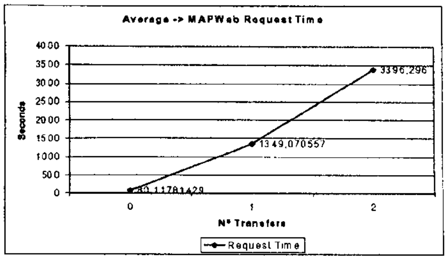

Fig. 6: Request time by *MAPWeb* without caching.

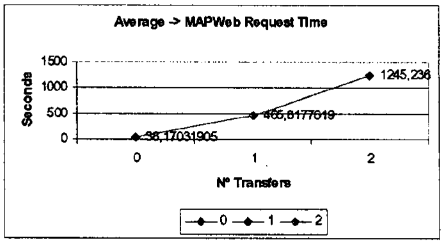

Fig. 7: Request time by *MAPWeb* with caching.

Request times are related to the hit ratio obtained by the WebAgents when they find the requested information in their own local databases. Table 1 shows the total number of actual Web queries with and without caching, and the hit ratio when using caching. Only 1 transfer problems are considered. On average, the number of Web queries was reduced by 25%.

N° Web Access	Iberia WebAgent	Avianca WebAgent	4Airlines WebAgent	Amadeus WebAgent
No Caching	60	60	60	60
With Caching	39	53	47	40
Hit ratio	35%	12%	22%	33%

Table 1: Average of Web queries cached by WebAgents.

Table 2 shows the different answer time by the WebAgents when the caching techniques are used. This table shows how the performance of the slower agents (Amadeus-WebAgent) could be improved through the caching technique, and how the agents that have a lower hit ratio of success (Avianca-WebAgent) do not improve very much their request time.

Request Time	Iberia WebAgent	Avianca WebAgent	4Airlines WebAgent	Amadeus WebAgent
No Caching	88,4 sec	96,4 sec	156,4 sec	1440,5 sec
With Caching	39 sec	84,6sec	50,1sec	56,3 sec

Table 2: Average of request time for different WebAgents in the *MAPWeb* Topology.

Figs. 8, 9 10, and 11 show the number of solutions found for each of the 35 user queries (on the x-axis). There are no important differences between the two configurations of *MAPWeb*. These figures show that the caching techniques allow the system to gain efficiency without loosing possible solutions.

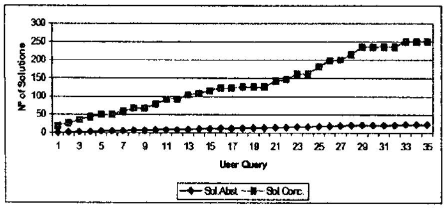

Fig. 8: Number of solutions found by *MAPWeb* for 0 transfers flights without caching.

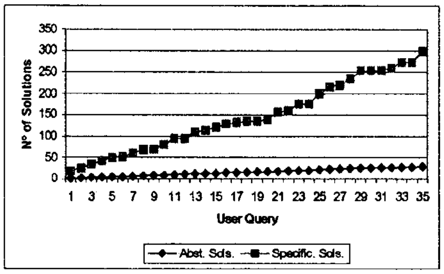

Fig. 9: Number of solutions found by *MAPWeb* for 0 transfers flights with caching.

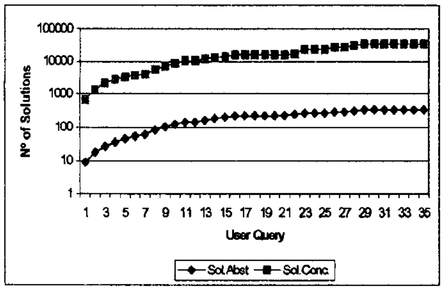

Fig. 10: Number of solutions found by *MAPWeb* for 1 transfer flights without caching.

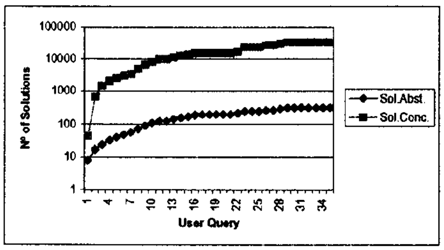

Fig. 11: Number of solutions found by *MAPWeb* for 1 transfer flights with caching.

5 Conclusions

Planning in real domains is a complex task. We present here a solution that separates abstract planning from the actual data, that is obtained from the Web. In order to give detailed solutions, for a particular travel domain, *MAPWeb* implements two different steps. First, it uses abstract planning to build a skeletal solution for the problem. And second, it uses information gathering techniques to retrieve travel information from Web sources. However, Web access to the information sources is quite time and computational expensive. In this paper, we show how it is possible to minimize the number of Web queries by using caching techniques based on relational databases, and therefore, to find solutions in a more efficient way. Experimental results show that the reduction in Web access time is quite important, while maintaining the number of solutions found.

Acknowledgments

The research reported here was carried out as part of the research project funded by CICYT TAP-99-0535-C02. (http://decsai.ugr.es/$\sim$lcv/SEPIA/tap99-0535-c02-01.html)

References

1. Ambite J.L., Knoblock C.A. Planning by rewriting: Efficiently generating high-quality plans. In proceedings of the Fourteenth National Conference on Artificial Intelligence. 1997.
2. Ambite J.L., Knoblock C.A. Agents for information gathering. IEEE Expert: Intelligent Systems and their Applications. September/October (1997).
3. Brenner W., Zarnekow R., Wittig H.. Intelligent Software Agents. Foundations and Applications. Springer-Verlag, 1998. ISBN: 3-540-63411-8.
4. Camacho D., Borrajo D., Molina J.M.. TravelPlan: A Multiagent System to Solve Web Electronic Travel Problems. Fourth International Conference on Autonomous Agents (Agents 2000). Workshop Agent-based Recommender Systems (WARS 2000). Barcelona,, Spain. June, 2000.
5. Camacho D., Molina J.M., Borrajo D.. A Multiagent Approach for Electronic Travel Planning. Second International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS-2000), Austin, Texas, USA. August 2000. AAAI Press.

6. Camacho D., Molina J.M., Borrajo D., Aler R.. *MAPWEB*: Cooperation between Planning Agents and Web Agents. Information & Security: An International Journal. Volume 7. 2001

7. Finin T., Fritzson R., McKay D., McEntire R. KQML as an agent communication language. In Proceedings of the International Conference on Information and Knowledge Management. ACM Press, New York. 1994.

8. Hullen J., Bergmann R., Weberskirch F. WebPlan: Dynamic Planning for Domain-Specific Search in the Internet. Workshop Planen und Konfigurieren (PuK-99). 1999.

9. Hunhs M.N., Singh M.P. Readings in Agents. San Francisco California, Morgan Kaufmann. 1997.

10. 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. Vol. 7, pages 81-120, 1995.

11. Wooldridge M. , Jennings N.R. Intelligence Agents: Theory and Practice. Knowledge Engineering Review. October, 1994.