Dynamic Reconfiguration of Multi-Agent Systems Based on Autonomy Oriented Computing

Li Tao [1] and Zili Zhang [1,2]

[1]Laboratory of Intelligent Software and Software Engineering,
Southwest University, Chongqing, 400715, China
{tli, zhangzl}@swu.edu.cn
[2]School of Engineering and Information Technology, Deakin University,
Geelong, VIC 3217, Australia
zzhang@deakin.edu.au

Abstract

Dynamic reconfiguration has been listed as one of the key challenges in support of agent adaptation to environments, which has attracted much attention of researchers world wide. To tackle this tough problem, an Agent-Based Dynamic Reconfiguration Model (ADRM) is proposed from the autonomy-oriented computing (AOC) point of view. The ERA (Environment–Reactive rules–Agents) algorithm used in AOC is improved to support the organization formation behavior, which is essential in dynamic reconfiguration. To test the efficiency of this model and the effectiveness of different reactive behaviors, the performance of this model was investigated under different selection probabilities.

1. Introduction

Multi-agent systems and agent-based hybrid intelligent systems are well suited to engineering complex software systems [1]. But how to dynamically reconfigure an agent system based on different tasks and changing environment is a key issue which remains unsolved.

Actually, dynamic reconfiguration is listed as one of the key challenges in support of agent adaptation to environments [2], which is called 'run-time reconfiguration and redesign'. To solve this problem, researchers proposed a few techniques from different aspects, which include employing different kinds of middle agents[3], reconfiguring individual agents at micro-level[4], modifying the geographical distribution of an application[5], and so on. Dynamic reconfiguration of multi-agent systems is similar to dynamic coalition formation in some parts. It is the process of coalition management, and which also involves coalition formation. The researches on dynamic coalition formation can be found in [7][8][9], etc.

The emphasis of this paper is on the dynamic reconfiguration with different tasks and environment changes. There is no efficient solution reported to date.

In this paper, an emerging computational paradigm called Autonomy Oriented Computing (AOC) [10] has been used for modeling dynamic reconfiguration of agent-based systems. AOC has been effectively used in a variety of domains covering constraint satisfaction problem solving [11], optimization [12], etc.

Using AOC framework to model dynamic reconfiguration of agent-based systems, we need to clearly describe what are the environment, primitive behaviors and behavioral rules of autonomous entities (here are agents), and the interactions between agents and their environment. The dynamic reconfiguration of agent-based systems is then reduced to the self-organization of AOC systems.

Moreover, dynamic reconfiguration can be generalized to Constraint Satisfaction Problem (CSP) [6]: (1) The agents who would like to provide services can be regarded as variables X in CSP; (2) The web environment in which agents exist is the domain D of variables; (3) The requirements of tasks can be naturally described as constraints C.

From the analysis above, it is evident that the solutions for CSP can be used for dynamic reconfiguration of agent-based systems. ERA (Environment–Reactive rules–Agents) is a multi-agent oriented approach to solving constraint satisfaction problems [11]. It has been successfully used for solving typical CSPs, such as n–queen problems and coloring problems. In this paper, the algorithm IERA (Improved ERA) which is improved upon ERA will be used for supporting the organization formation behavior, which is essential in dynamic reconfiguration.

Proceedings of the IEEE/WIC/ACM International
Conference on Intelligent Agent Technology (IAT'06)
0-7695-2748-5/06 $20.00 © 2006 IEEE

2. Agent-Based Dynamic Reconfiguration Model ADRM

Reconfiguration behavior of agent systems performs during the period of task implementation. For better illustrating the problem we are going to solve, the International Trading Agents Competition for Supply Chain Management (http://www.sics.se/tac) (TAC SCM) was selected as an example, and some minor changes are made for our case. The changes are: (1) The suppliers with own quoted price dynamic change . (2) The component supply of each supplier is instable. (2) Customer requirements are allowed to change before production.

In such scenarios, neither typical static supply chain management models nor the strategies of good players in TAC SCM game are appropriate. Because they ignored the dynamic changing situation showed above.

Generally speaking, the capabilities required to support reconfiguration include: (1) information service mechanisms able to locate appropriate suppliers, and determine their abilities; (2) description methods used for describing the capability of suppliers so that manufacturer can compare two suppliers easily; (3) searching algorithms for finding appropriate suppliers in a given time; (4) control mechanisms for maintaining and reconfiguring agent systems.

In essence, the ADRM architecture is composed of several different agents, including Member Agent (MAgent), Construct Agent (CAgent), Employee Agent (EAgent), Yellow Page Agent (YP) [3], and Reconfiguration Controller Agent (RC), as shown in Figure 1. MAgent is the services provider (components suppliers in the scenario). CAgent undertakes the task for searching appropriate MAgents. In the example, on behalf of the manufacturer, the CAgents search and negotiate with suppliers to decide which supplier is best for manufacturer. EAgents are those who carry out tasks. They are the suppliers who have been selected by manufacturer. YP can perform some basic functions, such as allowing MAgents registering and organizing the registration information. RC monitors and controls the reconfiguration behavior of the whole agent system.

Specifically, to achieve the goal of reconfiguration, YP has been used for agents discovery and registration in ADRM. Methods for Constraint Solving Problems (CSP) has been borrowed for second issue. Every requirement of user can be regarded as one constraint, and the ability of each agent can be represented as a constraint value. IERA algorithm will be used for solving problem (3). An algorithm ADSR will be developed for question (4). Due to the limit of space, this paper focuses on the main idea of the ADRM model and the IERA algorithm.

As Figure 2 shows, ADRM model performs reconfiguration behavior as follows: During the period of task implementation, YP will receive the applications of MAgents

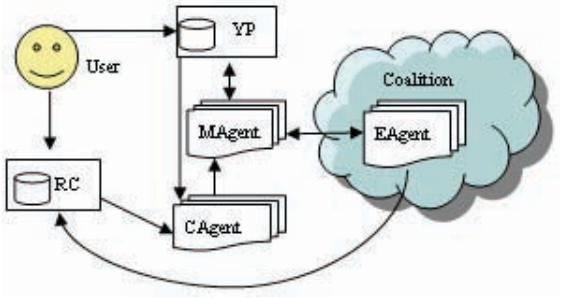

Figure 1. The ADRM system architecture

(Suppliers) about joining or exiting continuously. At the same time, YP will arrange the information of joining MAgents in a two dimension lattice as the logical environment. RC is used for producing CAgents, dispatching CAgents with searching tasks according to user's requests. Moreover, RC should monitor the EAgents implementation situation. If some of the EAgents capability decrease to a certain range, e.g., 4% of the EAgents have disabled, it will conduct reconfiguration behavior. If the number of leaving EAgents is too large, e.g., 40% of the EAgents have disabled, the RC will disband the old agent system, build a new one, instead of reconfiguring the old ones.

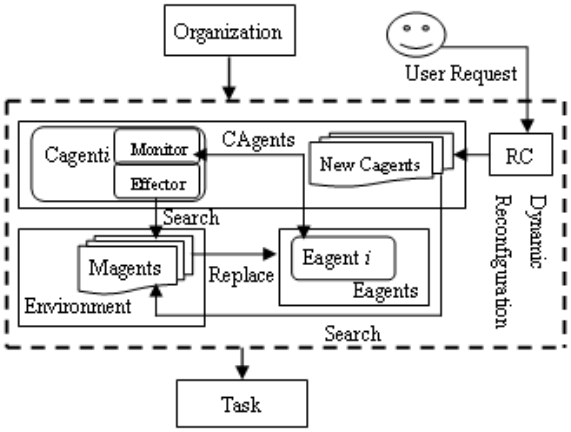

Figure 2. The ADRM system architecture

3. Formal Framework of ADRM Model

In this section, the formal framework of ADRM will be briefly introduced. In our method, the CAgent is the autonomous entity in AOC. The lattice formed by MAgents registration information is represented as multi-entity environment. Thus, the problem of finding a solution to a dynamic reconfiguration is reduced to that of how a group of CAgents find a certain desired state by performing their primitive behaviors in such an environment.

Proceedings of the IEEE/WIC/ACM International
Conference on Intelligent Agent Technology (IAT'06)
0-7695-2748-5/06 $20.00 © 2006 IEEE

3.1 Definition of Key Elements in ADRM

As discussed, there are five different types of agents in ADRM. One of the key components is CAgent. It is an autonomous entity which implements the self-organization computation.

Definition 1 (**Construct agent**) : An construct agent (CAgent) is a tuple <CID, BV, BVL, CL>, where CID is the id of CAgent. BV is the best utility value found by this CAgent until now. BVL is the location of the MAgent who owns the best utility value. CL points the position of this CAgent in logical environment.

Definition 2 (**Construct agent primitive behaviors**): The primitive behaviors set for CAgent is R= {better-move, best-move, across-move}, which are improved from ERA system [11].

better-move: A CAgent moves to a better position with a probability of better-move. The better-move behavior can be expressed as function Ψ_{better}:

$\Psi_{better}(x, y) = j \mid j \in [low, |D_x|]$,
$e(x, j).value \geq e(x, y).value$

Where

$$low = \begin{cases} 1 & y - s \leq 1, \\ y - s & y - s > 1. \end{cases}$$

Note that in this function, *e(x,y)* is the current position of CAgent in logical environment. *x* is the row number. *y* is the column number. *s* means search step. *low* is the lower limit of the search range.

best-move: A CAgent moves to a best position with a probability of best-move.

$\Psi_{best}(x, y) = j \mid j \in [low, top]$,
$(\forall t \in [low, top])\ e(x, j).value \geq (x, t).value$

Where

$$low = \begin{cases} 1 & y - s \leq 1, \\ y - s & y - s > 1. \end{cases}$$

$$top = \begin{cases} n & y + s \geq |D_x|, \\ y + s & y + s < |D_x|. \end{cases}$$

Note that in this function, *e(x,y)* is the current position of CAgent. *x* is the row number. *y* is the column number. *s* means search step. *low* is the lower limit of the search range, and *top* is the upper limit.

across-move: A CAgent moves to a new row with a probability of across-move.

$$\Psi_{across}(x, y) = \begin{cases} i & \forall t[1, n], \sum_{j=1}^{|D_i|} e(i, j) > \sum_{j=1}^{|D_t|} e(t, j) \\ Random(n) & \end{cases}$$

Definition 3 (**Environment E**): Particularly, here, **E** is a lattice composed by utility values of all the MAgents.

The data structure of **E** can be defined as:

– E =$\langle \text{row}_1, \text{row}_2, \ldots, \text{row}_n \rangle$.

– $\forall$i $\in$ [1,m], $\text{row}_i \Leftrightarrow$ the domain of i*th* CAgent$_i$ $\Leftrightarrow$ D_i, so row_i has $|\text{D}_i|$ columns.

– row_i =<lattice_{i1}, lattice_{i2}, …, $\text{lattice}_{i|D_i|}$>.

– e(i, j).value records the j*th* value of row_i. It is the utility value of MAgent_{ij}.

The utility value is calculated through the matching of constraint conditions. Due to the limit of space, the detailed calculation process is omitted.

3.2 IERA Algorithm

IERA algorithm which has been improved from ERA undertakes the searching task in dynamic reconfiguration. The main improvements include: (1) the definition of autonomous entity primitive behaviors; (2) the implementation ways of autonomous entity primitive behaviors; (3) the interaction content between environment and autonomous entities; (4) the goal of the whole system.

The IERA algorithm includes three main function modules. Function Initialize initializes the parameters of CAgents, such as search footstep, primitive behaviors probability, etc. Function SelectBehavior uses Roulette method [6] to select one primitive behavior according to different behavior probability. Function OrganizationFormation is the main program of IERA. The CAgents will do some primitive behaviors to find the proper Eagents, which will form the organization to undertake user's tasks.

4. Experiment and Discussion

The experiment in this paper focuses on two aspects. Can IERA converge? What effects do different better-move/best-move/random-move probability ratios make? The experiments were conducted with the number-of-MAgents = {32, 100, 1000, 10000}, number-of-CAgents= {4, 10, 50, 100}, better-move/best-move/random-move probability between [0.0, 1.0]. And each case runs 100 times.

The experiment shows that if only the behaviors probability is appropriate, the IERA can convergence in a receivable time. The experiment also shows that the behaviors probability has impact on IERA convergence and the speed of convergence significantly. Figure 3 is one of the experiment results to show the relationship between behaviors probability and average convergence speed of CAgents.

By analyzing the results, some valuable rules for behavioral probability settings can be obtained. (1) The probability of across_p should be neither too small, nor too big. Because if across_p is too small (e.g. across_p=0.0), the CAgent will be restrict to a small area. If it is too large, the whole agent system will be instable. (2) When across_p fixed, the bigger the better_p is, the faster convergence speed will be have. (3) Best_p should not be too big. When

across_p fixed and best_p $\in$ [0.1, 0.3], fastest average convergence speed can be obtained. Because best-move is a traversal in an area that will waste too much time to find a solution.

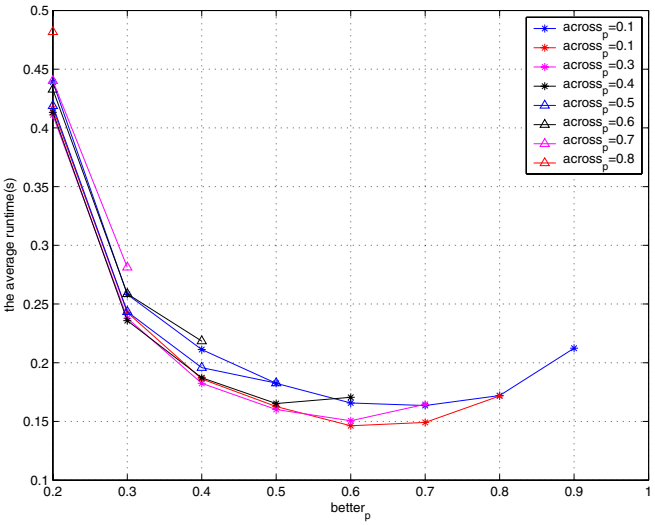

Figure 3. Relationship between behaviors probability and average convergence speed of CAgents. (Parameters Setting: MAgent_Number=1000, CAgent_Number=50, best_p=1.0–across_p–better_p)

5. Conclusions

The emphasis of this article is on the dynamic reconfiguration of agent-based systems at macro-level with different tasks and with changing environment. In this paper, we have described an AOC-based approach to solving multi-agent dynamic reconfiguration. Dynamic reconfiguration of agent-based systems has been modeled by AOC framework. The environment, primitive behaviors and behavioral rules of CAgents, and the interactions between agents and their environment have been clearly defined. In addition, the multi-agent oriented approach to solving CSP called ERA has been proposed for supporting reconfiguration problems. The Experiment of IERA algorithm shows that behaviors probability have significant influence on the convergence and the convergent speed. Some practical rules for behavioral settings have been found.

Although we have done some work on multi-agent dynamic reconfiguration, a lot of work remains. In the future, we hope to further improve the IERA algorithm by introducing heuristic rules, and to extend our reconfiguration approach to virtual organization formation and management in Grid Computing domain.

References

[1] Z. Zhang and C. Zhang. *Agent-Based Hybrid Intelligent Systems: An Agent-Based Framework for Complex Problem Solving*. LNAI 2983, Springer, 2004.

[2] M. Luck, P. Mcburney and C. Preist. A Manifesto for Agent Technology: Towards Next Generation Computing. *Autonomous Agents and Multi-Agent Systems*, Vol. 9, No. 3, pp. 203-252, 2004.

[3] K. Sycara, J. Lu, M. Klusch, S. Widoff. Matchmaking among Heterogeneous Agents on the Internet. Proc. AAAI Spring Symposium on Intelligent Agents in Cyberspace, Stanford, USA, 1999.

[4] M. Hannebauer. *Autonomous Dynamic Reconfiguration in Multi-Agent Systems: Improving the Quality and Efficiency of Collaborative Problem Solving*. LNAI 2427, Springer, 2002.

[5] N. De Palma, L. Bellissard and M. Riveill. Dynamic Reconfiguration of Agent-Based Applications. Third European Research Seminar on Advances in Distributed Systems, Madeira Island (Portugal), April 23rd-28th, 1999.

[6] S. Russell, P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall (2nd edition), 2002.

[7] K. Lerman, O. Shehory. Coalition Formation for Largescale Electronic Markets. Proceedings of the International Conference on Multi-Aent Systems, 2000.

[8] C. Preist, A. Byde, C. Bartolini. Economic Dynamics of Agents in Multiple Auctions. Proceedings of the 5th International Conference on Autonomous Agents, ACM Press, pp. 545-551, 2001.

[9] J. Yamamoto, K. Sycara. A Stable and Efficient Buyer Coalition Formation Scheme for E-Marketplaces. Proceedings 5th International Conference on Autonomous Agents, Montreal, Canada, ACM Press, 2001.

[10] J. Liu, X. Jin and K. Tsui. *Autonomy Oriented Computing: From Problem Solving to Complex Systems Modeling*. Springer, 2005.

[11] J. Liu, H. Jing, Y. Y. Tang. Multi-agent Orinented Constraint Satisfaction. *Artificial Intelligence*, Vol. 136, No. 1, pp. 101-144, 2002.

[12] K. C. Tsui and J. Liu. Evolutionary Diffusion Optimization. Proceedings of the 2002 Congress on Evolutionary Computation, Honolulu, Hawaii, May 12-17, 2002.