Solving Complex Decision-Making Problems through Agent-Matrices Cooperation

Hao Lan Zhang[1], Jiming Liu[2], Yong Tang[3], and Chaoyi Pang[1]

[1] NIT, Zhejiang University, Ningbo, Zhejiang Province, China
[2] Hong Kong Baptist University, Kowloon Tong, Hong Kong
[3] South China Normal University, 55, West Zhong Shan Avenue, China
haolan.zhang@nit.zju.edu.cn, jiming@comp.hkbu.edu.hk,
ytang@scnu.edu.cn, chaoyi.pang@csiro.au

Abstract. Making decisions in a dynamic environment is complicated and indefinite because of various unpredictable factors and large volumes of information. It has become a key issue for modern organizations to have efficient systems and tools to support rational decision-making in problem-solving processes, particularly, for problems with a lack of pre-defined structure. This paper provides a novel agent-based framework for Decision Support Systems (DSS) based on the analysis of the major problems in existing DSS. The proposed multi-agent-based framework, namely the Agent-Matrices framework, provides an open, efficient and flexible architecture for DSSs. The Agent-Matrices framework overcomes the compatibility and connectivity problems in most traditional DSS applications. This framework utilises Matrices to allow agents to acquire information, self-upgrade, perform tasks, travel to other Matrices, and be reused. This paper primarily introduces the methods used for coordinating the Matrices and agents to solve a complex problem.

Keywords: Agent-based Systems, DSS, Agent-Matrices.

1 Introduction

Agent architectural design has become a primary issue in the intelligent agent area as more and more concrete developments has emerged in the intelligent agent area such as JACK (agent development environment) [1], KAoS [2], etc. Existing agent architectural design methodologies brought forth several basic questions including: How would agent architectural design impact on the efficiency of an agent-based application? What are the major factors affecting the agent architectural design methodologies? Researchers in the field are pursuing answers to these questions.

In this paper we suggest several new methods for agent architectural design based on Agent-Matrices framework, which includes the concept of Matrices, the core components of the framework, etc. In this framework, agents are assembled through a mediator, i.e. the Matrix, which can coordinate agents to solve a complex problem.

The reminder of this paper is organized as follows: the next section describes the core components in the Agent-Matrices framework. Section 3 introduces the Matrices concept. Section 4 illustrates the structural design of Matrices. Section 5 introduces the unified Matrices structure and the last section concludes the research findings.

2 The Matrix Concept

The Matrix concept in the Agent-Matrices framework suggests a new method for agent cooperation and coordination, which incorporates and extends the agent society concept into the Matrix design. Unlike the middle-agent-based DSS, the Matrix not only acts as a middle-agent or an agent facilitator but also as a living platform for agents. The Matrix provides a small community environment for agents, where agents can self-upgrade and cooperate with other agents.

The unified Matrices structure provides a larger and more comprehensive structure that allows the cooperation among various Matrices. This unified Matrices structure extends the agent society concept that heterogeneous agents are distributed in various agent groups (communities) and each group is dominated by a Matrix. These groups, i.e. Matrices, form a large scale of interconnected Matrix network as shown in Figure 1.

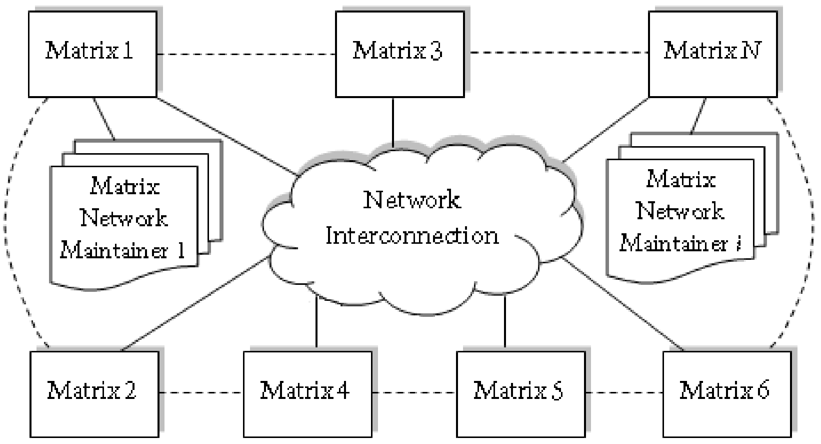

Fig. 1. Matrices Interconnection

2.1 Components in the Agent-Matrices Framework

An Agent-Matrices system basically consists of four major components, which include a number of agents, a number of Matrices, several databases (or knowledge-bases) that can be accessed by the Matrices, and an Agent-Matrices network maintainer for managing the connections between the Matrices and agent transportations. Table 1 lists the core components deployed in the Agent-Matrices framework and the major sub-components used in each of these core components. An Agent-based Matrix normally assembles a group of agents to solve the problems for a specific organization. Beyond the advantages of utilizing a flexible and open structure in the Agent-Matrices framework, the unified Matrices structure can support the cooperation between local agents and external (remote) agents.

Table 1. Core Components of Agent-Matrices

Core Components	Major Sub-Components
Matrix	Matrix register, Matrix control panel, Matrix learning centre
Agents	BDI Model, Capability register, Travel control centre, DSC container
Databases	Data, rules, information, etc
Matrix Maintainer	Matrix distribution information centre

This unified Matrices structure allows the heterogeneous agents in different domains (Matrices) to work cooperatively to solve complex problems without redundantly developing new agents for organizations. For instance, an agriculture department's Agent-Matrices-based DSS temporarily needs the population statistics of a specific region for a specific task analysis. It is costly to develop a new agent or several new agents to perform this task. Fortunately, a similar Agent-Matrices-based agent has been developed in a statistical company and this agent is also plugged into the company's Agent-Matrices-based Matrix. Therefore, the agriculture department could just send their requests to the statistical company's Agent-Matrices-based DSS for the results. This framework is cost-effective for the organizations that temporarily need outsourcings.

2.2 The Structural Design of Matrices

Based on the concepts of the Matrix, the four fundamental components in the Matrix are listed, which include:

(1) the agent society that provides a virtual space to agents;
(2) the Matrix learning centre that acquires information, i.e. new Domain-Specific-Component (DSC) items from the external environment;
(3) the Matrix control panel which is the core part that mainly handles agent matching, requests processing, and resource allocation;
(4) the DSC Usage Centre that allows the functional components to be used/reused by agents.

Each Agent-Matrices-based agent carries a DSC container, which holds a number of DSC items. Each DSC item can perform one (or several) specific task(s) and produce results; and each item can be plugged/ unplugged into/from the DSC container, which offers a flexible structure for upgrading agents' capabilities. These DSC items in an agent represent the agent's problem-solving capabilities. The Matrix control panel manages the agent society through establishing relationships with agents. An agent society can be regarded as the aggregation of a set of agent-relationships.

The Matrix-Agent connection design is based on a hybrid network topology. The connections between agents and Matrix are centralised, whereas the connections between agents are decentralised. The major role of the Matrix is to organise the agents to accomplish the tasks sent by users.

3 A Unified Matrices Structure for Matrices Cooperation

Each Matrix in the Agent-Matrices framework is connected by a number of agents that can deal with various problems in different domains. However, not every request can be solves by the internal agents in a Matrix. Once a request cannot be solved in a Matrix, the Matrix will forward the request to its nearest or most familiar Matrix for further processing. In this paper, a unified Matrices structure is employed based on the previous work [9, 10]. This is the fundamental motivation of establishing the unified Matrices structure in the Agent-Matrices framework.

3.1 Matrix Searching Algorithms for Service-Provider Matrices

Two primary guidelines are deployed for searching a service-provider Matrix based on a requesting Matrix in a unified Matrices structure [3], including the Most familiar partner method and the Supplemental partner method. The following sections incorporate the Agent-Matrices framework with the previous work conducted for unified Matrix design structure [10].

There are two situations when dealing with a request for cooperation (searching for external Matrices): one situation is to search for a partner Matrix that has similar capabilities to the requesting Matrix; the other situation is to find a partner Matrix that can provide some services that the requesting Matrix does not have. For instance, when an accounting agent in a company's Matrix system requires last year's regional taxation statistics, then it would look for another accounting agent that could provide regional statistical data, and the cooperation between these two agents might happen regularly. In this case, we use the most familiar partner method.

The following example explains the matching process. There is a huge natural disaster in the region where this company is located and the company's accounting agent requires a climate forecast report to analyse: (1) whether it is necessary to inject more funds to strengthen the company's buildings; (2) the amount of funds. In this situation, the requesting Matrix is looking for a very unfamiliar service provider Matrix, which has very different functionalities. Therefore, the supplemental partner method is employed.

3.2 Most Familiar Partner Method

We use a Matrix capability description table to describe the functionalities and capabilities of a Matrix as shown in Table 2.

Table 2. Matrix Capability Descriptions

Agent Functionality Keywords	Agent ID	Domain
Salary statistics, superannuation expenses, facility maintenance, etc.	Agent 1	Accountant
Warehouse stocks, sales condition, transportation expenses, etc.	Agent 2	Accountant
....	....	....
Market share history, market trends, major rivals' products, etc.	Agent i	Marketing

This table contains the information on all the connected agents' capabilities and the domain information, which indicates agents' major tasks or responsibility. Each agent's capability is described by several keywords and these keywords also represent the capabilities of the Matrices as Figure 2 shows [10].

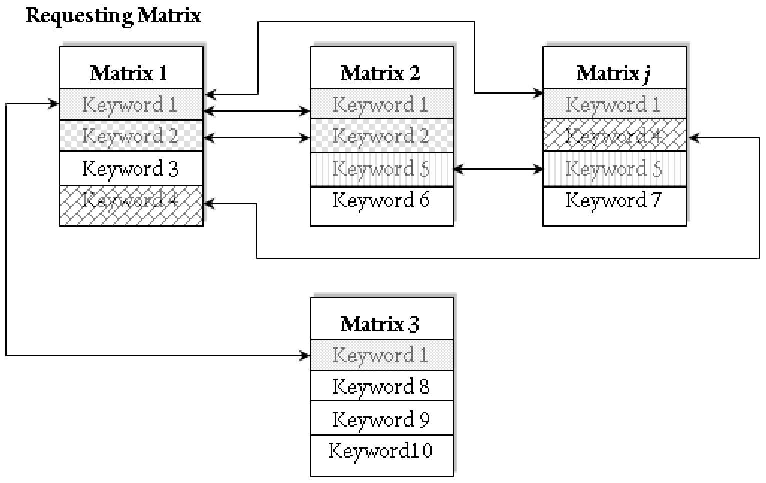

Fig. 2. Intersections of Matrices Capabilities [3]

In Figure 2, we aggregate all the keyword descriptions of the other Matrices, which intersect with Matrix 1's keyword descriptions. If there is one keyword intersection between the requesting Matrix and a corresponding Matrix, we add a score, namely the intersection score, for these two joined Matrices, expressed as $R_{i \to j}$, which means Matrix i and j intersect and their intersection score is $R_{i \to j}$.

If a corresponding Matrix has the most keyword intersections with a requesting Matrix (compared to other Matrices), then this Matrix is regarded as the most familiar partner of the requesting Matrix.

When there are two or more corresponding Matrices having the same number of keyword intersections with a requesting Matrix, then the Matrix that has keyword intersections least shared by other Matrices is selected as the most familiar partner. Thus, the calculation of the most familiar partner method can be expressed as follows. In the following example, we search a most familiar partner of Matrix 1.

Step 1. $\forall S_i \in \Omega, \forall K^j \in S_i$
where $i = 1, 2, n$. S_i denotes a set of keyword aggregation of Matrix i; Ω denotes the total aggregation of all keywords in all Matrices. $j = 1, 2, m$; j denotes the keyword number in a set; K_j denotes the keyword j in a keywords set; Φ denotes empty set.

$\exists(S_1 \cap S_j \neq \Phi) \to (R_{1\to j} = |S_1 \cap S_j|)$

$\exists(S_1 \cap S_j \neq \Phi) \to (R_{1\to j} = 0)$

where S_1, S_j denote the keywords set 1, j of Matrix 1, j, respectively. $R_{i\to j}$ denotes the intersection score of Matrix 1 to Matrix j.

Step 2. If $R_{i\to j}$ is the maximum value among all the other intersection scores with Matrix 1, then Matrix j is the most familiar partner of Matrix 1 and the most familiar partner process is completed. If $\exists(R_{1\to j} = R_{1\to g})$ then go to Step 3 ($R_{i\to g}$ denotes the intersection score of Matrix 1 to Matrix g).

Step 3. $\exists(K^i \subset (S_1 \cap S_j \cap ... \cap S_x)) \to (R_{1\to j} = R_{1\to i} + 1/M_i)$
where M_i denotes the total number of the Matrices (including the Matrix 1) that have the same intersections with Matrix 1. S_x denotes a keyword set of Matrix X that intersects with S_1 and S_j at keyword K_i.

$R_{1\to j} = \sum_{i=1}^{S_{1\to j}} (R_{1\to i} + 1/M_i)$

where $S_{1\to j}$ denotes the total elements in Set 1.

$\exists(K^i \subset (S_1 \cap S_g \cap ... \cap S_y)) \to (R_{1\to g} = R_{1\to i} + 1/Z_i)$

where Z_i denotes the total number of Matrices (including the Matrix 1) that have same intersections with Matrix 1. S_y denotes a keyword set of Matrix y that intersects with S_1 and S_g at keyword K_i.

$R_{1\to g} = \sum_{i=1}^{S_{1\to g}} (R_{1\to i} + 1/Z_i)$

If $R_{1\to j} < R_{1\to g}$ then Matrix g is the most familiar partner of Matrix 1 and the process is over. If $R_{1\to j} = R_{1\to g}$ then go to Step 4.

Step 4. Choose the nearest neighbour as the most familiar partner of Matrix 1. If there are two or more nearest neighbours then randomly choose a Matrix among them.

3.3 Supplemental Partner Method

In previous work [10], the supplemental partner method has been introduced. In such a method, a service-provider Matrix of a requesting Matrix is choosen that has minimum keywords intersections with the requesting Matrix. The supplemental partner method seeks a service provider Matrix that has a minimum intersection score. We still use the same example to search a supplemental partner for a specific Matrix. The detailed steps are described as follows, which mainly based on the previous work [10].

Step 1. It is the same procedure as in the most-familiar-partner method .

$\forall S_i \in \Omega, \forall K^j \in S_i$

$\exists(S_1 \cap S_j \neq \Phi) \rightarrow (R_{1\rightarrow j} = |S_1 \cap S_j|)$

$\exists(S_1 \cap S_j \neq \Phi) \rightarrow (R_{1\rightarrow j} = 0)$

Step 2. If $R_{1\rightarrow j}$ is the minimum value among all the other intersection scores with Matrix 1, then Matrix j is the supplemental partner of Matrix 1 and the supplemental partner process is completed.

If $\exists(R_{1\rightarrow j} = R_{1\rightarrow j} = 0)$ then choose a nearest partner among these same intersection score Matrices. If there are two or more nearest neighbours then randomly choose a Matrix among them.

If $\exists(R_{1\rightarrow j} = R_{1\rightarrow j} > 0)$ then go to Step 3 ($R_{1\rightarrow g}$ denotes the intersection score of Matrix1 to Matrix g).

Step 3. In this step, the supplemental partner method seeks a minimum intersection score.

$R_{1\rightarrow j} = \sum_{i=1}^{S_{1\rightarrow j}} (R_{1\rightarrow i} + 1/M_i)$

$R_{1\rightarrow g} = \sum_{i=1}^{S_{1\rightarrow g}} (R_{1\rightarrow i} + 1/Z_i)$

If $R_{1\rightarrow j} > R_{1\rightarrow g}$ then Matrix g is the supplemental partner of Matrix 1 and the process is over. If $R_{1\rightarrow j} = R_{1\rightarrow g}$ then go to next step.

Step 4. Choose a nearest partner among these same intersection score Matrices. If there are two or more nearest neighbours, then randomly choose a Matrix among them.

The most familiar partner and supplemental partner methods are basically for matching capabilities and functionalities among Matrices. For more specific agent capability search and agent relationship management, we use a novel Agent-rank algorithm. The agent-rank algorithm is the most efficient means to search a service provider for a request. The most familiar partner and supplemental partner methods help to narrow the searching range for the agent-rank algorithm. These two methods based on Matrix matching are complementary; the combination of these two methods could enhance the matching efficiency and accuracy.

4 The Optimised Model and Performance Evaluation

The unified Matrices structure introduced in the previous sections has its limitations on central control and fault-tolerance. These limitations lead to the circumstances of low efficiency in the matching process, massive traffic chaos in the cooperation process and vulnerability to partial system breakdown. In the Agent-Matrices framework, an optimised model, namely Super-node model, is introduced.

4.1 Super-Node Model in the Matrices Cooperation

Based on the early work [10], a super-node model is suggested to tackle the above problems. The super-node model [4] has been extensively applied to many online systems, such as KaZaA [5], SkypeNet [6], etc. In a super-node model, a number of nodes are selected as super nodes, which manage a limited number of other nodes. The major advantages of the super-node model include reducing time and bandwidth for search, enhancing manageability, etc.[4]. This model avoids mesh peer-to-peer connections, which minimizes the probability of the occurrence of concurrency. The concurrency problem often causes repetitions of communications, which increases network traffic volume.

The criteria of determining whether a computing terminal is suitable to be a super-node are quite simple, which include: (1) it must have a high bandwidth connection; (2) it needs to have a reasonable computing capability; (3) it should be flexible and efficient in entering and exiting a network. Generally, each super-node Matrix is directly connected to a Matrix network maintainer.

4.2 Performance Evaluation

The Agent-Matrices framework employs the optimised methods including the most familiar partner method, supplemental partner method, and super-node model. Figure 3 and 4 show the performance comparison between optimised and non-optimised Agent-Matrices-based framework. The following figures illustrate the performance comparison.

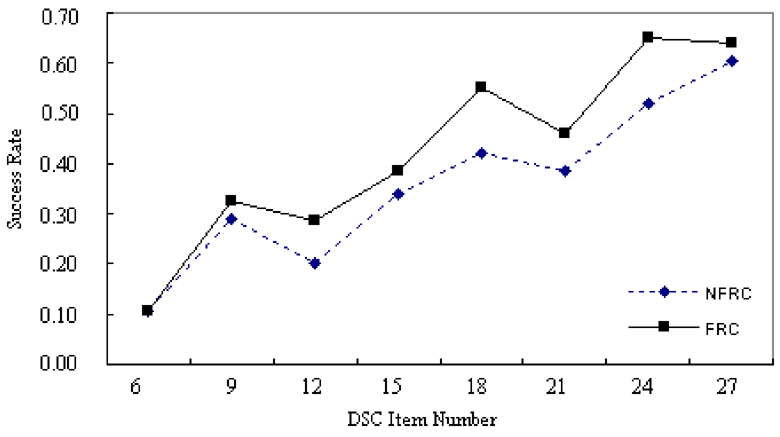

Fig. 3. Success rate comparison based on NFRC and FRC (200 requests)

The performance comparison clearly indicate that the most familiar partner and supplemental partner methods adopted in the Agent-Matrices framework can successfully improve the matching success rate and reduce the matching

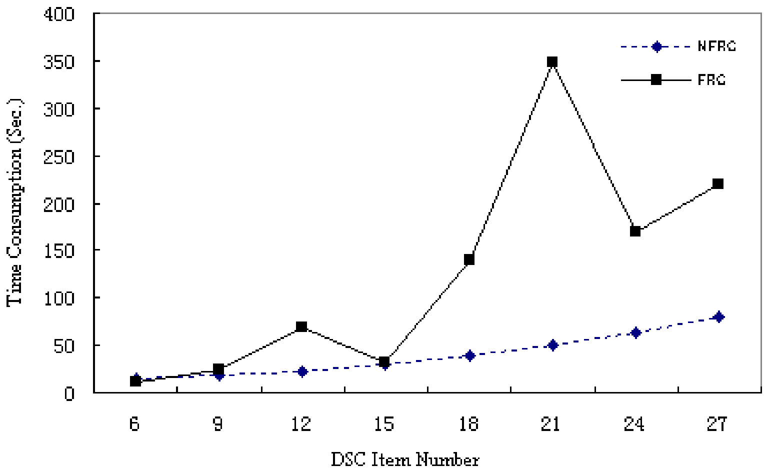

Fig. 4. Time consumption based on NFRC and FRC (200 requests)

time consumption. The application of the super-node model in the Unified Matrices Framework enhances the fault-tolerance capability and the system manageability. In the unified Matrices structure, the super-nodes are those selected Matrices that play a coordinator role in processing the other Matrices' requests. These super-nodes receive the requests from the other Matrices and search the corresponding Matrices.

5 Conclusion

Autonomy oriented computing (AOC) unifies the methods for effective analysis, modelling, and problem-solving in complex systems [7, 8]. This paper addresses AOC-by-prototyping and AOC-by-self-discovery issues through introducing the Matrix concept [7].

The Matrix provides a virtual platform for intelligent agents through managing Matrix-agent relationships in an agent society. The Matrix maintains a superior transmission performance in its virtual platform through combining centralized and decentralized topologies. The Matrix also presents seemly mobility as the peer-to-peer connections between agents only exist temporarily.

The Matrix employs a Domain-Specific-Component usage centre to assure that every agent is upgradeable. Different from agent mediators and facilitators in other middle-agent-based systems, a Matrix in the Agent-Matrices [9, 10, 11] framework not only coordinates its internal agents but also provides its internal agents with various information resources, such as the DSC items, data resources, etc. An agent can perform self-upgrade through obtaining new DSC items from the Matrix and replacing dated items. This Matrix-agent connection design simplifies the self-upgrade processes of an agent, which are rather complicated in many agent-based systems.

The unified Matrices structure consists of a number of Matrices groups; and one of the Matrices in each group is selected as a super-node to manage a group. These selected super-node Matrices normally possess superior computing performances; and each super-node Matrix manages the group through using a Matrix network maintainer that contains the Matrix distribution information in the group.

Acknowledgement. This project is funded by the Zhejiang Philosophy and Social Science Project Grant (No. 11JCSH03YB), National Excellent Science and Technology Fund for Overseas Scholars (Ministry of HR of China, 2011[508]), Ningbo Nature Science Grant (No. 2012A610060, 2012A610025), Ningbo Soft Science Grant (No. 2012A10050), Ningbo International Cooperation Grant (No. 2012D10020) and National Nature Science Grant (No. 61272480).

References

1. Evertsz, R., Fletcher, M., Frongillo, R., Jarvis, J., Brusey, J., Dance, S.: Implementing Industrial Multi-agent Systems Using JACK. In: Dastani, M., Dix, J., El Fallah-Seghrouchni, A. (eds.) PROMAS 2003. LNCS (LNAI), vol. 3067, pp. 18–48. Springer, Heidelberg (2004)
2. Bradshaw, J.M., Dutfield, S., Benoit, P., Woolley, J.D.: KAoS: Toward an Industrial-Strength Generic Agent Architecture. In: Software Agents, pp. 375–418. AAAI/MIT Press, Cambridge (1997)
3. Padgham, L., Winikoff, M.: Prometheus: A Methodology for Developing Intelligent Agents. In: Proc. of AAMAS, Bologna, Italy, pp. 37–38 (2002)
4. Fiorano, Software Inc.: Super-Peer Architectures for Distributed Computing. Technical report, Fiorano Copyrights (2007)
5. KazaA, Inc.: How Peer-to-Peer (P2P) and Kazaa Software Works. Technical report, KazaA Copyrights (2006)
6. Skype, Inc.: Skype Guide for Network Administrators. Technical report, Skype Copyrights (2005)
7. Liu, J., Jin, X., Tsui, K.C.: Autonomy Oriented Computing: From Problem Solving to Complex Systems Modeling. pp. 8–10. Springer (2004)
8. Liu, J., Jin, X., Tsui, K.C.: Autonomy oriented computing (AOC): Formulating computational systems with autonomous components. IEEE Transaction on Systems, Man and Cybernetics, Part A: Systems and Humans 35(6), 879–902 (2005)
9. Zhang, H.L., Leung, C.H.C., Raikundalia, G.K.: Topological analysis of AOCD-based agent networks and experimental results. Journal of Computer and System Sciences 74(2), 255–278 (2008)
10. Zhang, H.L., Leung, C.H.C., Raikundalia, G.K.: Matrix-Agent Framework: A Virtual Platform for Multi-agents. Journal of Systems Science and Systems Engineering 15(4), 436–456 (2006)
11. Zhang, H.L., Pang, C., Li, X., Shen, B., Jiang, Y.: A Topological Description Language for Agent Networks. In: Sheng, Q.Z., Wang, G., Jensen, C.S., Xu, G. (eds.) APWeb 2012. LNCS, vol. 7235, pp. 759–766. Springer, Heidelberg (2012)