A Cooperative Planning Algorithm to Improve Performance in Web Domains

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

Computer Science Department, Universidad Carlos III de Madrid
Madrid, 28911, Spain, Fax: (34) 91-6249129
E-mail: {dcamacho, dborrajo, molina}@ia.uc3m.es, aler@inf.uc3m.es

Abstract—
In this paper, we present MAPWEB , a multiagent framework that integrates planning agents and WEB information retrieval agents. The goal of this framework is to deal with problems that require planning with information to be gathered from the WEB. Because of flexibility and efficiency reasons, MAPWEB decouples planning from information gathering, by splitting a planning problem into two parts: solving an abstract problem and validating and completing the abstract solutions by means of information gathering. Here, we focus on the planning process in order to improve its efficiency. There are two ways of improving the efficiency of the MAPWEB planning algorithm: by accelerating the planning process itself and by storing previously solved planning problems. The first issue has been achieved by designing a cooperative planning algorithm that allows a set of planning agents to share plans and cooperate while planning to gain in efficiency. The second issue is currently being designed and developed to allow the planning agents to reuse the acquired knowledge. Finally, this paper presents the experimental evaluation of the cooperative planning process when it is used by the planning agents.

Keywords— **Multi-agent Systems, Intelligent Software Agents, Distributed Planning, Cooperative Planning.**

I. INTRODUCTION

In recent years there has been a lot of work in Web information gathering using Artificial Intelligence (AI) techniques [1], [2], [3], [4], [5]. AI 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 [5]. Many different kinds of systems, named *mediators*, have been developed that integrate information from multiple distributed and heterogeneous information sources, like database systems, knowledge bases, or electronic repositories. An example is the *SIMS* [3] architecture. In order these systems to be 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], [6] builds queries by using planning techniques. Heracles uses a constrained network to guide the query process [4]. There are other related examples of information gathering systems like Ariadne [3] or WebPlan [2].

Some of these previous 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. Our previous work has presented MAPWEB, an information gathering system that also uses planning, but with a different purpose (some preliminary work can be found in [7], [8]). MAPWEB uses planning for both deciding the appropriate generic sources to query, and solving planning problems. For instance, in a travel planning assistant domain (e-tourism) [8],[1] where the user needs to find a plan to travel between several places, each plan not only defines what steps the user must perform, but which information sources should be accessed. If a step in the plan is to go from A to B by plane, the system provides to the user the information of what airplane companies should be consulted for further information. This domain is similar to the travel planning assistant built using the Heracles framework. However, Heracles constrained network, which is a kind of plan schema, needs to be reprogrammed everytime the planning domain changes. MAPWEB is more flexible by using planning techniques to automatically generate the plans, when the domain definition changes.

Because of planning being a hard task, MAPWEB decouples it into two processes: creating abstract plans (without accessing the Web) and then instantiating them by using the appropriate Web sources. Time is saved because generating abstract plans is less time consuming and querying the Web can be guided by domain dependent heuristics. However, the problem is still too large if problems are complex. There are two ways of improving MAPWEB efficiency: in the abstract planning process and in the Web access process. This paper will focus in the former, and describes the cooperative planning algorithm which is used by the reasoning agents (PlannerAgents) in MAPWEB to solve the user problems. Planning is used to decompose the problems (using domain dependent methods) and then distributing the work to different agents with planning skills. Finally, several experiments have been carried out to measure the increase in performance, and analyze in which cases this improvement is obtained.

The paper is structured as follows. Section II describes the architecture of MAPWEB , the multi-agent framework used to test the proposed algorithm. Section III describes the cooperative planning algorithm used by the planner agents in MAPWEB. Section IV

[1] This domain is a modified version of the Logistics domain [9].

shows an experimental evaluation for several multi-agent configurations tested. Finally, Sections V and VI summarize the conclusions and future lines of work.

II. MAPWeb Architecture

This section describes the kind of agents who are involved in MAPWeb. The main goal of the agent society is to solve problems by using several AI classical techniques with the data retrieved from the Web. So first, the multi-agent society will be described and how these agents using their skills find different solutions for a given problem. Second, the cooperative planning algorithm that is used to integrate classical planning with the Web retrieved information will be described in section III.

MAPWeb is structured into several layers whose purpose is to isolate the user from the details of problem solving and Web information retrieval. More specifically, we considered three layers between users and the Web: *the Interface or User Agents* that pay attention to the users queries, *the Reasoning Agents* that actually only includes a set of planning agents, and finally *the Web Access Information Agents* that are specialized in retrieving the information. This three-layer agent architecture can be seen in Figure 1.

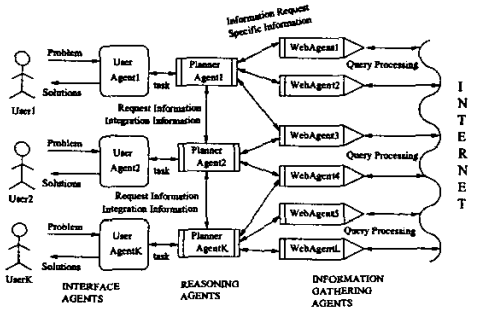

Fig. 1. MAPWeb three-layer architecture.

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 the solution(s) found by the system to the users. When an UserAgent receives problem queries from the users, it sends them to the PlannerAgents and when they answer back with the plans, the UserAgent provides the solutions to the user. This agent allows the user to change the behaviour of the multi-agent system modifying parameters such as the time response in the system, or the number of desired solutions.
- **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 by the PlannerAgents 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 by using a wrapper approach.

These agents cooperate as follows (Section III and Figure 2 provide a detailed description of this process). First, the user interacts with the UserAgent to input the query. In the case of the e-tourism application, the query captures information such as the departure and return 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. With the specific information provided by the user and with the abstract representation of the problem, the PlannerAgent tries to divide the problem in subproblems and search for solutions using its planning skills, and the cooperation with other PlannerAgents in the society. The planning skill includes two modules:

- A non-linear planner generates several abstract solutions for the user query. The planning operators in the abstract solutions require to be completed and validated with actual information which is retrieved from the Web. Therefore, the PlannerAgent sends information queries to specialized WebAgents, that return several records for each information query.
- A module that allows the cooperation with other agents in the society, especially with other PlannerAgents to request solving subproblems using their planning skills, and with the WebAgents to request for specific information to complete the abstract solutions.

When the solutions to the subproblems are found, the PlannerAgent who initially received the problem, integrates and validates the solutions and returns them to the UserAgent, which in turn displays them to the user.

Finally, MAPWeb agents use a subset of the KQML Speech Acts [11] as the communication language, to coordinate and cooperate among the agents. Some of the implemented performatives are standard, and the other have been built to solve different problems detected in the multi-agent architecture. Tables I, and II show the subset of the KQML of standard and adapted performatives used in MAPWeb.

Name	Meaning
achieve	S wants R to achieve something
insert	S asks R to add some contents to its VKB
tell	the sentence in S's VKB

TABLE I
SUBSET OF STANDARD KQML PEFORMATIVES USED BY MAPWeb AGENTS.

In both tables, S and R represent the *Sender* and the *Receiver* agents involved in the communication process, and VKB represent the *Virtual Knowledge Base* that any agent manages to model its environment. These performatives are used by different agents to implement several processes like:

Name	Meaning
request-info	S asks to R for some information
tell-info	R answers to R with some information
finish	R does not have more information

TABLE II
ADAPTED PEFORMATIVES USED BY MAPWEB AGENTS.

• (*achieve, tell*): are used by different agent to require the execution of a task and to answer for the obtained results.
• (*request-info, tell-info, finish*): are used between agents to implement a protocol that allows them to send all the desired information in several steps.

III. COOPERATIVE PLANNING ALGORITHM

Since it is possible to use several PlannerAgents when a MAPWEB agents configuration is created, they will use a cooperation process that allows to achieve two goals: minimize the computational effort in the solving process task; and obtain better and faster solutions through the acquired experience of other agents in the system. This section shows how the PlannerAgents achieves those goals, by using a planning algorithm and case-based techniques. The algorithm is divided into three main processes:
1. Search, Retrieval and Adaptation of old cases, or solutions, stored by the PlannerAgents. Those solutions can be a complete (or a part of a) solution.
2. Solving process using the planning skills for the global problem or the subproblems [7], if no solution(s) is found in the Case Base. This process needs a set of subprocesses to allow the cooperation with the WebAgents that will search in the WEB for the necessary information to complete the abstract solutions.
3. Integration of the different partial solutions to generate the set of solutions, to the requested problem.

Figure 2 shows the information flow in the PlannerAgent tasks. These processes begin when any PlannerAgent receives a query from another agent in the system. Then, the flow can be described as:
1. A PlannerAgent receives a problem from another agent in the system.
2. The PlannerAgent will search for the appropriate plans in its plan base. If it finds a solution for the problem request, it will send the solutions as the answer.
3. If the PlannerAgent does not find a solution, it will search for solutions to *similar* problems, which will be subsequently adapted and tested for validity.
4. If the agent found no solution, it will ask for help to other agents in the system. Other agents in the system will cooperate using their own knowledge to search for solution(s) that finally will be sent to the PlannerAgent that required them.
5. If no valid solution was obtained from the other agents, the agent will use its own problem solving skills.
6. The solutions found will be integrated and sent to the agent that requested them.

This algorithm uses the acquired knowledge (old successful plans) by PlannerAgents to improve the solving process. Any PlannerAgents can use several knowledge sources in the whole solving process:
• *Plan Base.* They are used by any PlannerAgent to learn from its previous experience in problem solving. They can store both complete plans and abstract plans (without details).
• *Other PlannerAgents.* If a PlannerAgent has no stored information in its plan base, and has to carry out several tasks, it can ask for help to other PlannerAgents in the systemm. Those agents can use their own plan bases and their planning skills. This avoids that all the planning workload is carried out by a single PlannerAgent.
• *Prodigy4.0.* If no solution was found by other means, the agent can use its own planning skills.
• *WebAgents.* Once one or several abstract plans are ready, and no specific information was found to complete and validate them, the appropriate WebAgents will be asked to perform this task.

Once the solving process has been described, we will describe how the plan bases are used by the PlannerAgents. A plan base is used by each PlannerAgent for learning from its own experience. It allows to store both complete plans and abstract plans. Thus, the PlannerAgents can search for old plans to solve new problems, and, therefore, avoid spending time while planning. In the extreme, if the PlannerAgent could always retrieve complete plans, then cooperation with other agents, like WebAgents, would not be necessary.

Whenever a PlannerAgent stores a successful plan, it creates two indices to represent it. The first one, called the `Goal-index`, is used to represent the specific information associated to the plan that was stored. The second one, called `Abstract-Goal-index`, allows to represent the abstract plan that includes the previous information. Those indices are built as follows. Once the problem has been solved, the goals of the problem are associated with a key that identifies them, which is then introduced in the plan base. Figure 3 shows two successful plans that are stored in the plan base. The operators are instantiated with specific information (`records`) that could be shared among different plans. The plan is stored and characterised using the sequence of operators and the associated records. As it is shown in Figure 3, Plan-1 and Plan-2 share specific information that was retrieved by the WebAgents. This representation allows to store the plans in two tables, that facilitate the retrieval of complete plans or only parts of a specific (or abstract) plan.

For instance, let us consider a two-goal problem like "travel from Madrid to Barcelona by plane *and* book a room for two nights". The indices would be built by using the parameters [airplane, MAD, BCN, hotel, BCN, 2], which contain the specific information of the two goals to fulfill (MAD and BCN represent Madrid and Barcelona, respectively). Table III shows a representation of these indices for a set of plans stored by the PlannerAgent.

Once a set of abstract plans (or subplans that solve a subproblem) and the asociated records has been obtained (from either other PlannerAgents or the We-

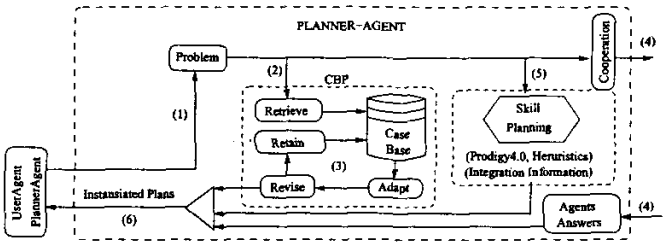

Fig. 2. Information flow in a PlannerAgent when it receives a problem request from another agent in the system.

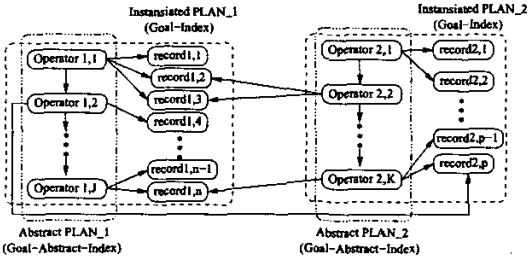

Fig. 3. Relationship between abstract and specific information for the stored plans in the plan base.

Abstract-Goal-index	Goal-index
[travel,city1,city2 / book-hotel-room, city2]	[airplane,MAD,BCN / hotel,BCN]
[travel,city1,city2 / travel,city1,city2 / book-hotel-room,city2]	[airplane,MAD,VLC / airplane,VLC,BCN / hotel,BCN]
[travel,city1,city2 / travel,city1,city2 / book-hotel-room,city2]	[airplane,MAD,ALC / airplane,ALC,BCN / hotel,BCN]
...	...
[travel,city1,city2 / book-hotel-room,city2 / travel,city2,city3 / book-hotel-room,city3 / travel,city3,city1]	[airplane,JFK,MAD / hotel,MAD / airplane,MAD,BCN / hotel,BCN / airplane,BCN,JFK]
[travel,city1,city2 / book-hotel-room, city2 / travel,city2,city3/ [travel,city3,city4 / book-hotel-room,city4 / travel,city4,city1]	[airplane,JFK,MAD / hotel, MAD / airplane,MAD,VLC / airplane,VLC,BCN / hotel,BCN / airplane,BCN,JFK]
...	...

TABLE III
KEYS AND INDICES USED TO STORE INSTANTIATED PLANS.

bAgents), it is necessary to generate the set of global solutions that define the answer returned by the agent. So, if the problem is made of several stages, specific solutions will be obtained for each one of them. Any stage for which there is no solution will be labelled as unsolved. Then, in order to generate the global solutions, the next algorithm is used:

1. Once all queried agents have answered, those operators in the plan sequence that were fully instantiated will be selected. Every such operator will have associated one or more (non-repeated) records.
2. Every instantiated operator is associated to its related stage. Those stages whose operators have been completely instantiated are considered as correct.
3. Only plans whose all stages are correct are considered correct.

IV. Experimental Evaluation

This section evaluates empirically the previous cooperative planning algorithm. The experiments measure how it is possible to reduce the planning complexity of the problem by dividing it and distributing the work load using several agents. Different WebAgents will be used to retrieve heterogeneous information, and two different configurations, or topologies, of MAPWeb will be tested. The first topology will only have a PlannerAgent an all the planning problems will be achieved by itself. In the second topology two PlannerAgents will be used and the cooperation among them is possible. Table IV shows the number of PlannerAgents and the specific WebAgents employed to build the different topologies.

A set of ten problems will be posed for each topology. They are travel problems, where the different legs in the travel need planning and specific information to execute completely the trip. Table V shows a description of the test problems used to evaluate the PlannerAgent planning algortithm. The problems have an increasing complexity due the number of legs of the travel (from two legs to five).

Topology	N° PlannerAgents	Cooperation	WebAgents
1	1	No	Iberia, 4Airlines, Amadeus-Flights Amadeus-Hotels, Amadeus-RentCar
2	2	Yes	Iberia, 4Airlines, Amadeus-Flights Amadeus-Hotels, Amadeus-RentCar

TABLE IV

SPECIALIZED AGENTS IN MAPWEB USED FOR THE EXPERIMENTAL EVALUATION OF THE COOPERATIVE DISTRIBUTED PLANNING ALGORITHM.

N°	Leg 1	Leg 2	Leg 3	Leg 4	Leg 5
1	Op:trip (Madrid,Paris)	Op:book room Paris			
2	Op:trip (London,Paris)	Op:book room Paris			
3	Op:trip (Valencia,Bruselas)	Op:book room Bruselas	Op:trip (Bruselas,New York)		
4	Op:trip (Bruselas,New York)	Op:rent car Nueva York	Op:trip (New York,Bruselas)		
5	Op:trip (Madrid, Sao Paulo)	Op:trip (Sao Paulo,Caracas)	Op:trip (Caracas,Madrid)		
6	Op:trip (Valencia,Boston)	Op:rent car Boston	Op:book room Boston	Op:trip (Boston, Lisboa)	
7	Op:trip (Milan,Berlin)	Op:book room Berlin	Op:trip (Berlin,Alicante)	Op:rent car Alicante	
8	Op:trip (Madrid, Valencia)	Op:book room Valencia	Op:trip (Valencia,Barcelona)	Op:trip (Barcelona,Madrid)	
9	Op:trip (Madrid,Barcelona)	Op:book room Barcelona	Op:trip (Barcelona,Paris)	Op:book room París	Op:trip (Paris,Madrid)
10	Op:trip (Madrid,Barcelona)	Op:trip (Barcelona,Milan)	Op:book room Milan	Op:trip (Milan,Paris)	Op:trip (Paris,Madrid)

TABLE V

PROBLEMS USED TO EVALUATE THE PLANNERAGENTS DISTRIBUTED PLANNING ALGORITHM IN MAPWEB.

Different features have been measured to obtain the behaviour of the algorithm. The features can be summarized in:

- **Independent variables.** Test problems have legs that involve a travel between two cities. Each leg can be performed by a number of possible transfers (intermediate cities) to be used for travelling between origin and destination cities. In the experiments the number of transfers considered were 0-transfers and 1-transfer.
- **Dependent variables.** We measured the number of solutions found by MAPWEB (N_{sol}), the number of final plans that the PlannerAgent validates (N_{PA-dev}), the percentage of legs distributed among the different agents, and the time spent to solve the problem by the multi-agent topology (T_{resp}).

Tables VI and VII show the experimental results for the requested problems. These tables display the empirical evaluation for the topologies described in Table IV, for problems with 0-transfers and 1-transfer allowed. The first two columns in the tables show the number of the experiment and the number of legs in the travel. The next columns show the dependent variables (number of instantiated solutions, number of solved problems and the time to answer) for each topology: `[topology1/topology2]`. Finally the last column shows the time saving when both topologies are compared.

V. DISCUSSION AND CONCLUSIONS

When Tables VI and VII are analyzed, it can be concluded that by using several PlannerAgents, the workload is distributed among them, and the problem solving time is reduced. Two factors influence the workload:

- The abstract planning process is distributed, which reduces the planning time.
- The query generation of a process is also distributed, which reduces the Web access time.

However, the time reduction depends on the complexity of the problem (i.e. number of stages). If there are few stages or the complexity of every stage is small, the gain is also small. In fact, in some cases the overload due to the agent communication process is larger than the gains due to parallelization, and the total gain is negative. For instance, there is no gain if only 0-transfers and 2 stages problems are sent to the system because the negotiation required to distribute the workload takes longer than the actual savings obtained by parallelization. However, as the system solves harder problems, the distribution of work increases up to 35% the gain in time to answer.

VI. FUTURE WORK

We can summarize the main future lines as follows:

1. So far, we have only used WebAgents for airplane, hotels and rental car companies. However, MAPWEB is very well suited for integrating information coming from

Prob.	N° Legs	N_{sol}	N_{PA-dev}	% legs shared	T_{resp} (min.)	% time saving
1	2	[36/36]	[2/2]	[-/50%]	[9.00/9.12]	-1.33%
2	2	[42/42]	[2/2]	[-/50%]	[9.83/9.97]	-1.40%
3	3	[0/0]	[0/0]	[-/33%]	[11.30/11.15]	1.33%
4	3	[0/0]	[0/0]	[-/33%]	[11.70/10.95]	6.41%
5	3	[0/0]	[0/0]	[-/33%]	[12.40/9.89]	20.24%
6	4	[0/0]	[0/0]	[-/33%]	[11.99/9.36]	21.93%
7	4	[0/0]	[0/0]	[-/50%]	[13.30/10.00]	24.81%
8	4	[78/78]	[8/8]	[-/50%]	[17.32/11.35]	34.58%
9	5	[103/103]	[8/8]	[-/40%]	[20.63/14.53]	29.71%
10	5	[92/92]	[4/4]	[-/40%]	[23.50/16.30]	30.63%

TABLE VI

NUMBER OF SOLUTIONS, NUMBER OF SOLVED PROBLEMS AND TIME RESPONSE IN MAPWEB, USING TRAVEL PROBLEMS WITH SEVERAL HETEROGENEOUS LEGS AND USING 0-TRANSFERS IN TRAVEL-LEGS.

Prob.	N° Legs	N_{sol}	N_{PA-dev}	% legs shared	T_{resp} (min.)	% time saving
1	2	[95/95]	[4/4]	[-/50%]	[20.00/19.67]	1.65%
2	2	[104/104]	[4/4]	[-/50%]	[22.83/21.64]	5.21%
3	3	[136/136]	[5/5]	[-/33%]	[24.23/20.98]	13.41%
4	3	[340/340]	[7/7]	[-/33%]	[35.20/30.00]	14.77%
5	3	[40/40]	[2/2]	[-/33%]	[29.25/21.78]	25.54%
6	4	[190/190]	[2/2]	[-/50%]	[28.15/20.34]	27.74%
7	4	[430/430]	[6/6]	[-/50%]	[36.90/25.18]	31.76%
8	4	[1980/1980]	[60/60]	[-/50%]	[58.15/39.24]	33.03%
9	5	[1357/1357]	[47/47]	[-/60%]	[55.36/32.92]	40.53%
10	5	[515/515]	[8/8]	[-/60%]	[43.62/28.91]	33.72%

TABLE VII

NUMBER OF SOLUTIONS, NUMBER OF SOLVED PROBLEMS AND TIME TO ANSWER IN MAPWEB, USING TRAVEL PROBLEMS WITH SEVERAL HETEROGENEOUS LEGS AND USING 1-TRANSFER IN TRAVEL-LEGS.

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.

2. Reuse of information stored in WebAgents, given that these agents can learn from experience, and reuse information retrieved previously to reduce WEB access.

VII. ACKNOWLEDGEMENTS

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

REFERENCES

[1] José L. Ambite and Craig A. Knoblock, "Planning by rewriting: Efficiently generating high-quality plans," in *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, 1997.

[2] J. Hullen, Ralph Bergmann, and F. Weberskirch, "Webplan: Dynamic planning for domain-specific search in the internet," in *Workshop Planen und Konfigurieren (PuK-99)*, 1999.

[3] Craig A. Knoblock and Steven Minton, "The ariadne approach to web-based information integration," *IEEE Intelligent Systems*, vol. 13, no. 5, September/October 1998.

[4] Craig A. Knoblock, Steve Minton, José-Luis Ambite, Maria Muslea, Jean Oh, and Martin Frank, "Mixed-initiative, multi-source information assistants," in *The Tenth International World Wide Web Conference (WWW10)*. ACM, May 1-5 2001.

[5] Eric Lambrecht and Subbarao Kambhampati, "Planning for information gathering: A tutorial survey," Tech. Rep., Arizona State University, May 1997, ASU CSE Techincal Report 96-017.

[6] Jose Luis Ambite and Craig A. Knoblock, "Flexible and scalable query planning in distributed and heterogeneous environments," in *Proceedings of the Fourth International Conference on Artificial Intelligence Planning Systems*. AAAI, Pittsburgh,Pennsylvania, 1998.

[7] David Camacho, Daniel Borrajo, José Manuel Molina, and Ricardo Aler, "Flexible integration of planning and information gathering," in *Proceedings of the European Conference on Planning (ECP-01)*, Toledo, Spain, September 2001, Springer-Verlag. Series LNAI.

[8] David Camacho, Daniel Borrajo, and Jose Manuel Molina, "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, no. 4, pp. 385–390, December 2001.

[9] Manuela Veloso, *Planning and Learning by Analogical Reasoning*, Springer-Verlag, December 1994.

[10] M. Veloso, J. Carbonell, A. Perez, D. Borrajo, E. Fink, and J.Blythe, "Integrating planning and learning: The prodigy architecture," *Journal of Experimental and Theoretical AI*, vol. 7, pp. 81–120, 1995.

[11] Tim Finin, R. Fritzson, D. Mackay, and R. McEntire, "Kqml as an agent communication language," in *Proceedings of the Third International Conference on Information and Knowledge Management (CIKM94)*, Gaithersburg, Maryland, 1994, New York: Association of Computing Machinery, pp. 456–463, ACM Press.