Using a Distributed Multi-Agent Architecture for Optimizing the Performance of a Case-Based Planning Mechanism

Sara Rodríguez, Juan F. De Paz, Dante I. Tapia, Juan M. Corchado

Departamento de Informática y Automática, Universidad de Salamanca
Plaza de la Merced s/n, 37008, Salamanca, España
{srg, fcofds, dantetapia, corchado}@usal.es

Abstract. This paper presents how a distributed multi-agent architecture facilitates the integration of services and applications to optimize the performance of a Case-Based Planning mechanism. This mechanism has been modelled as a service and is part of a multi-agent system aimed at enhancing health care for Alzheimer patients in geriatric residences. Several tests have been done to demonstrate that this approach is adequate to build complex systems with distributed functionalities.

Keywords: Multi-agent Architecture, Services Oriented Architectures, Case-Based Reasoning, Case-Based Planning.

1. Introduction

Agents and multi-agent systems have been successfully applied to several scenarios, such as education, culture, entertainment, medicine, robotics, etc. [5, 16]. The characteristics of the agents make them appropriate for developing dynamic and distributed systems, as they have the capability of adapting themselves to the users and environmental characteristics [10].

This paper describes how FUSION@[1] (*Flexible User and ServIces Oriented multi-ageNt Architecture*) can optimize the performance of a Case-Based Planning mechanism. The architecture proposes a new and easier method of building distributed multi-agent systems, where the functionalities of the systems are not integrated into the structure of the agents; rather they are modelled as distributed services and applications which are invoked by the agents acting as controllers and coordinators. The agents in this architecture are based on the deliberative Belief, Desire, Intention (BDI) model [11, 4], where the agents' internal structure and capabilities are based on mental aptitudes, using beliefs, desires and intentions [3, 8]. FUSION@ have modelled the agents' characteristics [11] to provide them with

[1] FUSION@ is an experimental multi-agent architecture developed by the BISITE Research Group at the University of Salamanca, Spain. For more information please visit http://bisite.usal.es

mechanisms that allow solving complex problems and autonomous learning, including Case-Based Reasoning and Case-Based Planning mechanisms.

FUSION@ has been applied to develop a multi-agent system aimed at enhancing health care for Alzheimer patients in geriatric residences. One of the most important characteristics of this system is the use of Case-Based Reasoning (CBR) [1] and Case-Based Planning (CBP) mechanisms, where problems are solved by using solutions to similar past problems [5]. Solutions are stored into a cases memory which the mechanisms can consult in order to find better solutions for new problems. CBR and CBP mechanisms have been modelled as external services. Deliberative BDI agents use these services to learn from past experiences and create optimal solutions.

Next, the main characteristics of FUSION@ are briefly explained. Then, the whole system structure is presented, including the Case-Based Planning mechanism used to dynamically assign tasks to the medical staff. Finally, there are presented some results obtained after implementing this approach into a real scenario.

2. The Architecture Model

The continuous advances in technology and software requires creating increasingly complex and flexible applications, so there is a trend toward reusing resources and share compatible platforms or architectures. In some cases, applications require similar functionalities already implemented into other systems which are not always compatible. At this point, developers can face this problem through two options: reuse functionalities already implemented into other systems; or re-deploy the capabilities required, which means more time for development, although this is the easiest and safest option in most cases. While the first option is more adequate in the long run, the second one is most chosen by developers, which leads to have replicated functionalities as well as greater difficulty in migrating systems and applications. Moreover, the absence of a strategy for integrating applications generates multiple points of failure that can affect the systems' performance. This is a poorly scalable and flexible model with reduced response to change, in which applications are designed from the outset as independent software islands.

Excessive centralization of services negatively affects the systems' functionalities, overcharging or limiting their capabilities. Classical functional architectures are characterized by trying to find modularity and a structure oriented to the system itself. Modern functional architectures like Service-Oriented Architecture (SOA) consider integration and performance aspects that must be taken into account when functionalities are created outside the system. These architectures are aimed at the interoperability between different systems, distribution of resources, and the lack of dependency of programming languages [6]. Services are linked by means of standard communication protocols that must be used by applications in order to share resources in the services network [2]. The compatibility and management of messages that the services generate to provide their functionalities is an important and complex element in any of these approaches.

One of the most prevalent alternatives to these architectures is agents and multi-agent systems which can help to distribute resources and reduce the central unit tasks

[2, 18]. A distributed agents-based architecture provides more flexible ways to move functions to where actions are needed, thus obtaining better responses at execution time, autonomy, services continuity, and superior levels of flexibility and scalability than centralized architectures [5]. Additionally, the programming effort is reduced because it is only necessary to specify global objectives so that agents cooperate in solving problems and reaching specific goals, thus giving the systems the ability to generate knowledge and experience.

The integration and interoperability of agents and multi-agent systems with SOA and Web Services approaches has been recently explored [2]. However, the systems developed using these frameworks are not open at all because the framework is closed and services and applications must be programmed using a specific programming language that support their respective proprietary APIs. FUSION@ has an advantage regarding development because we have already implemented it into a real scenario. It not only provides communication and integration between distributed agents, services and applications; it also proposes a new method to facilitate the development of distributed multi-agent systems by means of modelling the functionalities of the agents and the systems as services.

The model of FUSION@ has been mainly designed to develop dynamic and distributed multi-agent systems. Thanks to the agents' capabilities, the systems developed can make use of reasoning mechanisms or learning techniques to handle services and applications. Agents, applications and services can communicate in a distributed way, even from mobile devices. As can be seen on Figure 1, FUSION@ framework defines four basic blocks:

1. *Applications*. Represent all the programs that can be used to exploit the system functionalities. Applications are very important for achieving high levels of human-computer interaction. The use of distributed applications expands the possibilities to create multi-modal interfaces which adapt to the users' characteristics and needs.
2. *Services*. Represent the activities (e.g. mechanisms, routines, algorithms, etc.) that the architecture offers.
3. *Agents Platform*. This is the core of FUSION@, integrating a set of agents, each one with special characteristics and behaviour.
4. *Communication Protocol*. It allows applications and services to communicate directly with the *Agents Platform* and is completely open and independent of any programming language.

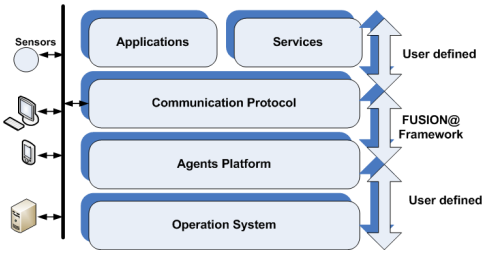

Fig. 1. FUSION@ framework

Users can access the system through distributed applications which run on different types of devices and interfaces (e.g. computers, cell phones, PDA). The agents analyze all requests and invoke the specified services either locally or remotely. Services process the requests and execute the specified tasks. Then, services send back a response with the result of the specific task.

FUSION@ is a modular multi-agent architecture, where services and applications are managed and controlled by deliberative BDI agents [11, 4]. Deliberative BDI agents are able to cooperate, propose solutions on very dynamic environments, and face real problems, even when they have a limited description of the problem and few resources available. These agents depend on beliefs, desires, intentions and plan representations to solve problems [3, 8]. Deliberative BDI agents are the core of FUSION@. There are different kinds of agents in the architecture, each one with specific roles, capabilities and characteristics. This fact facilitates the flexibility of the architecture in incorporating new agents. However, there are pre-defined agents which provide the basic functionalities of the architecture:

- The *CommApp Agent* is responsible for all communications between applications and the platform. It manages the incoming requests from the applications to be processed by services
- The *CommServ Agent* is responsible for all communications between services and the platform. The functionalities are similar to CommApp Agent but backwards.
- The *Directory Agent* manages the list of services that can be used by the system. For security reasons [17], the list of services is static and can only be modified manually; however, services can be added, erased or modified dynamically.
- The *Supervisor Agent* supervises the correct functioning of the other agents in the system.
- The *Security Agent* analyzes the structure and syntax of all incoming and outgoing messages.
- The *Admin Agent* decides which agent must be called by taking into account the users preferences.
- The *Interface Agents* are designed to be embedded in users' applications. Interface agents communicate directly with the agents in FUSION@ so there is no need to employ the communication protocol, rather the FIPA ACL specification.

FUSION@ is an open architecture that allows developers to modify the structure of these agents, so that agents are not defined in a static manner. Developers can add new agent types or extend the existing ones to conform to their projects needs.

FUSION@ has been employed to develop an improved version of ALZ-MAS (ALZheimer Multi-Agent System) [5], a multi-agent system aimed at enhancing the assistance and health care for Alzheimer patients living in geriatric residences. The main functionalities in the system include Case-Based Reasoning and Case-Based-Planning mechanisms that are embedded into deliberative BDI agents. These mechanisms allow the agents to make use of past experiences to create better plans and achieve their goals. Because CBR and CBP mechanisms are the core of the system, they must be available at all times. The system depends on these mechanisms to generate all decisions, so it is essential that they have all processing power available in order to increase overall performance.

To demonstrate the capabilities of FUSION@ in integrating complex services, the most complex and resources demanding task, in this case the CBP mechanism, has

been modelled as a service, so any agent can make use of them. The next section describes the CBP mechanism that has been extracted from the agents' structure and modelled as a service.

3. Case-Based Planning Service

Case-based Reasoning (CBR) is a type of reasoning based on past experiences [14]. The primary concept when working with CBR systems is the concept of case, which is described as a past experience composed of three elements:

1. An initial state or problem description that is represented as a belief.
2. A solution, which provides the sequence of actions carried out in order to solve the problem.
3. A final state, which is represented as a set of goals.

CBP comes from CBR, but is specially designed to generate plans (sequence of actions) [5]. In CBP, the proposed solution for solving a given problem is a plan. This solution is generated by taking into account the plans applied for solving similar problems in the past. The problems and their corresponding plans are stored in a plans memory. The reasoning mechanism generates plans using past experiences and planning strategies, which is how the concept of Case-Based Planning is obtained [9].

CBP consists of four sequential stages:

1. The retrieve stage, which recovers the past experiences most similar to the current one.
2. The reuse stage, which combines the retrieved solutions in order to obtain a new optimal solution.
3. The revise stage, which evaluates the obtained solution.
4. The retain stage, which learns from the new experience.

The problem description (initial state) and the solution (situation when final state is achieved) are represented as beliefs, the final state as a goal (or set of goals), and the sequences of actions as plans. The CBP cycle is implemented through goals and plans. When the goal corresponding to one of the stages is triggered, different plans (algorithms) can be executed concurrently to achieve the goal or objective. Each plan can trigger new sub-goals and, consequently, cause the execution of new plans. In practice, what is stored is not only a specific problem with a specific solution, but also additional information about how the plans have been derived. As with case-based reasoning, the case representation, the plans memory organization, and the algorithms used in every stage of the case-based planning cycle are essential in defining an efficient planner.

FUSION@ can take advantage of these characteristics in order to enhance the services organization. Following FUSION@ model, these mechanisms have been modelled as services linked to agents, thus increasing ALZ-MAS overall performance. Figure 2 shows the mains steps to generate a new plan. First, an application (e.g. *Interface Agent*) sends a SOAP message to the platform. The message is processed by the *Admin Agent* which decides to use the planner service. The platform invokes the service sending a SOAP message. The location of each nurse and patient in the residence is also sent to the planner service. The service

receives the message and starts to generate a new plan by means of the CBP cycle. Then, the solution is sent to the platform which delivers the new plan to all nurses connected to the system.

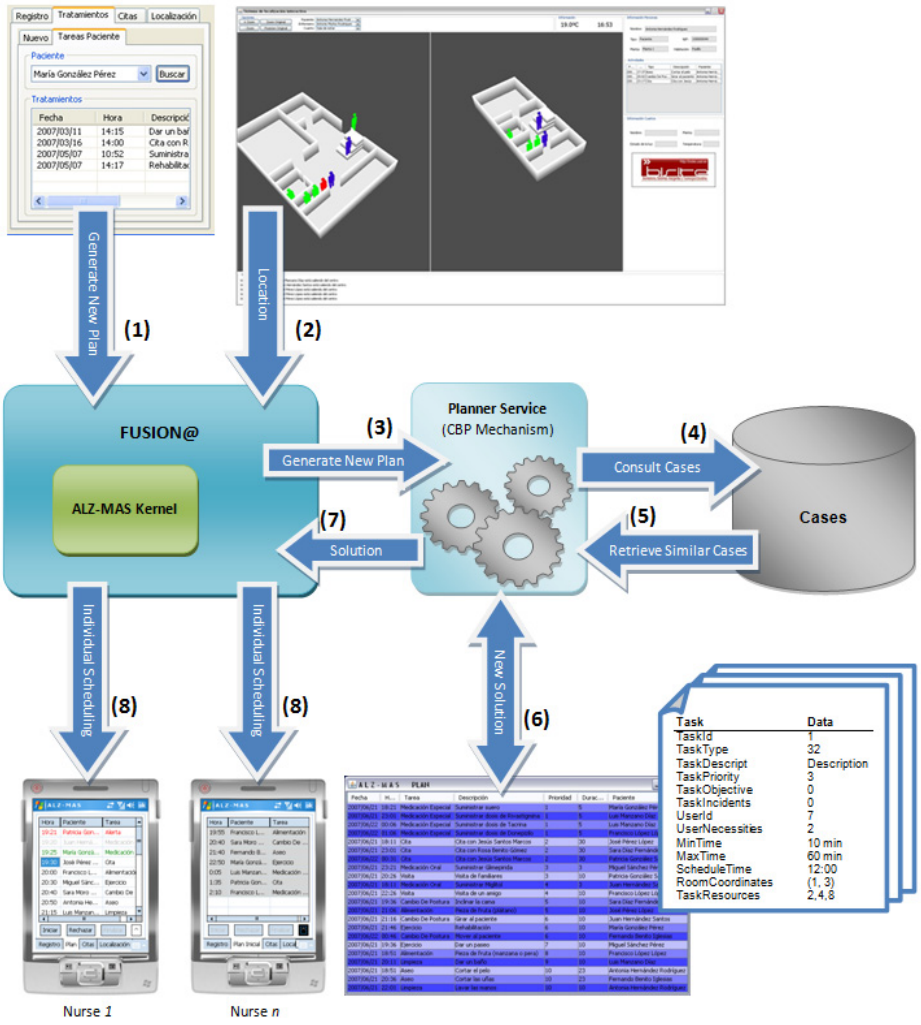

Fig. 2. Steps to generate a new plan

The CBP service creates optimal paths and scheduling in order to facilitate the completion of all tasks defined for the nurses connected to the system. A task is a java object that contains a set of parameters: *TaskId, TaskType, TaskDescript, TaskPrority, TaskObjective, TaskIncidents, UserId, UserNecessities, MinTime ,MaxTime, ScheduleTime , RoomCoordinatesand TaskResources*. In this case, a plan is the name given to a sequence of tasks that defines the best path for each nurse. A problem description (belief) will be formed by the tasks that the nurses need to execute, the

location, the resources available, and the times assigned for their shift. In the retrieve stage, those problem descriptions found within a range of similarity close to the original problem description are recovered from the beliefs base. In our case, a tolerance of 20% has been permitted. In order to do this, the CBP service allows the application of different similarity algorithms (cosine, clustering etc.). Once the most similar problem descriptions have been selected, the solutions associated with them are recovered. One solution contains all the plans (sequences of tasks) carried out in order to achieve the objectives for a problem description (assuming that replanning is possible) in the past, as well as the efficiency of the solution being supplied. The chosen solutions are combined in the reuse stage to construct a plan [9]. The reuse is focused on the objectives and resources needed by each task, as well as on the objectives that the nurses need to perform and the resources available in order to carry out the global plan (compilation of all tasks for all nurses). The objectives that each nurse has are aimed to attend the patients and not exceed eight working hours. The time available is a problem restriction. The revision of the plans is made by the nurses. If the evaluation of the plan is at least a 90% similar, the case is stored in the cases memory.

The variation of the plans will essentially be induced by: the changes that occur in the environment and that force the initial plan to be modified; and the knowledge from the success and failure of the plans that were used in the past, and which are favoured or punished via learning.

If there are no similar cases, the entire planning process must be performed. As the objective of this paper is not describing in detail the CBP mechanism but the distributed approach used to model it as a service, this mechanism is briefly introduced. The planning is carried out through a neural network based on the Kohonen network [12]. The basic Kohonen network [13] cannot be used to resolve our problem since it attempts to minimise distances without taking into account any other type of restriction, such as time limits. In our case, the planner is based on Kohonen networks but with a number of improvements that allow us to reach a solution far more rapidly [15]. Furthermore, once a solution has been reached, it is re-modified in order to take restrictions into account. As such, by modifying the basic algorithm, we are aiming to make the solution search more flexible. In order to achieve this, the basic vicinity function used in the Kohonen network is modified and the number of neurons in the output layer corresponds to the places that the subject wishes to visit.

Next, the results obtained after applying a distributed approach in ALZ-MAS by means of FUSION@ are presented.

4. Results and Conclusions

FUSION@ facilitates the distribution of functionalities (services and applications) and proposes an alternative where agents are based on the BDI model and act just as controllers and coordinators. This approach releases the agents from high demanding computing tasks and enhances their abilities to recover from errors.

FUSION@ has been used to improve ALZ-MAS, a multi-agent system prototype implemented in a geriatric residence [5]. Several tests have been done to determine the performance of the system when executing a specific activity. The tests consisted of a set of requests delivered to the planning mechanism (CBP) described in section 3, which in turn had to generate paths for each set of tasks. For every new test, the cases memory of the CBP mechanism was deleted in order to avoid a learning capability, thus requiring the mechanism to accomplish the entire planning process.

ScheduleTime is the time in which a specific task must be accomplished, although the priority level of other tasks needing to be accomplished at the same time is factored in. The CBR mechanism increases or decreases *ScheduleTime* and *MaxTime* according to the priority of the task:

```
ScheduleTime = ScheduleTime-5min*TaskPriority

MaxTime = MaxTime+5min*TaskPriority
```

Once these times have been calculated, the path is generated taking the *RoomCoordinates* into account. There were 30 defined agendas each with 50 tasks. Tasks had different priorities and orders on each agenda. Tests were carried out on 7 different test groups, with 1, 5, 10, 15, 20, 25 and 30 simultaneous agendas to be processed by the CBP mechanism. 50 runs for each test group were performed, all of them on machines with equal characteristics. Several data have been obtained from these tests, notably the average time to accomplish the plans, the number of crashed agents, and the number of crashed services.

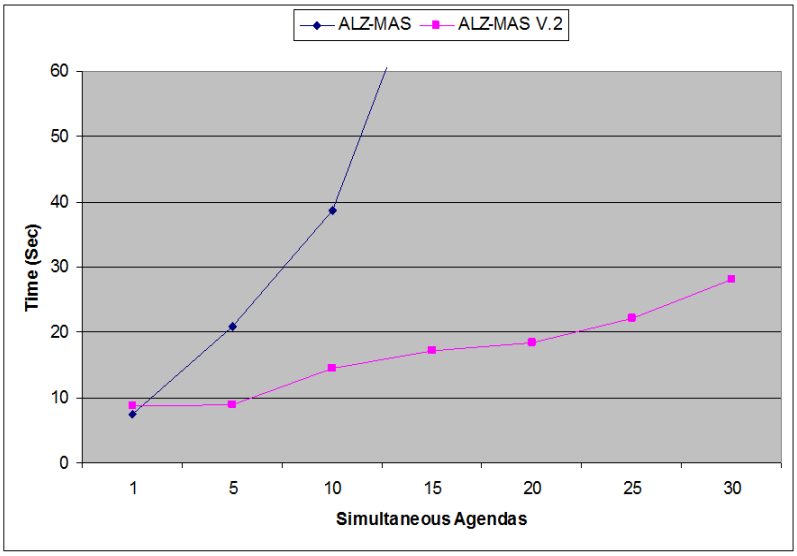

Fig. 3. Time needed for both systems to generate paths for a group of simultaneous agendas

As can be seen on Figure 3, the previous version of the system (centralized architecture) was unable to handle 15 simultaneous agendas and time increases to

infinite because it was impossible to perform those requests. However, the new system (distributed architecture) had 5 replicated services available, so the workflow was distributed and allowed the system to complete the plans for even 30 simultaneous agendas. None of the tests where agents or services crashed were taken into account to calculate this specific data, so these tests were repeated.

Figure 4 shows that the previous version of ALZ-MAS is far more unstable than the new one, especially when comparing the number of crashed agents. These data reveal that this approach provides a higher ability to recover from errors.

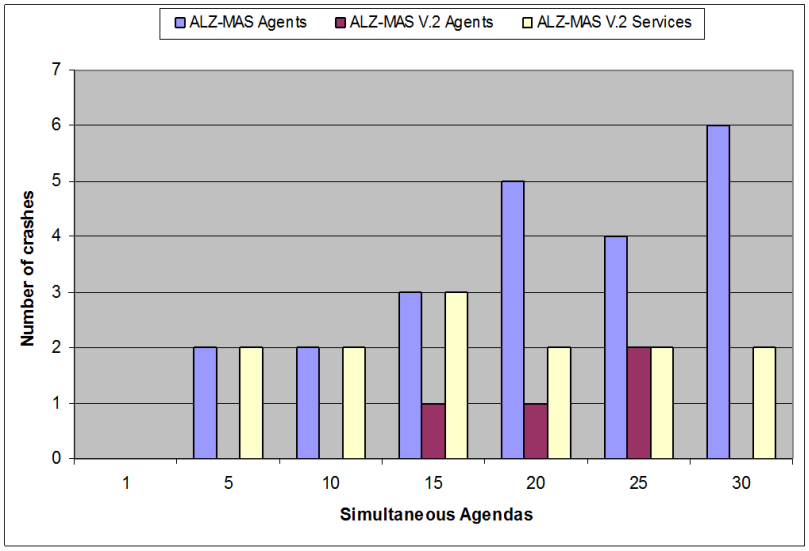

Fig. 4. Number of agents and services crashed at both versions of the system

Although FUSION@ is still under development, preliminary results demonstrate that it is adequate for building complex systems and exploiting composite services, in this case the CBP mechanism presented. This mechanism is just an example that shows how to deploy distributed services using FUSION@. However, services can be any functionality (mechanisms, algorithms, routines, etc.) designed and deployed by developers.

It is necessary to continue developing and enhancing the architecture presented. We are currently working on adding security mechanisms to analyze messages and embedded queries (e.g. SQL queries). The release of FUSIO@ for public use has been scheduled for Q4 2008.

Acknowledgements. This work has been partially supported by the TIN2006-14630-C03-03 and the IMSERSO 137/07 projects.

References

1. Aamodt, A., Plaza, E.: Case-Based Reasoning: foundational Issues, Methodological Variations, and System Approaches. AI Communications. Vol. 7. pp. 39--59 (1994)
2. Ardissono, L., Petrone, G. and Segnan, M.: A conversational approach to the interaction with Web Services. Computational Intelligence, Blackwell Publishing. Vol. 20. pp. 693—709 (2004)
3. Bratman, M.E.: Intentions, plans and practical reason. Harvard University Press, Cambridge, MA, USA (1987)
4. Bratman, M.E., Israel, D. and Pollack, M.E.: Plans and resource-bounded practical reasoning. Computational Intelligence, Blackwell Publishing. Vol. 4. pp. 349—355 (1988)
5. Camarinha-Matos, L.M., Afsarmanesh, H.: A Comprehensive Modeling Framework for Collaborative Networked Organizations. Journal of Intelligent Manufacturing, Springer Netherlands. Vol. 18(5). pp. 529-542 (2007)
6. Cerami, E.: Web Services Essentials Distributed Applications with XML-RPC, SOAP, UDDI & WSDL. O'Reilly & Associates, Inc. 1st Edition (2002)
7. Corchado, J.M., Bajo, J., De Paz, Y., Tapia, D.I.: Intelligent Environment for Monitoring Alzheimer Patients, Agent Technology for Health Care. Decision Support Systems, Eslevier, Amsterdam, Netherlands. In Press. ISSN 0167-9236 (2008)
8. Georgeff, M.,Rao, A.: Rational software agents: from theory to practice. Agent Technology: Foundations, Applications, and Markets, N.R. Jennings and M.J. Wooldridge (Eds), Springer-Verlag, New York, USA (1998)
9. Glez-Bedia, M.;Corchado, J.M.:A planning Strategy based on Variational Calculus for Deliberative Agents.Computing and Information Systems Journal.Vol. 10.pp.2—14 (2002)
10. Jayaputera, G.T., Zaslavsky, A.B., Loke, S.W.: Enabling run-time composition and support for heterogeneous pervasive multi-agent systems. Journal of Systems and Software. Vol. 80(12). pp. 2039—2062 (2007)
11. Jennings, N.R., Wooldridge M.: Applying agent technology. Applied Artificial Intelligence, Taylor & Francis. Vol. 9(4). pp. 351--361 (1995)
12. Jin, H.D., Leung, K.S., Wong, M.L., Xu, Z.B.: An Efficient Self-Organizing Map Designed by Genetic Algorithms for the Traveling Salesman Problem. IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics. Vol. 33(6). pp. 877—888 (2003)
13. Kohonen, T.: Self-organization and associative memory. Springer-Verlag New York, Inc., NY, USA. 3rd Edition.(1989)
14. Kolodner, J.L.: Case-based reasoning. Morgan-Kauffman, San Mateo, CA, USA (1993)
15. Martín, Q., De Paz, J.F., De Paz, Y., Pérez, E.: Solving TSP with a modified kohonen network. European Journal of Operational Research, Elsevier B.V (2007)
16. Nealon, J. and Moreno, A.: Applications of Software Agent Technology in the Health Care domain. Birkhauser, Whitestein series in Software Agent Technologies.(2003)
17. Snidaro, L.,Foresti, G.L.: Knowledge representation for ambient security. Expert Systems, Blackwell Publishing. Vol. 24(5). pp. 321—333 (2007)
18. Voos, H.: Agent-Based Distributed Resource Allocation in Technical Dynamic Systems. In Proceedings of the IEEE Workshop on Distributed intelligent Systems: Collective intelligence and Its Applications (DIS'06). IEEE Computer Society, Washington, DC, USA. pp. 157-162 (2006)