MATOURA: MULTI-AGENT TOURIST ADVISOR

Constantin Halatsis[1], Panagiotis Stamatopoulos[1], Isambo Karali[1], Constantin Mourlas[1], Dimitris Gouscos[1], Dimitris Margaritis[1], Constantin Fouskakis[1], Angelos Kolokouris[2], Panagiotis Xinos[2], Mike Reeve[3], André Véron[3], Kees Schuerman[3], Liang-Liang Li[3]

[1]Univ. of Athens, Dept. of Informatics, Athens, Greece
[2]Expert Systems International, Athens, Greece
[3]ECRC, Munich, Germany

MaTourA is a tourist advisory system about Greece that is being implemented in the parallel constraint logic programming language ElipSys. The purpose of MaTourA is to facilitate the work carried out in travel agencies by providing an interactive way to construct personalized tours, select predefined package tours and handle the underlying touristic information. The system has been designed as a set of high-level interacting agents. In this direction, the ElipSys language was extended with the appropriate features to support the development of multi-agent systems.

Introduction

The introduction of the on-line computer services in the area of tourism has contributed significantly to improving services and sales. The need, however, remains largely unfulfilled, although immediately apparent, for an integrated and friendly access to the relevant data. Thus, tourism informatics as well as other applications of on-line systems need a presentation and usage environment that would make the computer immediately accessible. The necessary element that is missing so far and which is confining the effectiveness of the systems in use is above all the picture.

An effective advising process for the traveler requires the capture of the integrated information material in every possible form from all the sources and services. To be specific, the information from the hotel units must be connected to the relevant updates for cultural activities by the authorities on the exact areas that interest the visitor. Technology and the method utilized to communicate with and inform the tourist must be characterized by a global presentation that includes information in many forms, e.g. text, sound, photograph, and movie clips.

Following the above, a specific application on tourism, called MaTourA (**M**ulti-**a**gent **Tour**ist **A**dvisor) [7, 6] is presented in this paper. MaTourA is currently under development by the University of Athens and Expert Systems International in the context of the ESPRIT 6708 APPLAUSE Project (Application & Assessment of Parallel Programming Using Logic). MaTourA has been designed and is currently being implemented in the ElipSys language [2, 9, 4], a parallel constraint logic programming system developed at the European Computer-Industry Research Centre (ECRC).

APPLAUSE is a three year project that began in May 1992. The main aim of the project is to assess the advantages (and disadvantages) of the ElipSys language in various application areas, namely planning & scheduling, decision support and multi-agent systems. ElipSys is a pure parallel logic programming system which supports, apart from parallelism, other additional features, such as constraint satisfaction over finite domains, that may be exploited to design and implement applications useful in a real-world environment.

The multi-agent systems approach adopted in the case of MaTourA is a major research area of Distributed Artificial Intelligence (DAI) [3], in which agents of various types and capabilities cooperate in problem solving. MaTourA comprises a set of autonomous agents reflecting the procedures involved in a tourist advisory environment. In order to support the development of MaTourA, ElipSys was extended appropriately with a three-layered socket based communication scheme capable to satisfy the interaction requirements of the MaTourA agents.

MaTourA is the evolution of another application, the single-agent PErsonalized Tourist INformation Advisor (PETINA) [12, 13] developed by the University of Athens in the ElipSys language. PETINA is a prototype with rather shallow knowledge and simplistic form of reasoning. Thus, it was straightforward to implement it as a conventional ElipSys application. However, this is not the case with MaTourA, so a multi-agent architecture seems appropriate and *sine qua non* for the application.

In the following, short overviews of the multi-agent systems technology and the ElipSys language are presented, as well as a brief description of the communication oriented extension of ElipSys. Then, the MaTourA functional and structural specifications are discussed and the agents of the system are described. Finally, the implementation issues concerning the first prototype of the system are presented.

Multi-agent Systems

As it is stated in [3], in the multi-agent systems area of DAI, "*research is concerned with coordinating intelligent behaviour among a collection of (possibly pre-existing) autonomous intelligent agents, how they can coordinate their knowledge, goals, skills and plans jointly to take action or to solve problems*".

The approach that has to be followed for the development of a multi-agent system is to build a separate subsystem for each problem domain and, then, make these susbsystems cooperate. There are many advantages in a procedure like that. Firstly, the modularity achieved reduces the complexity of the whole system, which, in addition, is more reliable, in the sense that it can continue to operate even if part of it fails. Moreover, there might be a speedup in execution, since the subsystems can operate in parallel. Another advantage is that the subsystems are reusable. Finally, the knowledge acquisition procedure is facilitated, since it is easier to find experts in narrower domains.

The cooperation among the agents of a multi-agent system may be achieved in various ways, according to the model employed for this reason. The research in this direction has produced models for cooperation, such as the blackboard [8, 5], the contract net [11] and the actor [1] models. In any case, there are some problems that have to be addressed, such as description, decomposition and allocation of problems among a group of intelligent agents, synthesis of results as well as representation and reasoning by an agent about the actions, plans and knowledge of other agents.

The ElipSys Language

ElipSys is a parallel logic programming language which incorporates various powerful execution mechanisms. The language supports OR-parallelism, AND-parallelism, data-parallelism, a data driven computation rule and constraint satisfaction techniques over finite domains. All these features make the language ideal for stating and describing search-based applications and combinatorial problems.

OR-parallelism aims at the concurrent exploration of the various alternative clauses that define an ElipSys procedure. AND-parallelism results from the concurrent execution of two goals in conjunction. This feature is not provided by the ElipSys execution model, neither it is supported by the run time environment of the language. It resides only at the language level and is actually compiled into OR-parallelism. Data-parallelism is a kind of parallelism arising from the concurrent treatment of the elements of a set of data.

In addition, ElipSys provides a data driven computation rule on top of the usual depth-first left-to-right execution strategy of Prolog. This rule modifies the reduction order of goals according to instantiations of variables by declaring that every goal that refers to a specific predicate has to be delayed, if its arguments are not adequately instantiated. A delayed goal is awakened when the desirable degree of instantiation is achieved. Finally, ElipSys supports constraint satisfaction techniques over finite domains, which lead to a priori pruning of the search space and thus, they result in more optimized execution.

ElipSys has been designed for maximum portability between different parallel platforms. A hybrid scheme is used, based on shared memory and message passing. It is currently available on various machines, namely Sun workstations (in sequential and pseudo-parallel mode), Sequent Symmetry, Sparc based multiprocessors from Sun and ICL and the KSR-1 distributed memory machine. On the latter, the hardware supported shared virtual memory is used.

A Communication Scheme in ElipSys

The features of the ElipSys language that were described earlier aim to support the efficient solution of problems that may be mapped to a search of OR-trees. The parallelism offered may serve as a means for the concurrent exploration of branches of these trees and the constraint satisfaction mechanism, as well as the data driven computation to a lesser extent, may be used to prune the OR-trees and reduce the search space. Unfortunately, this computational environment is not sufficient to support the development of multi-agent systems like MaTourA. The reason is that for such kind of systems, a framework is needed to allow the agents to communicate and exchange information. In any case, the searching facilities provided by ElipSys are indispensable at the intra-agent level, especially if someone wants to have really intelligent agents.

For this reason, an appropriate extension of ElipSys was needed that could be integrated smoothly with the core system and that would be able to support the interaction requirements of MaTourA. A framework such as the one provided by SICStus Prolog [10] would be profitable. What was actually done for the MaTourA case was to enhance ElipSys with SICStus-like communication facilities and to extend them so as to support the transparent interaction among agents. The whole framework may be viewed as a three-layered communication scheme. These layers consist of the following:

Low-level built-ins: These correspond very closely to the actual UNIX system calls used by an ordinary process interface to network entities. They implement all the required primitives to support the communication among processes through stream sockets in the Internet domain. The low-level built-ins were implemented in C and connected to the core language through the ElipSys to C interface.

Medium-level predicates: These can be used to pass arbitrary terms between a blackboard server and various clients. These predicates are functionally similar to those used in SICStus Prolog for communication with a Linda server. This set of predicates was implemented in ElipSys on top of the previous low-level built-ins and supports the upper layer of the communication scheme.

High-level predicates: These predicates were tailored to be used for a virtual point-to-point message passing (through the blackboard) by multi-agent applications. They are the ones that support the development of MaTourA as well as other applications of similar nature. ElipSys processes are able to exchange messages through predicates, such as *send_request/3*, *get_message/3*, *send_answer/3* etc. which were also implemented in ElipSys.

The Functionality of MaTourA

MaTourA is a tourist advisory system whose purpose is to help travel agents satisfy the needs of their customers/tourists. The system has been designed using a multi-agent architecture in a way compatible with DAI principles. More precisely, the different procedures carried out in a travel agency correspond to MaTourA agents. Some of these agents are responsible to handle primitive touristic information, i.e. information related to accommodation, transportation etc. The agents of MaTourA cooperate in order to solve certain problems given to the system. Each agent contributes to the solution by using its knowledge to solve subproblems of the global problem.

The functionality of MaTourA covers the different topics that characterize the work in a travel agency. More precisely, the system handles various types of requests to construct personalized tours that satisfy the tourists' wishes. Certain kinds of requests refer to the selection of precompiled package tours offered by travel agencies. Information retrieval and maintenance is also supported by the system and, finally, training is supported, so as to facilitate the use of the system by travel agents.

The Structure of MaTourA

MaTourA comprises a set of agents each reflecting procedures involved in a tourist advisory environment. Each agent accomplishes a specific functionality and manipulates specific knowledge. These agents may either answer queries on their own or coordinate their knowledge and communicate with each other in case joint action is required.

MaTourA consists of the following agents: Tour Generation Agent, Activity Agent, Event Agent, Site Agent, Accommodation Agent, Transportation Agent, Ticketing Agent, Package Tour Agent, User Interface Agent, Info Retrieval/Maintenance Agent and Training Agent.

Tour Generation Agent: The Tour Generation Agent is responsible for the generation of personalized tours, taking into account user wishes. These tours are time/location schedules providing, for each day, a desirable timetable. The term "locations" refers to the elementary sites maintained by the Site Agent. The Tour Generation Agent constructs the requested personalized tours. In order to carry out this task, it needs information supplied by many agents of the MaTourA system. More precisely, the Activity, Event, Site, Accommodation and Transportation Agents are queried by the Tour Generation Agent. Requests of four types are accepted. These are the following in increasing degree of complexity: i) A full time/location schedule is provided to the agent, e.g. 15 Jul 93 — 19 Jul 93 in Athens, 19 Jul 93 — 25 Jul 93 in Rhodes, 25 Jul 93 — 31 Jul 93 in Heraklion, ii) The visit period of the tour is given, but not the very locations. However, a wider area/division is supplied or, alternatively, a starting location for the tour, e.g. 15 Jul 93 — 5 Aug 93 in Cyclades, iii) Specific locations are given but no information about time, e.g. a tour in Thessaloniki, Edessa, Kavala, Alexandroupoli and iv) Neither specific locations nor information about time is provided, but a tour with a given starting location or in a wider area/division is requested, e.g. a tour starting from Ioannina.

For each type of request, cost criteria as well as interest preferences are given to the agent. Moreover, acceptable transportation means and accommodation constraints are taken into consideration, if requested. Finally, a set of Daily Plan Templates (DPTs) are supplied to the agent. Each DPT corresponds to a typical day that the tourist would like to spend on a specific location. A DPT consists of time/action pairs, e.g. 10:00 — 14:00 swimming, 18:00 — 21:00 sightseeing.

Activity Agent: The Activity Agent holds information about activities and is able to answer related requests coming from other MaTourA agents. Activities are possible tourist's visits to various spots, such as visits to museums, galleries, archaeological sites etc., or even more, beaches for swimming, surfing etc. In general, activities are related with spots that might be of interest to a tourist.

Event Agent: The Event Agent holds information about events and, as is the case with the Activity Agent, it answers requests relevant to this information which come from other MaTourA agents. Events differ from activities in that they are short term shows with rather temporary nature, such as exhibitions, music concerts etc., while the activities are more permanent in time.

Site Agent: This agent deals with the sites of Greece. A site is a geographical entity of the country. The concept of sites is an important one for MaTourA as the whole tour generation facility of the system works considering the sites as a fundamental starting point. In addition, plain site information is vital for any system that deals with tourism. The concept of sites is related with a site inclusion relation. This inclusion considers three levels of sites: i) *elementary sites* which are the cities, towns and villages, ii) *areas* which are the islands and the provinces of Greece and iii) *divisions* which are large islands, island groups, regions and land areas.

Accommodation Agent: This agent handles information about lodgings where a tourist may be accommodated and returns it after the appropriate requests are made. In a real-world tourist advisory system, this agent has to be connected with external reservation systems.

Transportation Agent: The Transportation Agent holds information about connections between elementary sites through different transportation means. Thus, various transportation graphs, each one for a specific transportation means, are coded for all the elementary sites in Greece. As transportation means, *private car*, *bus*, *train*, *boat* and *airplane* are considered, resulting to five different transportation graphs. The Transportation Agent supports plain information retrieval and it is also capable to answer more complicated requests, such as ones that refer to routing problems. Whenever it has to solve a problem that involves ticketing issues, the Ticketing Agent is requested.

Ticketing Agent: The Ticketing Agent has information about the connections that are established between different elementary sites all over Greece by public transportation means. Various information retrieval requests are supported, however the most complicated task that the agent carries out refers to the computation of time schedules for given routes through elementary sites. As is the case with the Accommodation Agent, in a real-world system, the Ticketing Agent has to be connected to external reservation systems.

Package Tour Agent: This agent is responsible for producing package tours that match requirements described in requests which are sent to it. A package tour is a precompiled tour as it has been constructed by a travel agency.

User Interface Agent: This agent is responsible for controlling all user interactions with the MaTourA system. The main theme of the User Interface Agent is to observe, analyze, constrain and guide the user interaction with MaTourA. The model of interaction makes use of advanced dialogue techniques to implement a user friedly and responsive system. During the interaction, the User Interface Agent attempts to extract implicitly as much information of the user as it can capture, before this becomes annoying to the user and a considerable load to the dialogue.

Info Retrieval/Maintenance Agent: This agent satisfies the MaTourA requirement to deal with large in volume, persistent data. The functionality supported by this agent deviates from the common functionality supported by the contemporary commercial RDBMSs in the sense that MaTourA needs to store non-conventional data, such as complex data structures and, to a lesser extent, deductive rules.

Training Agent: The Training Agent stands by the User Interface Agent to assist the user in the use of the tourist advisor. This agent may either be invoked by the user or automatically intervene during a session when the User Interface Agent decides. The users of MaTourA are modelled according to user stereotypes (naive, intermediate and experts). When presenting information to the user, the intelligent help facility formulates the contents, the context and the style of the presented help information to the appropriate level of the type of the individual user.

The First Prototype

A first prototype of the MaTourA system has been already implemented in ElipSys version 0.6 and runs on Sparc-based Sun workstations under SunOS 4.1.2. This prototype covers most of the MaTourA agents, some of them with their full functionality (AcA, EvA, AmA, TkA) and others partially (TGA, StA, TpA). The purpose of the development of the prototype was twofold. Fistly, it aimed at the verification of the correctness of the three-layered communication scheme that was developed as an extension of the ElipSys language and, secondly, it served as an initial demonstrator of the functionality of MaTourA.

Conclusions

MaTourA is a Multi-agent Tourist Advisor, an application which is being implemented in the ElipSys language in the context of the ESPRIT 6708 APPLAUSE Project. ElipSys is a parallel constraint logic programming system that was extended with a socket based communication framework, in order to be able to support the design and development of multi-agent systems, such as MaTourA.

MaTourA major capabilities are to construct tours that satisfy user wishes and to provide information about a variety of entities that present touristic interest. User wishes involve various information that logically and physically refers to different subsystems which, actually, are the system agents. In addition, maintenance of the stored information as well as user training are provided by the system.

MaTourA functionality needs clearly a multi-agent system platform both as a programming tool and as a computational model. The system consists of a set of agents each accomplishing specific functionality and manipulating specified knowledge. These agents may either answer the query that was put to them on their own, or communicate with each other in case supplementary information is needed by an agent.

A prototype version of MaTourA has been implemented in the extended ElipSys language. This version is currently being updated, so as to achieve the full functionality of MaTourA.

Acknowledgments

The APPLAUSE Project is partially funded by the ESPRIT Programme of the Commission of the European Communities as ESPRIT Project 6708.

References

[1] G. Agha and C. Hewitt. Concurrent programming using actors: Exploiting large-scale parallelism. AI Memo 865, Massachusetts Institute of Technology, 1985.

[2] U. Baron, S. Bescos, S. Delgado-Rannauro, P. Heuzé, M. Dorochevsky, M. Ibáñez-Espiga, K. Schuerman, M. Ratcliffe, A. Véron, and J. Xu. The ElipSys logic programming language. Technical Report DPS-81, ECRC, December 1990.

[3] A. Bond and L. Gasser, editors. *Readings in Distributed Artificial Intelligence.* Morgan Kaufmann Publishers, Inc., San Mateo, California, 1988.

[4] *ElipSys User Manual for Release Version* 0.6, April 1993.

[5] R. Engelmore and T. Morgan, editors. *Blackboard Systems.* Addisson-Wesley Publishing Company, Wokingham, England, 1988.

[6] C. Halatsis, P. Stamatopoulos, D. Margaritis, I. Karali, C. Mourlas, D. Gouscos, and C. Fouskakis. Tool assessment. APPLAUSE Deliverable D.WP3.4, University of Athens, May 1993.

[7] C. Halatsis, P. Stamatopoulos, Z. Palaskas, I. Karali, C. Mourlas, D. Gouscos, D. Margaritis, and C. Fouskakis. MaTourA specification. APPLAUSE Deliverable D.WP3.2, University of Athens — Expert Systems International, May 1993.

[8] B. Hayes-Roth. A blackboard architecture for control. *Artificial Intelligence*, 26:251–321, 1985.

[9] S. Prestwich. ElipSys programming tutorial. APPLAUSE Deliverable D.WP4.ECRC.4A, ECRC, November 1992.

[10] *SICStus Prolog User's Manual*, August 1992.

[11] R. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12):1104–1113, December 1980.

[12] P. Stamatopoulos, I. Karali, and C. Halatsis. PETINA — Tour generation using the ElipSys inference system. In *Proceedings of the 1992 ACM/SIGAPP Symposium on Applied Computing*, volume 1, pages 320–327, 1992.

[13] P. Stamatopoulos, I. Karali, and C. Halatsis. A tour advisory system using a logic programming approach. *Applied Computing Review*, 1(1):18–25, 1993.