Multi-agent plan based information gathering

David Camacho · Ricardo Aler · Daniel Borrajo · José M. Molina

Abstract The evolution of the Web has encouraged the development of new Information Gathering techniques. Artificial Intelligence techniques, such as Planning, have also been used for Information Gathering in order to go beyond merely retrieving Web data. Planning has been used traditionally to generate a sequence of actions that specify how information sources should be accessed. In this paper, planning is used mainly for integrating information found in heterogeneous sources. For instance, two different Web sources about flight and train travels, can be represented by two different planning operators, which will be subsequently combined and integrated by a single plan. We have found that a Multi-Agent framework is very appropriate to implement our technique. In order to evaluate our approach empirically, it has been applied to a tourism domain (MAPWEB-ETOURISM), whose purpose is to help a customer to plan his/her trips. In this domain, several specialized Web agents have been used to query travel Web sources, whose results are subsequently integrated by a planning agent to build complete travel solutions. Experimental results show that, by means of integration, more solutions can be found than by using single information sources or even travel meta-searchers. Also, MAPWEB-ETOURISM can find new types of solutions by integrating information gathered from heterogeneous Web sources (i.e. flights and trains).

D. Camacho (✉)
Universidad Autónoma de Madrid, Computer Science Department, C/Francisco Tomás y Valiente, n° 11, CP 28049, Madrid, Spain
e-mail: David.camacho@uam.es

R. Aler · D. Borrajo · J. M. Molina
Universidad Carlos III de Madrid, Computer Science Department, Avenida de la Universidad n° 30, CP 28911, Leganés, Madrid, Spain

Keywords Information gathering · Multi-agent systems · Planning · Web agents

1. Introduction

The evolution of the Web has originated new possibilities that go beyond what traditional Information Retrieval (IR) [26] and searching techniques provide. These possibilities arise because many complex problems can be solved using the information available in many electronic sources. Information Gathering [8, 14, 20, 21] (IG) intends to integrate a set of different information sources with the aim of querying them as if they were a single information source.

Many different kinds of systems, named *mediators*, have been developed. They try to integrate information from multiple distributed and heterogeneous information sources, like database systems, knowledge bases, web servers, electronic repositories . . . (an example is the *SIMS* [4] architecture). In order that these systems are practical, they must be able to optimize the query process by selecting the most appropriate WEB sources and ordering the queries. For this purpose, different algorithms and paradigms have been developed. For instance, *Planning by Rewriting* (*PbR*) [1] builds queries by using planning techniques. These approaches use planning techniques to select the appropriate WEB sources and order the queries to answer generic user queries. That is, they use planning as a tool for selecting and sequencing the queries.

In this paper we describe MAPWEB, a multi-agent information gathering system that also uses planning, but with a different purpose. MAPWEB uses planning for both determining the appropriate generic sources to query and solving actual planning problems. For instance, in this paper, the MAPWEB framework is applied to a travel planning assistant domain, where the user needs to find a plan to travel between several places. Each plan not only determines what steps the user must perform, but which information sources should be accessed. For instance, if a step is to go from A to B by plane, the system provides the user the information of what airplane companies should be consulted for further information. Using planning has the advantage that, if it is desired to add a new information source to the system, it is only necessary to change the planning domain. For instance, if taxi fares were made suddenly available in the WEB, it would only be necessary to add a move-by-taxi operator along with the associated WebAgent.

RETSINA [23, 29] is a multi-agent architecture with 3-layers (interface, task, and information layers) where agents have planning capabilities. Our multi-agent system follows a similar architecture. However, in our case planning is not just a skill for agents to achieve their goals. Planning is mainly used to determine which information is to be queried and to integrate heterogeneous gathered data into a detailed solution.

The main contribution of this paper is to show empirically how our planning-based approach can be used to coordinate different Web (information) agents by using a plan as a template to decide which agent and which information source will be queried. And then, this plan will be used to integrate the data gathered from the Web. In particular, our results show that by integrating data from heterogeneous web sources, more travel problems can be solved that cannot be solved without information integration.

This paper is structured as follows. In Section 2, the Multi-Agent architecture (MAPWEB) used to integrate our Plan-based IG technique is briefly described. Section 3 describes the Information Gathering process performed by MAPWEB, three main points will be addressed; how planning is used by MAPWEB to gather information and to solve problems; the role of the specialized WebAgents; and finally, how all the retrieved information are integrated to build new solutions. Section 4 provides the experimental results about the behaviour of the Plan-based IG technique implemented. Section 5 describes the related work. Finally, Section 6 summarizes the conclusions of the paper.

2. MAPWEB system architecture

In this Section, only a summary of the architecture is presented. A detailed description can be found in [5]. Our architecture, called MAPWEB (**M**ulti-**A**gent **P**lanning in the **Web**), is a generic MAS information gathering architecture that integrates planning and Web information gathering agents. This architecture provides a reusable code to help with the development of new Web gathering systems. The main goal of this framework is to easily allow the integration of AI solving problem techniques (like planning or machine learning) in Web domains. There are three main roles in the system: users, solvers, and information agents. Therefore, our system follows a 3-layer architecture, as other approaches like RETSINA [23]. As we also want to implement teams of agents, a new kind of agents has been included (control agents). Figure 1 shows one possible MAPWEB topology, or configuration. This configuration is built by two operative *teams* managed by a *Manager* agent, and every team is locally managed by a *Coach* agent. *Team*$_1$ has the minimun set of agents to be operative, whereas *Team*$_2$ is built by several UserAgents, PlannerAgents and WebAgents. In addition, it is possible to use the following agents:

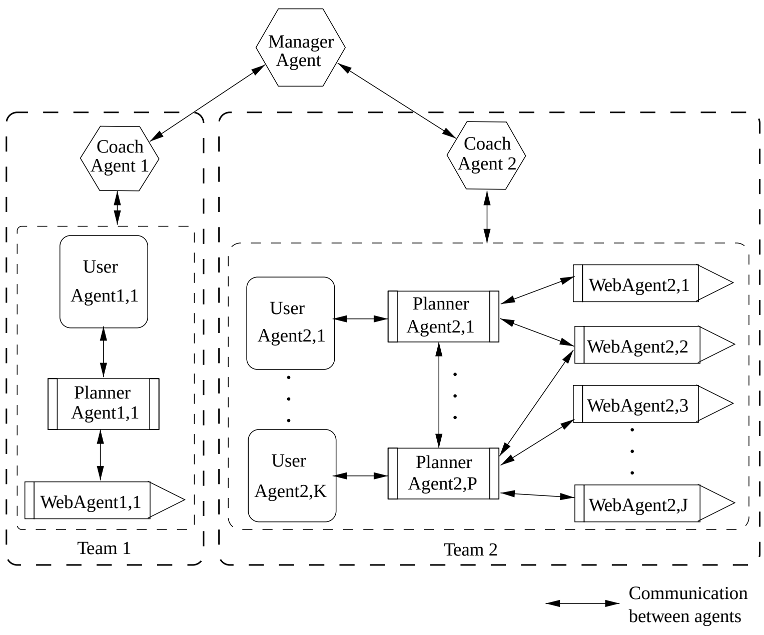

Fig. 1 MAPWEB IG architecture

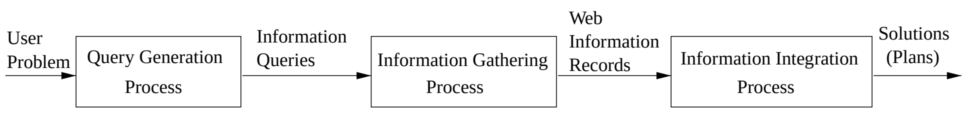

Fig. 2 Plan-based information gathering processes in MAPWEB-ETOURISM

- UserAgents are the bridge between the users and the system. They only implement basic input/output skills to acquire problem descriptions from users and to show the solutions found to them, and to acquire the problem information.
- PlannerAgents are able to solve planning problems using the information gathered from the Web.
- WebAgents are able to provide the requested Web information like a set of relational records to the PlannerAgents using wrapping techniques. These agents have implemented learning skills (caching) that are used to stored useful information in their own local data base to reduce the number of access to the WEB.
- ControlAgents (Manager and Coach Agents) that are responsible to manage and coordinate the previous of agents. These agents implement several control tasks like register or unregister agents. These agents have similar skills to the Agent Manager System (AMS) or the Facilitator agents in the FIPA architecture (http://www.fipa.org).

Agents in MAPWEB use a common representation (ontology) for their knowledge. This characteristic allows to simplify the processes of sharing and reasoning with the knowledge. The coordination among the agents is carried out using a standard communication language (KQML [9, 10]) to perform actions over their environment. A message in KQML is called *performative* (this term is taken from the speech act theory [6]) and can be understood like a request for an specific action to be carried out.

3. MAPWEB information gathering process

The MAPWEB framework is a general agent-based approach that could be applied in different WEB domains. To implement a specfic version of this architecture (that we named MAPWEB-ETOURISM) a travel planning assistant domain has been selected. This domain is a modified version of the Logistics domain [31], where the user needs to find a plan to travel between 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-ETOURISM agents solve planning problems by means of cooperation and knowledge integration between PlannerAgents and WebAgents. Interactions between MAPWEB-ETOURISM agents allow to find solutions from the retrieved information. This whole process can be seen like an Information Gathering process that uses planning techniques to decide both, which information sources will be access and how the specific information found will be integrated to build new solutions. This plan-based information gathering technique achieved by MAPWEB-ETOURISM can be summarized in three main subtasks:

1. *Query generation process*. The specific information provided by the user is properly translated into an abstract representation of the problem.
2. *Information gathering process*. Once the PlannerAgent has found a set of possible abstract solutions for the given problem, a set of information queries (partially instantiated) are built and sent to the appropriate WebAgents that will retrieve the specific information to complete and validate those abstract solutions.
3. *Integration process*. With the specific information, each PlannerAgent builds new solutions sharing and combining the specific records retrieved from the WEB.

Figure 2 shows the previous processes; the first process inputs the definition of the problem given by the user, and outputs a set of *information queries* that will be used by second process to retrieve specific information from the WEB. Finally the last process will use the WEB information retrieved to build a set of solutions that will be given to the user.

3.1. Query generation process

This process is achieved by the cooperation between two kind of agents, the UserAgents and the PlannerAgents. The user can fill in the details for every step in the travel problem through a set of Graphical User Interfaces (GUI). To illustrate this process, let us suppose that a user wants to travel from Madrid (MAD) to Barcelona (BCN) (using 0 or 1 transfers), staying three nights in Barcelona and finally returning to Madrid. Besides the information shown in Table 1, the user can also specify the locations inside the city where s/he wants to start or end the trip (like an airport, a train station, or a

Table 1 Travel example from Madrid to Barcelona by airplane or train

Leg	Stage	Date	Restrictions	Transfers
1	MAD → BCN	June 11th 2004	plane/train	0 or 1
2	3 nights stay	June 11th 2004	<90 €	–
3	BCN → MAD	June 14th 2004	plan/train	0 or 1

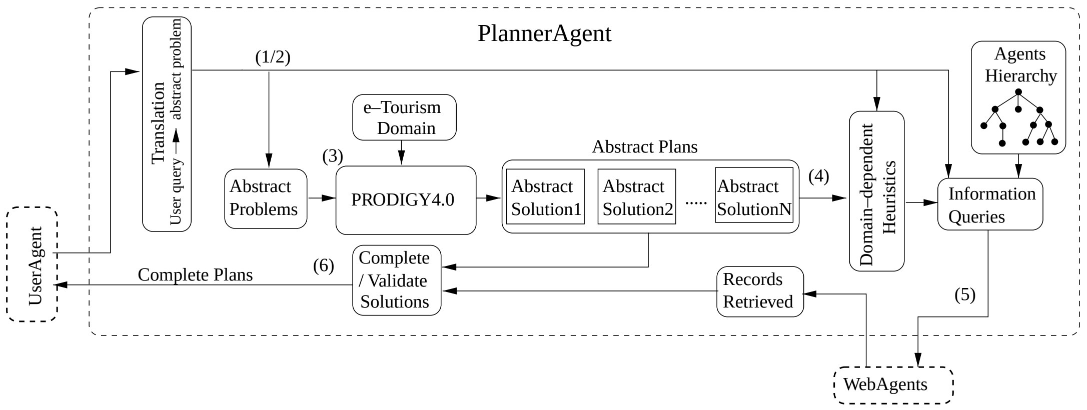

Fig. 3 High level description of the planning process carried out by PlannerAgents

bus station). Using the previous information a *user problem* is generated by the UserAgent and sent to a PlannerAgent. Using this information a *user problem* is generated by the UserAgent and sent to a PlannerAgent. Table 1, shows the user problem built from previous example. This problem will be received and analyzed by a PlannerAgent.

When a planner agent receives the user problem, it performs the following steps (see Fig. 3):

1. The PlannerAgent receives a query from UserAgent. This query is analyzed and translated into an abstract planning problem.
2. The planning problem is divided into a set of subproblems. Any planning problem can be divided if it has more than one goal. Therefore, the PlannerAgent splits it into one-goal subproblems.
3. The PlannerAgent uses its own skills and knowledge about the problem and tries to solve it. The abstract representation of the problem, and the description of the problem-domain (*e-tourism*) are given to a planner (*Prodigy4.0* [30]) that tries to obtain a set of *abstract* solutions for the subproblem.
4. These solutions are too general and only have the essential information for the planning process, so they need specific information to be completed and validated. The PlannerAgent builds a set of *information queries* for the WebAgents.
5. It is important to try to optimize the number of queries due to the large number of possible instantiations. So several domain-dependent heuristics are used by the PlannerAgents. When the queries have been built using these heuristics, the PlannerAgent selects from its yellow pages the set of WebAgents that will be queried.
6. Finally, when the WebAgents answer with the information found in the Web (if the WebAgents are successful) the PlannerAgent integrates all the specific information with the abstract solutions to generate the final solutions that will be sent to the UserAgent (see Section 3.3).

PRODIGY4.0 is a nonlinear problem solver derived from the PRODIGY architecture. PRODIGY4.0, follows a means-ends analysis backward chaining search procedure reasoning about multiple goals and multiple alternative operators relevant to the goals [30]. The inputs to the problem solver algorithm are: a domain theory, $\mathcal{D}$, that includes a set of generalized operators (similar concept to rules in KBS) and an object hierarchy; a problem to be solved specified in terms of an initial state (starting knowledge) and a set of goals; and control knowledge (heuristics), described as a set of control rules, that guide the search process. PRODIGY4.0 follows a cycle where first a goal is selected from the set of open goals at a given moment), then an operator is chosen, and finally the bindings (values to be assigned to variables of the operator) are determined. PRODIGY4.0 is used as a skill of the PlannerAgents as Fig. 3 shows.

After removing unnecessary details, the PlannerAgent tranforms the user query into an abstract problem. First, it defines an abstract city (`city0`). That includes all possible local transports, but only the long range transport terminals that the user wishes to use are included, like airports or train stations. Then, this abstract city is copied as many times as the maximum number of transfers supplied by the user. It is important to remark that the cities are abstract cities (i.e. they have no attached names, so they are present in the abstract plan to represent the initial, intermediate, and final travel points). The rest of details provided by the user are ignored at this stage. The abstract problem represents the initial state and the goals of the problem that are the inputs to PRODIGY4.0.

In this case, from the first leg of the trip (travel from Madrid to Barcelona), with only one transfer, the PlannerAgent would generate an abstract problem, where The user wants

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

```
Problem: 0-Transfers
Solution 1:
 <travel-by-airplane user1 plane0 airport0 airport2>
 <move-by-local-transport user1 lbus2 bustop20 train-station21 city2>

Solution 2:
 <move-by-local-transport user1 lbus0 bustop00 train-station01 city0>
 <travel-by-train user1 train0 train-station0 train-station2>

Problem: 1-Transfer
Solution 3:
 <travel-by-airplane user1 plane0 airport0 airport1>
 <move-by-local-transport lbus1 bustop10 train-station11 city1>
 <travel-by-train user1 train1 train-station1 train-station2>

Solution 4:
 <travel-by-airplane user1 plane0 airport0 airport1>
 <travel-by-airplane user1 plane1 airport1 airport2>
 <move-by-local-transport lbus2 bustop20 train-station20 city2>
...
```

to begin her/his travel from an airport and wants to arrive to a train station inside the arrival city. Both characteristics are included in the abstract problem (initial state/goal) and are used in the planning process.

The abstract problem would be given to the PlannerAgent planner which would obtain several possible abstract solutions. In this case, the planner would generate the abstract plans. Some of them appear on Fig. 4. They represent generic solutions for the given problem (solutions with 0 and 1 transfers).

This is a set of abstract plans that contain no details. Some of the plan steps might not even be possible because, for instance, there are no train-companies linking two specific cities. For instance, Solution 1, describes that it is necessary only to take a plane to the destination city and then the user needs to take a local transport (bus) to arrive to the desired train station. Solution 2 provides another possible solution, where the user goes first to a train station inside the departure city, and then takes a train to the destination city. Therefore, those plans need to be *completed* and *validated*. The abstract steps in the solution contain unbound variables that relate to transfer cities. They need to be bound before the WebAgents are queried. The PlannerAgent restricts the number of bindings by applying two simple heuristics; a *geographic heuristic* and a *population/distance heuristic*. The first heuristic is used to select a set of possible cities that could be used for a transfer. The second one is used to order (by means of a relative importance value) the initial list of cities. Finally a threshold is used so that only some of the cities are selected. The Geographic Heuristic performs the following three steps:

- If the origin and arrival cities belong to the same country, only the cities in that country are considered as possible transfer cities.
- Else, if the origin and arrival cities belong to the same continent, only the cities of that continent are considered.
- Otherwise, all cities are considered.

The Population/distance Heuristic is a combination of two values, the first one related to the population of the city and the second one related to its distance to the origin and destination cities. This heuristic is used to order the list of cities returned by the Geographic Heuristic. The population heuristic supposes that those cities with larger population (relative to the largest city in the country) are usually better connected by some transports. It uses Eq. (1) to obtain the relative importance of a city in its country. f_p is a value between 0 and 1 (values near to 0 represent smaller populations), P_X represents the population of the considered city and finally $P_{\max}$ represents the population of the biggest city in the country.

$$f_p = P_X / P_{\max} \tag{1}$$

The Distance heuristic uses Eqs. (2) to (5) to obtain the distance between two cities (a and b), where $(lat1, long1)$, $(lat2, long2)$ represents the longitude and latitude of two

Fig. 5 Agents hierarchy. It describes all the available agents in MAPWEB-ETOURISM and their information gathering skills

```
Agent
    |__UserAgent0
    |--ReasonerAgent
                    |__PlannerAgent0
    |__WebAgents
                |__Travel
                         |__Fly: WebAgent-Iberia,
                                 WebAgent-Amadeus-Flight
                         |__Train: WebAgent-Renfe,
                                   WebAgent-RailEurope
                         |__Bus
                |__Hotel: WebAgent-Amadeus-Hotel
                |__Car
    |__ControlAgents
                    |__ManagerAgent0
                    |__CoachAgent0
```

cities, and R is the radius of earth.

$$a = \sin(lat1) * \sin(lat2) \quad (2)$$

$$b = \cos(lat1) * \cos(lat2) * \cos(long2 - long1) \quad (3)$$

$$c = arc\cos(a + b) \quad (4)$$

$$d(a, b) = R * c \quad (5)$$

The Eq. (6) is used by the PlannerAgents to calculate how close a possible transfer city is from the straight way between the departure and arrival cities. $d_{a\to b}$ represents the distance between departure and arrival cities, $d_{a\to X}$ and $d_{X\to b}$ represent the distance between departure, arrival and the selected transfer city, respectively. The result is also a value between 0 and 1.

$$f_d = d_{a\to b}/(d_{a\to X} + d_{X\to b}) \quad (6)$$

Both parameters, f_d and f_p, are combined by Eq. (7), to obtain the goodness of the considered city to be used as a transfer. The δ and ρ parameters have values between 0 and 1 and satisfy: $[\delta + \rho = 1]$. Several empirical tests were made and finally the values $\delta = 0.75$, $\rho = 0.25$ were selected for the heuristic.

$$F = \delta f_d + \rho f_p \quad (7)$$

Finally, the value F obtained for each city is used to order the list of cities given by the Geographic Heuristic. From the ordered list only a subset of the cities will be used (to minimize the number of information queries that will be requested to the WebAgents). Only the top 10% of the list will be used. Several empirical test were made to estimate this threshold. Finally, 10% was selected because the experimental test showed that this was the minimum number of cities necessary to find information for most of the requested problems.

For instance, in the previous example the first leg of the trip, as Madrid and Barcelona belong to the same country, the Geographic Heuristic provides an initial list of possible transfer cities that belong to Spain (currently, about thirty). Then these cities are ordered using the Population/Distance Heuristic to finally (using the threshold) restrict this number of cities to the most promising candidates (three cities) to bind the unbound variables in the abstract plans. For instance, in the previous example the selected cities were: Valencia, Zaragoza, and Alicante. These heuristics are used only to minimize the number of Web accesses and to allow a better performance (in time response) of the system.

Once the unbound variables have been instantiated, the PlannerAgents need to select the appropiate WebAgents to ask for the information. Planning operators of the abstract solutions and Web sources are related by means of a WebAgent *hierarchy*. This hierarchy allows each PlannerAgent to know which WebAgents know how to retrieve the required information. The specific hierarchy that we built for this domain is represented in Fig. 5.

If a planning operator is repeated in different abstract solutions, it is only considered once, to avoid repeating queries. For instance, in the solutions for 1-transfer problems, the operators `<travel-by-airplane ...>` and `<travel-by-train ...>` would be finally translated as shown in Table 2.

Table 2 Queries partially instantiated to the appropriate WebAgents

Query sent to the WebAgents	WebAgent
(travel-by-plane user1 plane0? Mad Barcelona)	Iberia, Amadeus-Flight
(travel-by-train user1 train0? Mad Barcelona)	Renfe, RailEurope
(travel-by-plane user1 plane0? Mad Alicante)	Iberia, Amadeus-Flight
(travel-by-plane user1 plane0? Mad Valencia)	Iberia, Amadeus-Flight
(travel-by-train user1 train0? Mad Zaragoza)	Renfe, RailEurope
...	...

Those queries (and all the additional information given by the UserAgent) are sent to WebAgents that know about airplane and train travel information, respectively. Thus, the variables `plane0?` and `train0?` will be instantiated as well (if the gathering process in the WebAgents is successful).

3.2. The information gathering process

This process is achieved by WebAgents in MAPWEBETOURISM. These agents receive the information queries from the PlannerAgents, transform them into actual Web queries, access to its known Web source, and return the gathered information to the PlannerAgent in a standard format. The translation of the information is performed by Wrappers [25, 27]. Figure 6 displays the WebAgent architecture. A WebAgent is made of three main components: a control module, a record database, and one Wrapper.

Any WebAgent implements several processes that can be summarized as:

1. *Retrieve stored information*. When a WebAgent receives a query from a PlannerAgent, its control module tries to fulfill the query by retrieving the appropriate stored records from its local database.
2. *Build a Web query*. If the the local database fails, the agent builds a query to search for the requested information in the Web source. If the user does not provide all the necessary information to access the source, the WebAgent will fill in the necessary fields with predefined values.
3. *Wrapping process*. Once the query is built, the agent uses its automatic Web access skill to gather the information. The following tasks are performed in this process:

   (a) *Retrieving a Web document*. This task simply emulates the action of a human fetching the page from his/her Web browser.
   (b) *Extracting information*. Given that the electronic source returns is in HTML format, it is necessary to filter the page to extract the specific information. To simplify the gathering process, we are using semi-structured Web sources (these sources can be characterized because the relevant information is stored in tables or lists in the HTML page).
   (c) *Mapping information*. The information extracted is translated into a common internal structure (or *record*) for all agents in the system.
   (d) *Storing information*. The gathered records are stored into a relational database. This will avoid repeated accesses to the Web for the same information.

4. *Answer to the PlannerAgent*. Finally, from the database or the Web, the gathered records are sent to the PlannerAgent.

From the previous example, the following queries would be answered by several WebAgents (*WebAgent-Iberia*, *WebAgent-Amadeus-Flight*, and *WebAgent-Renfe*) with the records shown in Table 3:

- (*travel-by-plane user* 1 *plane* 0*? Madrid Barcelona*)
- (*travel-by-train user* 1 *train* 0*? Madrid Barcelona*)

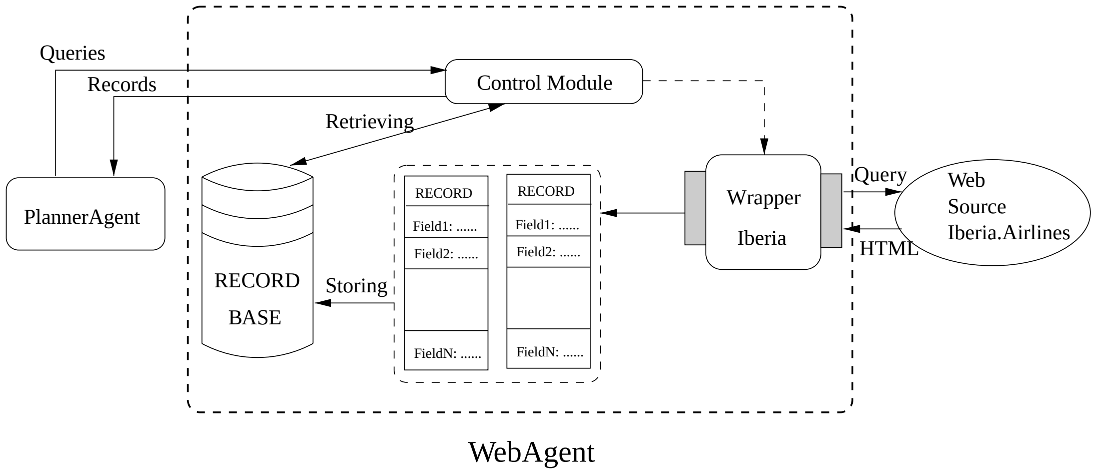

Fig. 6 WebAgent architecture

Table 3 Information retrieved by WebAgents from airfligths and train companies

Inf-FLIGHTS	record1	record2	record3	Inf-TRAINS	record1	record2
WebAgent	Iberia	Amadeus	Amadeus	WebAgent	Renfe	Renfe
air-company	Iberia	Iberia	Portugalia	train-company	RENFE	RENFE
http-address	w3.iberia.es			http-address	w3.renfe.es	w3.renfe.es
flight-id	IB8797	IB8819	NI711	train-id	07054	07056
ticket-fare	424.5			ticket-fare	4.7	4.7
currency	EUR	EUR	EUR	currency	EUR	EUR
flight-duration	3 h 45 min	2 h 00 min	2 h 10 min	departure-city	MAD	MAD
airp-depart-city	MAD	MAD	MAD	departure-date	11-09-01	11-09-01
departure-date	11-09-01	11-09-01	11-09-01	departure-time	6:30	8:30
airp-arrival-city	BCN	BCN	BCN	arrival-city	BCN	BCN
return-date	null	null	null	arrival-date	11-09-01	11-09-01
class	Tourist	null	null	arrival-time	7:53	9:47
num. passengers	1	1	1	class	Tourist	Tourist
round-trip	one-way	one-way	one-way			

Here, we use a simple extraction process from the html page. It is known that the information is stored in the html page in a table. Therefore, the html code before and after the table is removed. Then, the table is parsed to extract the desired raws and columns. Finally, this information is translated into a relational format that can be used by other agents in the system. In the future, we would like to automatize this process by using automatic Wrapper generation techniques [18].

3.3. The integration process

This last process is performed by a PlannerAgent that integrates the retrieved information (records) into a set of instantiated solutions, or plans, that finally are sent to the appropriate UserAgent. The abstract plans produced in the first phase are used by the PlannerAgent as a template to integrate heterogeneous data. Figure 7 shows two successful plans that are completed by the PlannerAgent. The operators are instantiated with specific information (records) that could be shared between different plans. Successful plans are stored in a relational *Plan Base*. It is possible to retrieve plans at two different levels of generality; either abstract plans or specific plans can be retrieved through indexes. The former are located by using the `Abstract-Goal-index`, and the latter by means of the `Goal-index`. Those plans in which one or several steps failed are rejected.

All the PlannerAgents in MAPWEB-ETOURISM use a *scheduling module* to avoid sequencing two records that can not be scheduled. Therefore, for any record gathered by the WebAgents, if this information relates to a travel step and the time is known (for instance, a flight from Madrid to Barcelona arriving at 10:00 am), it will only be possible to use this record with other travel records in the same plan if the new transport departures after an elapsed time. For instance, it will not be possible to take a train which leaves from Barcelona to Valencia before 11:00 am because the user needs to go from the airport to the train station. Any PlannerAgent uses a parameter named *elapsed-time* to schedule two (travel) records. Therefore, only fulfillable plans are sent back to

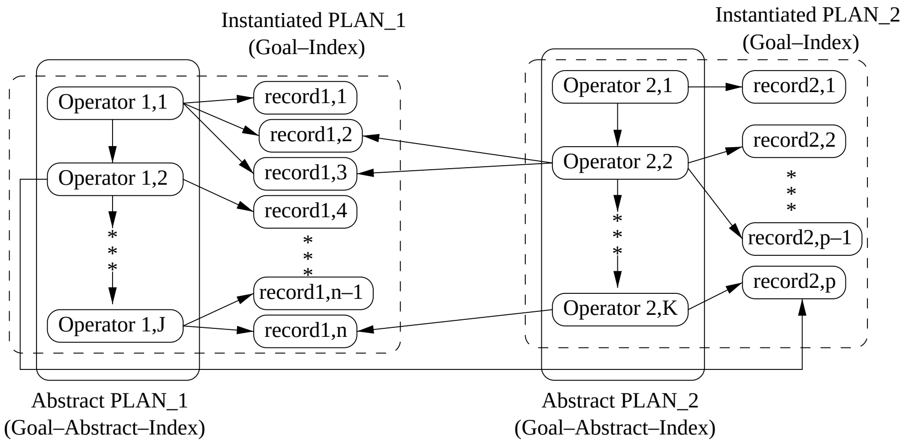

Fig. 7 Relationship between abstract and specific information instantiated in the PlannerAgents

Fig. 8 New agents hierarchy with a new specialized Web agent

```
Agent
    |__UserAgent0
    |--ReasonerAgent
                |__PlannerAgent0
    |__WebAgents
            |__Travel
                    |__Fly: WebAgent-Iberia,
                            WebAgent-Amadeus-Flight
                    |__Train: WebAgent-Renfe,
                              WebAgent-RailEurope
                    |__Bus
                    |--Subway: WebAgent-subway (new WebAgent)
            |__Hotel: WebAgent-Amadeus-Hotel
            |__Car
    |__ControlAgents
                |__ManagerAgent0
                |__CoachAgent0
```

the UserAgent. Every abstract plan will be instantiated into many different actual plans.

3.4. Adding new information sources

To show the flexibility of our Plan-based IG approach, let us suppose that we wish to add a new information source in MAPWEB-ETOURISM. This new source allows to consult the city subway system to travel between two places inside the city. How will this new type information affect the whole system? On the one hand, it is necessary to build a specialized WebAgent capable to extract information from the corresponding Web source. This new agent will be inserted in the hierarchy of the PlannerAgents to allow both, the communication between planning and Web agents and to integrate the gathered information into the new solutions. Figure 8 displays the new hierarchy that will be used by MAPWEB-ETOURISM agents.

On the other hand, the planning domain needs to include a new operator that is able to look for solutions taking into account that it is possible to take the subway inside the city. The new planning operator is shown in Fig. 9.

The description of the problem can now use two types of local transports (bus and subway) because there are two operators that are able to reason about this information. These new solutions integrate within the plan the information provided by the new WebAgent-subway. The new (abstract) solutions found by the PlannerAgent will be able to use the subway to achieve the user goals. Some of them are shown in Fig. 10.

Finally, the previous skeletal plans and the specific information provided by the Web specialized agent will be used to build the complete solutions.

4. Experimental evaluation

The aim of the experiment carried out in this section is to evaluate how the IG technique is able to find new solutions that cannot be found without information integration. The experimental setup is as follows:

- Three categories of problems have been considered: National trips (within Spain), European trips, and International trips. A set of 30 test problems was used. Ten planning problems were randomly generated for each category. To generate those problems, we selected randomly pairs of cities from a set containing 100 cities. 40 of them were national (Spanish) cities, 30 additional European

```
(operator TRAVEL-BY-SUBWAY
 (params <sb0> <start> <end> <city>)
  (preconds
   ((<sb0> SUBWAY)
    (<start> LOCATION)
    (<end>  LOCATION)
    (<city>  CITY))
   (and
    (loc-at <start> <city>)
    (loc-at <end> <city>)
    (loc-at <sb0> <start>)))
   (effects()
    ((del (loc-at <sb0> <start>))
     (add (loc-at <sbo> <end>)))))
```

Fig. 9 Travel-by-subway operator

Fig. 10 Some simple solutions when a new operator is added to the planning domain

```
Problem: 0-Transfers
Solution 1:
 <travel-by-airplane user1 plane0 airport0 airport2>
 <move-by-bus user1 lbus2 bustop20 train-station21 city2>
Solution 2:
 <travel-by-airplane user1 plane0 airport0 airport2>
 <travel-by-subway user1 sb2 airport2 train-station2 city2>
Solution 3:
 <travel-by-subway user1 sb0 airport0 train-station0 city0>
 <travel-by-train user1 train0 train-station0 train-station2>
...
```

cities, and 30 additional extra European cities. Finally, several travel characteristics, like departure date, number of passengers, type of transport to be used, etc. . . were also generated randomly.

- Then, several configurations (or *Topologies*) were implemented in MAPWEB-ETOURISM. Only one UserAgent and one PlannerAgent were used in the configurations. Only the number and type of WebAgents is variable. Three different WebAgents were used to build the topologies: Amadeus-Flights (AMF, http://www.amadeus.net), Iberia (IBE, http://www.iberia.com), and Renfe (RNF, http://www.renfe.es). Amadeus-Flights is a metasearcher; it can search information from many airplane companies. Iberia and Renfe can only search information about their own knowledge sources (flights and trains respectively).
- MAPWEB-ETOURISM was then tested using the previous problems. Both 0 and 1 transfers were allowed in the trip.

Table 4 shows the number of problems solved and the average number of solutions per problem. These quantities are shown for every topology (1, 2, and 3 WebAgents) and have been broken down for every type of trip: **N**ational/**E**uropean/**I**nternational.

Two points deserve to be highlighted. With respect to the two homogeneous sources (AMF and IBE), AMF is better in terms of problems solved, because it is a metasearcher engine which is able to retrieve information from different companies. It solves 15/21 problems (0/1 transfers, respectively). With respect to combinations of more than one Web agent, the three agent configuration (AMF-IBE-RNF) manages to solve 19/29 (0/1 transfers) out of the 30 problems. However, the integration of only two of them (AMF-IBE) allows to find many more solutions per problem because even though AMF is a metasearcher, it does not consider all solutions. This is specially true when 1 transfer is allowed. For instance, for 1 transfer, AMF-IBE obtains 195.4/122.6/78.3 solutions per problem for national, european, and international problems, respectively. On the other hand, the standalone WebAgents AMF and IBE obtain 28.4/41.3/19.2 and 40.1/25.8/14.2 solutions per problem, respectively.

In addition to showing that more solutions can be found, this experiment shows how the integration of heterogeneous sources allows to solve more problems as well, because new heterogeneous solutions (train + plane) are found that could not be retrieved from the homogeneous WebAgents alone. For instance, for 0 transfers, the AMF-IBE-RNF configuration solves 8/7/5 problems (National/European/International, respectively) whereas the best homogeneous configuration (AMF-IBE) solves only 5/6/4. For 1 transfer, results are even better: 10/10/9 vs. 5/10/6. Additionally, there is a high increase in the number of solutions returned.

In this paper, we have considered the number of solved problems and the number of solutions per problems. There can be two possible reasons for having unsolved problems:

Table 4 Number of solved problems (out of 30) using different topologies in MAPWEB-ETOURISM

		N° of problems solved		N° of solutions	
Topology type	Selected topology	0 Transfers N/E/I	1 Transfer N/E/I	0 Transfers N/E/I	1 Transfer N/E/I
1 WebAgent	AMF	5/6/4	5/10/6	9.6/6.8/1.5	28.4/41.3/19.2
	IBE	5/4/3	5/9/5	13.8/9.1/5.7	40.1/25.8/14.2
	RNF	8/2/0	10/3/0	10.8/1/0	36.5/8.4/0
2 WebAgents	AMF-IBE	5/6/4	5/10/6	18.4/11.9/6.1	195.4/122.6/78.3
3 WebAgents	AMF-IBE-RNF	8/7/5	10/10/9	29.2/12.9/6.1	830.8/540.5/398.7

on the one hand, there can be no possible solution within the information provided by the Web sources. On the other hand, and more unlikely, the heuristics used may filter out some possible solutions (this is not the case of the experiments reported here). Once the solutions have been obtained, they can be sorted by a user chosen criteria, like price or time. The only way to find the best quality solution is to obtain all possible solutions by trying all possibilities. However, due to the large number of data registers, we have decided to limit the number of solutions analyzed by means of special purpose heuristics, as described in Section 3.1. The heuristics have been carefully designed so that best quality solutions are not filtered out.

With respect to the on-line time required to obtain the total number of solutions, for all the 0-transfer problems, this time is smaller 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. In any case, the system returns a solution as soon as it is found, and in all cases, this time is smaller than 1 minute. Therefore, the approach is feasible as an on-line system.

5. Related work

Several systems, and techniques, have been designed to deal with heterogeneous information sources. These kinds of systems (SIMS [3, 4]), usually named mediators, implement several mechanisms that provide access to heterogeneous data and knowledge bases. These techniques can be used to build information agents, that are able to extract, query, and integrate data from electronic sources. Information agents have been used to implement different systems that are able to retrieve and integrate information from the WEB [14, 17]. The most important systems closer to our work are:

- Ariadne [16]: This system includes a set of tools to construct wrappers that make WEB sources look like relational databases. It also uses mediation techniques based on SIMS [3, 17]. The main focus of these systems is how to access the distributed information, so the integration problem is not a hard problem. However, besides accessing the appropriate information, we are interested in integrating the different sources and solve complex problems with the retrieved information.
- Heracles [2, 15]: This framework is used to develop different information assistant systems that employ a set of information agents (Ariadne, Theseus, Electric Elves). A dynamic hierarchical constraint propagation network (CPN) is used to integrate 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, or airports). In this framework the integration of the retrieved information is made by a CPN. Therefore, if the problem changes, the CPN needs to be rewritten by hand. MAPWEB is more flexible because it uses a planner to automatically generate the plans, which are the structures analogous to the CPN. For instance, if new transport sources like taxi or buses become available, it is only necessary to add a new planning operator for every new source and the PlannerAgent will use them to access these sources.
- Retsina [7, 23, 29]. Retsina is a well known multi-agent architecture that supports communities of heterogeneous agents. In this architecture coordination structures emerge from the relations between agents, rather than as a result of the imposed constraints of the infrastructure itself. Retsina does not employ centralized control within the MAS. This architecture implements distributed services that facilitate the interactions between agents, as opposed to managing them. This architecture has been successfully used to implement several MAS like MokSAF (logistics planning in military operations), or MOCHA (wireless, mobile communications) [19]. This architecture has been widely used in several domains like the WEB or for military applications. Our system parallels some of Retsina's ideas, but our agents can also use planning mainly in the Information Gathering framework, to determine which information agents would be queried and to integrate the heterogeneous gathered data into a detailed solution. Therefore, in our approach plans can be used as a template to coordinate the information agents and to guide the integration process.

WebPlan [12] is a WEB assistant for domain-specific search on the Internet based on dynamic planning and plan execution techniques. The existing planning system *CAPlan* [13, 32] has been extended in different ways in order to deal with incomplete information, information seeking operators, user interaction, and interleaving planning and execution. WebPlan is specialized in locating specific PC software on the Internet. Planning is used in this system to select the most appropriate sources to look for information, whereas MAPWEB uses planning to select the appropriate WEB sources and to build the solution to a user problem.

Finally, the travel assistant domain used as a testbed in this paper, has been widely used in the literature [2, 11, 22, 24, 28, 33].

6. Conclusions

In this paper, we have proposed to use planning for information gathering, to select and integrate information from heterogeneous Internet sources. From the experimental results of this paper, it can be shown how MAPWEB allows not only to gather information from different sources. but to solve more user problemas, and to find more solutions for them, by using sequences of planning operators (plans) to integrate data from heterogeneous sources.

The paper also shows that planning techniques make it easier to flexibly work with heterogeneous sources, because it is straightforward to relate a source with an specialized planning operator. New operators allow to find new types of solutions. In order to handle more complex problems, all that is required is to add new abstract planning operators and the appropriate WebAgents that provide the specific information.

Currently, we assume that the different sources contain all the information required to build a complete solution. But this is not always the case. For instance, not all travel information sources provide data about time and/or cost, and some assumptions will have to be made. Machine learning and statisticial techniques could be used to derive plausible values for missing data. Also, the Web is a dynamic environment and information sources can fail temporarily. We intend to treat this problem by means of local databases in the Web agents, that can store records retrieved at other times. This will also increase the uncertainty of plans provided to the user, as some plan steps will be based on old, possibly false, information. In the future, we intend to handle this and other types of uncertainty, so that possible plans can still be displayed to the user.

Also, although the system allows to add new information sources by extending the operator set and including new information agents, this process has to be carried out manually. In the future, we intend to automatically discover new information sources by taking advantage of new technologies related to the Semantic Web and Web Services. Currently, we are defining an ontology so that new Web sources can be more easily introduced in the system and to facilitate the exchange of information among agents.

References

1. Ambite JL, Knoblock CA (1997) Planning by rewriting: Efficiently generating high-quality plans. In: Proceedings of the 4'th National Conference on Artificial Intelligence
2. Ambite JL, Barish G, Knoblock CA, Muslea M, Oh J, Minton S (2002) Getting from here to there: Interactive planning and agent execution for optimizing travel. In: The 4'th Innovative Applications of Artificial Intelligence Conference (IAAI)
3. Arens Y, Chee CY, Hsu C-N, Knoblock CA (1993) Retrieving and integrating data from multiple information sources. International Journal of Cooperative Information Systems 2(2):127–158
4. Arens Y, Knoblock CA, Shen W-M (1996) Query reformulation for dynamic information integration. Journal of Intelligent Information Systems, Special Issue on Intelligent Information Integration 6(2/3):99–130
5. Camacho D, Aler R, Borrajo D, Molina JM (2005) A multi-agent architecture for intelligent gathering systems. AI Communications. The European Journal on Artificial Intelligence. (Ed. by IOS Press) To appear in Vol. 18(1)
6. Cohen PR, Perrault CR (1979) Elements of a planbased theory of speech acts. Cognitive Science 3(3):177–212
7. Decker K, Sycara K (1997) Intelligent adaptive information agents. Journal of Intelligent Information Systems 9:230–260
8. Fan Y, Gauch S (1999) Adaptive agents for information gathering from multiple, distributed information sources. In: Proceedings of 1999 AAAI Symposium on Intelligent Agents in Cyberspace. Stanford University
9. Finin T, Weber J, et al (1993) Draft specification of the KQML agent communication language 15 Jun
10. Finin T, Fritzson R, Mackay D, McEntire R (1994) Kqml as an agent communication language. In: Proceedings of the 3'rd International Conference on Information and Knowledge Management (CIKM94), Gaithersburg, Maryland, Association of Computing Machinery, ACM Press, New York pp 456–463
11. Fipa Foundation For. Fipa 97 draft specification part 4 personal travel assistance
12. Hüllen J, Bergmann R, Weberskirch F (1999) Webplan: Dynamic planning for domain-specific search in the internet. In: Workshop Planen und Konfigurieren (PuK-99)
13. Hüllen J, Weberskirch F (1999) Extracting goal orderings to improve partial-order planning. In: Workshop Planen und Konfigurieren (PuK-99)
14. Knoblock CA, Ambite JL (1997) In: Bradshaw J (ed), Agents for Information Gathering, chapter In Software Agents, AAAI/MIT Press, Menlo Park, CA
15. Knoblock CA, Minton S, Ambite JL, Muslea M, Oh J, Frank M (2001) Mixed-initiative, multi-source information assistants. In: The Tenth International World Wide Web Conference (WWW10). ACM, 1–5 May
16. Knoblock CA, Minton S, Ambite JL, Ashish N (1998) Modeling web sources for information integration. In: Proceedings of the Fifteenth National Conference on Artificial Intelligence, Madison, WI
17. Knoblock CA, Minton S, Ambite JL, Ashish N, Muslea I, Philpot A, Tejada S (2001) The ariadne approach to web-based information integration. International Journal of Cooperative Information Systems 10(1/2):145–169
18. Kushmerick N (2000) Wrapper induction: Efficiency and expressiveness. Artificial Intelligence 118(1–2):15–68
19. Lenox T, Hahn S, Lewis M, Payne T, Sycara K (2000) Task characteristics and intelligent aiding. In: Proceedings of the 2000 IEEE International Conference on Systems, Man, and Cybernetics IEEE pp 1123–1127
20. Levy AY, Rajaraman A, Ordille JJ (1996) Querying heterogeneous information sources using source descriptions. In: Proceedings

of the Twenty-second International Conference on Very Large Databases, Bombay, India. VLDB Endowment, Saratoga, California, pp 251–262

21. Oates T, Nagendra Prasad MV, Lesser VR (1994) Cooperative information gathering: A distributed problem solving approach. Technical Report 94-66, University of Massachusetts
22. Donie O'Sullivan et al. Experiences in the use of fipa agent technologies for the development of a personal travel application
23. Paolucci M, Kalp D, Pannu AS, Shehory O, Sycara K (1999) A planning component for retsina agents. In: Lecture Notes in Artificial Intelligence, Intelligent Agents VI
24. Ricci F, Werthner H (2002) Case base querying for travel planning recommendation. Information Technology and Tourism Journal 4(3/4):215–226
25. Sahuguet A, Azavant F (2001) Building intelligent web applications using lightweight wrappers. Data Knowledge Engineering 36(3):283–316
26. Salton G, McGrill MJ (1983) Introduction to modern information retrieval, McGraw Hill, Computer Science Series
27. Srinivasan P, Mitchell J, Bodenreider O, Pant G, Menczer F (2002) Web crawling agents for retrieving biomedical[-9pt] information. Available from http://dollar.biz.uiowa.edu/fil/Papers/bixmas.pdf
28. Staab S, Werthner H (2002) Intelligent systems for tourism. IEEE Intelligent Systems pp 53–66
29. Sycara K, Decker K (1996) Distributed intelligent agents. IEEE Expert 11(6):36–46
30. Veloso M, Carbonell J, Perez A, Borrajo D, Fink E, Blythe J (1995) Integrating planning and learning: The prodigy architecture. Journal of Experimental and Theoretical AI 7:81–120
31. Veloso M (1994) Planning and learning by analogical reasoning, Springer-Verlag
32. Weberskirch F (1995) Combining snlp-like planning and dependency-maintenance. Technical report, Technical Report LSA-95-10E, Centre for Learning Systems and Applications, University of Kaiserslautern, Germany
33. Yim HS, Ahn HJ, Kim JW, Park SJ (2004) Agent-based adaptive travel planning system in peak seasons. Expert Systems with Applications 27(2):211–222