Truth Maintenance Task Negotiation in Multiagent Recommender System for Tourism

Fabiana Lorenzi
Instituto de Informatica, UFRGS
Caixa Postal 15064 CEP 91.501-970
Porto Alegre, RS, Brasil
Universidade Luterana do Brasil
Av. Farroupilha, 8001 - CEP 92420280, Canoas, RS, Brasil
lorenzi@ulbra.br

Ana L.C. Bazzan and Mara Abel
Instituto de Informatica, UFRGS
Caixa Postal 15064 CEP 91.501-970
Porto Alegre, RS, Brasil

Abstract

This paper describes a multiagent recommender approach based on the collaboration of multiple agents exchanging information stored in their local knowledge bases. A recommendation request is divided into subtasks handled by different agents, each one maintaining incomplete information that may be useful to compose a recommendation. Each agent has a Distributed Truth Maintenance component that helps to keep the integrity of its knowledge base. Further, we show a case study in the tourism domain where agents collaborate to recommend a travel package to the user. In order to help the coordination among the agents during the recommendation, the Distributed Constraint Optimization approach is applied in the search process.

Introduction

Recommender systems are being used in e-commerce web sites to help customers in selecting products more suitable to their needs. In the last years, the internet has grown exponentially and the information overload problem has appeared. In order to aggregate information and to match the recommendations with the information people is looking for, Recommender Systems have been developed (Resnick *et al.* 1994).

Despite recommender systems being efficient, for some recommender applications it is possible that a single information source does not contain the complete information needed for the recommendation. For example, in the tourism domain, the travel package recommendation is composed by several information components such as flights, hotels and entertainment and specific knowledge to assemble all the components is necessary. Due the distributed nature of the information, multiagent systems (MAS) are promising to retrieve, filter and use information relevant to the recommendation. MAS can be used to avoid unnecessary processing and can be built to deal with dynamic changes in the information source.

We propose a multiagent recommender approach where the agents work in a distributed and cooperative way, sharing and negotiating knowledge with the global objective of recommending the best travel package to the user. In order to achieve this goal, this approach uses distributed truth maintenance system (TMS) as in (Huhns & Bridgeland 1991) and a distributed constraint optimization (DCOP) (Modi *et al.* 2003) applied in the tourism domain to be validated in a real-world application. The distributed TMS helps to keep the integrity of knowledge base of each agent, while the distributed constraint optimization approach is used to help the coordination among the agents during the search processes.

Multiagent Recommender Systems

The multiagent recommender system uses agents to help in the solution process, trying to improve the recommendation quality. The agents cooperate and negotiate in order to satisfy the users, interacting among themselves to complement their partial solutions or even to solve conflicts that may arise.

In (Macho, Torrens, & Faltings 2000) the authors present a multiagent recommender system that arranges meetings for several participants taking into account constraints for personal agendas. In this system, three different agents were proposed: the *personal assistant agent* that is the interface between the user and the MAS. The *flight travel agent* that is connected to a database of flights; and the *accommodation hotel agent* that is reponsible to find an accommodation on the cities involved in the meeting.

Another example of multiagent case-based recommender system is CASIS (Lorenzi, Santos, & Bazzan 2005). The authors proposed a metaphor from swarm intelligence to help the negotiation process among agents. The honey bees' dancing metaphor is applied with case-based reasoning approach to recommend the best travel to the user. The recommendation process works as follows: the user informs his/her preferences; the bees visit all cases in the case base and when they find the best case (according to the user's preferences) they dance to that case, recruiting other bees to that case; and the case with the most number of bees dancing for it is the one recommended to the user. The advantage of this application is that the bees always return something to recommend to the user. Normally, case-based recommender systems use pure similarity to retrieval the best cases. The recommendation results depend on the defined threshold and sometimes the system does not find cases that match the threshold. Despite the controversy that sometimes is better not recommend instead of recommend wrong products,

this is not true in the tourism domain. CASIS system always returns some recommendation, which is specific for the current season, given the dynamic nature of the approach). The disadvantage of this system is that the case-base is centralized. It is not possible to search information distributed in other sources.

Truth Maintenance Systems

Truth Maintenance Systems (TMS) (also called belief revision or reason maintenance systems) are algorithms that help to maintain the knowledge base integrity. The first proposed TMS is the Justification Based Truth Maintenance System (JTMS) presented by Doyle (Doyle 1979). The JTMS is divided in two components: a problem solver which draws inferences (beliefs) and a TMS which records these inferences (called justifications). Every datum has a status associated to it: in (believed) or out (not believed). The TMS believes in a datum if it has an argument for the node and the beliefs in the nodes involved in the argument. The created justifications can be use to provide an explanation to the user. The disadvantage of this algorithm is that only one point of the search space can be examined at a time (specific context).

The evolution of TMS leaded to the Assumption-based TMS (ATMS) presented by deKleer (de Kleer 1986). The ATMS is based on manipulating assumption sets. In the ATMS algorithm, every datum is labeled with the sets of assumptions and they are computed by the ATMS from the justifications supplied by problem solver. In this approach, the belief status can be a set of assumptions instead of just labels of type IN or OUT. This brings flexibility to the algorithm because it is possible to deal with multiple beliefs status.

A Distributed TMS (DTMS) was proposed in (Huhns & Bridgeland 1991), which aimed at seeking local consistency for each agent and global consistency for the data shared by the agents. Given a network of several agents, they interact by exchanging data. Each agent has two kinds of data in its knowledge base: shared and private. A private data becomes a shared one when the agent informs another agent. The concern of this approach is to maintain the consistency of the shared data, because they affect the problem-solving process of another agent. In this approach, the IN label was refined in two additional labels: *internal* and *external*. An *internal* data is one that is believed to be true and has a valid justification. An *external* data is believed to be true but it does not need to have a valid justification. Thus, the justification of an *external* data is "the other agent told me". An advantage of this approach is that the TMS is only triggered when there is a change in belief status of shared data, which means it does not introduce overhead for monotonic reasoning among agents. The authors presented an example of DTMS algorithm where two agents (an investor and a stockbroker) interact. The investor agent asks to the other agent to recommend a stock.

A multiagent truth maintenance system can be considered as a distributed version of a truth maintenance system. In Multiagent TMS, there are multiple agents and each one has its own truth maintenance system. Each agent has uncertain data that can be IN or OUT (believed or not believed) and each shares some data with other agents. Each agent must determine the label of its data consistently, and shared data must have the same label.

A Multiagent System with Truth Maintenance Applied to the Tourism Domain

Planning a travel is not an easy task for the travel agent. He needs to know every detail about the destination chose by the passenger and all details involving the whole trip such as the timetable of attractions, hotels or flights.

Architecture of the Agents

Using this travel recommendation example, we have created a basic multiagent scenario with a group of agents in the community with a common global goal (the recommendation) and separate individual goals (the component that each one needs to search). A community *C* consists of *n* agents $a_1, a_2, ..., a_n$, each located inside a predefined range of action *R*. There are two different types of agents within the community: the Assembler - *Asm* and the Searcher - *Src*.

Asm agents are responsible for the communication with the user and to show the final recommendation. It creates a list of tasks (flights, hotels and attractions) that is sent to the *Src* agents. The community may have several *Asm* agents and each one creates a different recommendation tailored to each user. *Scr* agents are responsible for choosing a task from a list.

The *Src* agent is represented by $Src = (P, LocalKB, TMS)$. *P* is the agent's profile defined as $P = (id, active, tcurrent, tsolved, tfreq, confind)$; *id* is the identification of the agent, *active* indicates if the agent is *inside* or *outside* the community, *tactual* is the current chosen task, *tsolved* is a list of the 10 recently solved tasks, *tfreq* is a list with the 10 most frequently solved tasks and *confind* is the confidence index of the agent in that type of task.

The confidence index helps the agent to specialize in some type of task and to help the maintenance of the agent's knowledge base. The specialization of the agent in a type of task improves the system's performance but more important than that, it leads to a more efficient recommendation because the agent becomes an expert in that type of recommendation.

This confidence index is calculated taking into account the frequency of the task's type, the *tsolved* list and the *tfreq* list. The agent calculates the confidence index every time it has to choose a task to perform and it checks which task available is better for it. After perform the task, the agent checks if that task has increased or not its confidence index and it decides to keep or not the task in its lists. This behavior is important to avoid that the agent specializes itself in seasonal recommendations.

LocalKB is the knowledge base (KB) of the agent. Each agent has in its KB the following attributes: idt, $typet, timecommit, requirements, solution.$ *Idt* is the task's identification, *typet* is the task's type (initially limited to flight, hotel or attractions), *timecommit* is the time the agent took to perform the task, *requirements* represent the user's preferences, i.e., the user's query. Every time the

agent saves a new task in its KB it saves the whole user's query as well. *Solution* represents the information the agent knows to solve that task.

The agent's KB is increased according to the number of tasks the agent solves and saves and it is necessary to control the size of its knowledge base. As bigger the KB becomes, worst is the search performance. In the other hand, a small KB forces the agent to search in the community or into the web, which also decreases the search performance. The confidence index helps the agent to decide which task should be saving in its KB.

TMS is the reason maintenance system component, responsible for maintaining the integrity of the transferred information among the agents and their KBs. The attributes of the TMS are: $agent, knows, solution$, that means which other *agent knows* the *solution* to the task it has to perform. This TMS component helps in the search process. When the agent does not have information in its own knowledge base, it can communicate to other agents to discover if any agent has the information it needs. Inconsistencies can appear during this communication and the knowledge bases can lost integrity.

Truth Maintenance System Component The distributed scenario described in the previous subsection deals with the problem of missing information during the recommendation process, allowing that each agent communicates to other agents ands ask for information needed. However this communication leads to another problem: the lost of integrity of each knowledge base. To avoid this problem, each agent has a TMS component that is responsible for keeping the integrity of the agent's knowledge base.

The TMS component also helps to keep the integrity of the recommendation. Sometimes agents have to guess information during the recommendation process. We adopted the ATMS (de Kleer 1986), creating a rule base - *R*, consisting of causal and logical rules derived from prior experience of the travel agents, a premise set - *I* that are the user's preferences, and an assumption set - *A*, that contains the assumptions of the agents. Based on these model components, the agents derive propositions - *rec* that represent parts of the final recommendation that will be presented by the *Asm* agent.

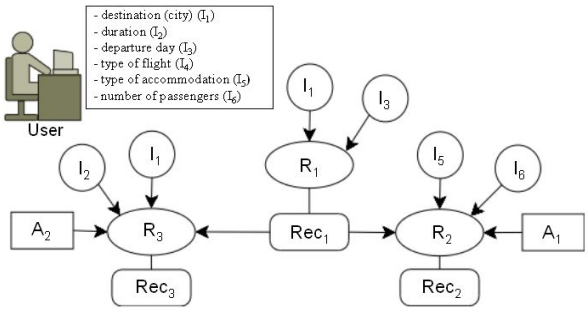

Figure 1: ATMS representation in the multiagent scenario

Figure 1 shows a piece of the ATMS representation. The first proposition (rec_1) depends on two preferences informed by the user: the city destination (I_1) and the departure day (I_3). The second proposition (rec_2) finds the best hotel for the user. The premises *day* and *time of arrival* come from the flight's information. If the user did not inform the type of accommodation then the agent uses the assumption A_1, assuming that the type of accommodation is economic. The proposition rec_3 depends on the destination, the duration and the flight's information. To search for the best attractions, for example, the agent needs information about the flights (duration of the package for example). Trying not to delay the recommendation process, this agent assumes a standard duration and starts its process, looking for attractions in the destination that fit in the period the user wants to stay there. The fact of assuming some information is very common in the real travel recommendation process. In the most cases the passenger is not sure about what he wants to do exactly, he depends on the travel agent recommendation. Figure 2 shows some examples of rules that have been defined to the travel package recommendation process. Each rule is used to lead to a proposition.

R1 = I1 (destination) ∧ I3 (departure) → rec1 (attraction)
R2 = I5 (accommodation) ∧ I6 (number of passengers) ∧ A1 (assumption) → rec2 (hotel)
R3 = I2 (duration) ∧ I1 (destination) ∧ A2 (assumption) → rec3 (flight)

Figure 2: Example of rules

Contradictions between the derived propositions in the recommendation process and what is demanded during the communication between the agents are viewed as signals that the set of assumptions should be modified. For example, if a proposition was derived with the information that the Louvre museum opens from Tuesday to Sunday but the agents are looking for a museum that opens on Monday, there is a contradiction and new assumptions that are compatible with the information needed should be generated using the ATMS. Assumptions are important because they affect the travel package recommendation and retracting these assumptions may change the recommendation. In the previous example, the agent has assumed that the user wants to stay in economic accommodation but if this hypothesis is not right and the user wants a first class hotel, there is a contradiction and the assumption should be retracted and a new hotel should be recommended.

Search Process

In a multiagent scenario like this it is necessary that agents coordinate their actions, especially when they need to communicate to each other and exchange information. The *Src* agents do not have a global view, goals and knowledge are local, making it difficult to cooperate. For this reason, we propose the combination of the TMS with the DCOP. This combination ensures that agents will coordinate their decision-making in this domain. A DCOP consists of *n* variables $V = x_1; x_2; ...; x_n$, each assigned to an agent, where the values of the variables are taken from a discrete domain $D_1; D_2; ...; D_n$, respectively. The agents choose values for

the variables trying to optimize a global objective function. Two agents are considered neighbors if they have a constraint between them. In our scenario, two agents only have a constraint between them if they are inside the community. The constraint between agents corresponds to the information the *Src* agent needs and the information it gets from other agent.

The special feature here is that the cost function defined works as a local similarity between the neighbor agents. To an agent, getting the lower cost means finding a neighbor with the information it needs. It means that the higher the difference between the agents, the higher the cost. The *Src* agents that are performing the tasks to a specific recommendation must find the most similar information in other agents. Each agent communicates to its neighbors to search for a perfect match or at least the most similar information. The cost function is the sum of the distance between the information each agent (a_u) is looking for, represented by i_{a_u} and the information the other agent (a_s) has, represented by (i_{a_s}) :

$$cost = \sum sim(i_{a_u}, i_{a_s})$$

where $sim(i_{a_u}, i_{a_s})$ represents the local similarity between the features. In numeric features, this value is given by the Euclidean distance between the values. In symbolic features the local similarity is $\{1 : i_{a_u} = i_{a_s} \; ; 0\}$. This procedure in the cost function ensures that the agent will always return some information to the *Asm* agent.

The Recommendation of a Travel Package

Let us consider that the user has chosen Paris as destination to his vacation and he would like to travel on 10th March. The first step in the recommendation process is the creation of the list of tasks. Agent *Asm* creates the list and let it available to the community. After this, each *Src* agent picks a task. The agent takes into account the index of confidence with the task to choose it. For example, $agent_2$ has performed more tasks about hotels so it prefers the task *hotel* because it has a bigger probability of having information about hotels in its knowledge base than information about flights or attractions.

Considering a scenario with three agents, $agent_1$ chose the flight task, $agent_2$ chose hotel task and $agent_3$ chose attraction task. The information about flights is necessary to define the attractions and the hotels. It means that $agent_2$ and $agent_3$ should wait for the information of $agent_1$. In the other hand, if the agents wait for this information to start their search processes, the performance of the final recommendation would be damage. Thus, $agent_2$ and $agent_3$ start their searches making some assumptions. $Agent_2$ assumed that the user is looking for three-star hotels and $agent_3$ assumed that the user would like to stay in Paris for a week (the duration of a standard travel package). After the search processes, the agents return to the *Asm* agent the information found.

$Agent_3$ found just one attraction about Paris in its knowledge base. However, it had assumed that the user will stay a week in the city, so it needs more attraction to fulfill the days. It has to communicate to the agents in the community to find out if some of them has the information about attractions in Paris.

Concluding Remarks

This paper presented a preliminary report that proposes and analyzes the utilization of distributed TMS approach applied in a recommender system in the tourism domain. This combination can yield good recommendations, considering this as a complex domain that needs specific knowledge distributed over different sources. The agents in this scenario are considered experts, i.e., travel agents that work together to compose a recommendation to the passenger. The confidence index was added in the agent's model to helps the agent to become an expert in a specific type of task. Another interested point is that the ideas presented here are being validated in a real scenario. A knowledge acquisition was done and the agents' knowledge bases were created with knowledge from a real travel agency.

References

de Kleer, J. 1986. An assumption-based tms, extending the atms, and problem solving with the atms. *Artificial Intelligence* 28(2):127–224.

Doyle, J. 1979. A truth maintenance system. *Artificial Intelligence* 12(3):231–272.

Huhns, M. N., and Bridgeland, D. M. 1991. Multiagent truth maintenance. *IEEE Transactions on Systems, Man, and Cybernetics* 21(6):1437–1445.

Lorenzi, F.; Santos, D. S.; and Bazzan, A. L. C. 2005. Negotiation for task allocation among agents in case-base recommender systems: a swarm-intelligence approach. In Aimeur, E., ed., *Proceedings of the Workshop Multi-Agent Information Retrieval and Recommender Systems - Nineteenth International Conference on Artificial Intelligence (IJCAI 2005)*, 23–27.

Macho, S.; Torrens, M.; and Faltings, B. 2000. A multi-agent recommender system for planning meetings. In *Workshop on Agent-based recommender systems (WARS'2000)*.

Modi, P. J.; Shen, W.-M.; Tambe, M.; and Yokoo, M. 2003. An asynchronous complete method for distributed constraint optimization. In *Second international joint conference on Autonomous agents and multiagent systems*, 161–168. New York, NY, USA: ACM Press.

Resnick, P.; Iacovou, N.; Suchak, M.; Bergstrom, P.; and Riedl, J. 1994. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings ACM Conference on Computer-Supported Cooperative Work*, 175–186.