International journal of Computer Science & Network Solutions August.2013-Volume 1.No1
http://www.ijcsns.com ISSN 2345-3397

Optimization Load Balancing Scheduling Algorithm in Distributed Multi-Agent Systems

A.Maryam Jafari, B. Hossein SHirazi, C. Sayed Ali razavi Ebrahimi

Department of computer, Sama University, Dezfoul, Iran
E-mail: maryamjafari_rk@yahoo.com

Department of Information Technology, Malek Ashtar University of Technology, Tehran, Iran
E-mail: shirazi@mut.ac.ir

Department of computer, Payamenur University, Tehran, Iran
E-mail: Dr_ali_razavi@yahoo.com

Abstract

Load balancing is very important in the distributed multi-agent system that it can improve the performance of the distributed multi-agent system effectively. In the existing methods scheduling optimization wasn't done for load balancing. In large-scale distributed multi-agent systems consisting of one million of agents using partial information about each agent's communication search time of the optimal agent set reduced, so algorithm execution time decreased. Optimized Load Balancing Scheduling Algorithm is proposed in the Distributed Multi-Agent Systems. The proposed algorithm performs distribution of computation load among servers for improving system performance. The proposed algorithm achieves optimized scheduling using parallel computing on all servers and partial information of agent communication. The simulation was performed to evaluate the performance of the algorithm. Experimental results show that the proposed algorithm is less time consuming. The experiments indicate that proposed Algorithm has the advantage of short execution time in large-scale distributed multi-agent systems.

Keywords: agent, Scheduling, multi-agent system (MAS), Load Balancing Algorithm.

I. Introduction

The distributed multi-agent systems are intelligent system; in them agents perform specific tasks to achieve specific goals automatically and intelligently. These agents connect and cooperate to achieve their goal. There is variety of agents in the distributed multi-agent systems such as buffering agent, broker agent, etc. The purpose of the distributed multi-agent system is to help and guide users for doing jabs and it predict results and suggest decisions.
The load balancing is one of the important issues in distributed multi-agent system. In the distributed simulation, agent-based Load imbalance happens in large-scale for the reasons such as frequent, dynamic changes in the tasks, agent load and agent communication (Long et al., 2011).
Load imbalance has great impact on performance of the distributed multi-agent system. The

Goals of the load-balancing algorithm is to minimize time and cost in distributed multi-agent systems. Often optimized scheduling did not accomplish in methods presented for load balancing.
So far there is no single definition for agent (Chen and Su, 2003). An agent is a computer system located in some environments and it operate autonomously to achieve its goals. There are two definitions for explaining the agent characteristics. In Ones Define, generally main features of agent are independence, reaction, interaction, and innovation (Long et al., 2011). And the other more common definition is that agent has three important features: independence, consistency and coordination. Independence means that it can have self-learning. Agents must actively complete their tasks without external intervention (human or other software). Consistency is the ability of the agent to understand and adapt with its external environment and multi-agent system. Coordination is an important feature, it means that agents are concordant and work together. Therefore, one agent cannot determine alone intelligent multi-agent system capability that shows by coordination between agents (Ilaria *et al.*, 2008; Sergio *et al.*, 2008).
A multi-agent system consists of numbers of agents that work together to solve problems that is beyond individual ability or knowledge of each inventory (Gao *et al.*, 2009). Agents interact together, by exchanging messages through the computer Network (Gao *et al.*, 2009). Multi-agent systems have been widely applied to analyze and solve complex problems. AMCC system (Chen and Tu, 2009), MASCOT (Sadeh *et al.*, 2001), AgentStra (Li, 2007), an agent-based intelligent system for supporting coordinate manufacturing execution and decision-making (Gao *et al.*, 2009), Agents International (Li and Li, 2010), for Intelligent Decision Support in Intensive Information Architecture Medicine (Santos *et al.*, 2009) and Human cognition as an intelligent decision support system for plastic products' design (Sancin *et al.*, 2010) have been developed.
Dynamic load balancing is allocation of limited resources among multiple tasks over time (Long et al., 2011). The multi-agent systems have limited resources and time constraints, therefore, needed to focus on agent schedules in them. There are different methods for load balancing such as partitioning, based on the economic theory, rule-based approach, and adaptive techniques. Partitioning method can properly distribute the workload generated by agents between the available distributed resources (Long et al., 2011). For example, adaptive method (Deng *et al.*, 2000), fast adaptive balance method (Zhang *et al., 2009*), considering the irregular shape of the regions (Vigueras *et al.*, 2010). However, the partitioning methods have heuristic nature, and spend much time, so they significantly are affected by the average response time (Long et al., 2011). Paper (Chow and Kwo, 2002) proposes load-balancing model based on given credit to each agent and it improve the volume loading more than balance oriented normal programs. Paper (Jang and Agha, 2004) proposed two mechanisms to improve the adaptive agent allocation: one mechanism aims minimizing agent communication cost, while the other mechanism attempts to prevent overloaded computer nodes from negatively affecting overall performance. In the paper (Miyata and Ishida, 2007) network of agents considered as a "small world", so the load balancing is performed based on the community. This method evaluates the communities to select the most appropriate set of agents to be moved. Paper (Shin *et al.*, 2009) proposed, load balancing based on the agent mode that it perform efficient and justly allocation of resources to agents. This method spends much time for searching the best mode.
Another scheduling algorithm is the distributed approximate optimized scheduling algorithm with partial information (DAOSAPI) (Long et al., 2011). This algorithm needs partial information of agent communication for searching, and do not need agent status information.

There are problems in dynamic load balancing of distributed multi-agent system. In the concentrated way there are two problems: the bottleneck and failure point. Due to heuristic nature of the partitioning method, it is much time consuming. The method of the immigration should be considered. Once proper purpose Search and did not find, search must be done again. By dividing societies, the cost will increase (Long et al., 2011).
The Load balancing algorithm in distributed multi-agent system is proposed that reduce time and cost. This paper introduces load balancing algorithm for distributed multi-agent systems using all system resources to reduce algorithm execution time and cost of agent communications. The proposed algorithm performs distribution of computation load among servers for improvement of system performance. The proposed algorithm achieves optimized scheduling using parallel computing on all servers and partial information of agent communication. The way was adopted for migration Agents selection and destination server selection that reduce communication costs.

II. Analysis of dynamic load balancing in distributed multi-agent system

Multi-agent system is a distributed network of multiple servers that agents reside within the servers. One of the servers selected as a central server in order to maintain time synchronization between servers in the load balancing algorithm. Load balancing Coordinator agent at the central server is for global communication. in all servers there is an agent as manager agent for communication with central server, server load monitoring, server overloading detection, performing load balancing algorithm, the selection of the optimal agent set for migration, selecting the destination server and migrating agents. Other agents in the server are as work agent that can communicate with the other agents directly. Agent communication architecture is shown in the figure 1.
When the manager agent on a server detects that server overloaded it will send time synchronization request for implement load balancing algorithm to the central server. The request is sent from manager agent to coordinator agent within the central server. If time synchronization established for other server, overloaded server must wait, otherwise the message will be sent by central server to all servers and implement time synchronization.
Assume multi-agent systems have M server that n agents are among those distributed. The load capacity of all servers is the same that divided among agents resident in server equally. Agents within the same server and two different servers can communicate with each other for cooperation and coordination.
The agents communication cost within the same server (intra-server interaction) is much less than two different servers communication cost. Three objectives that followed to improve the system performance are:

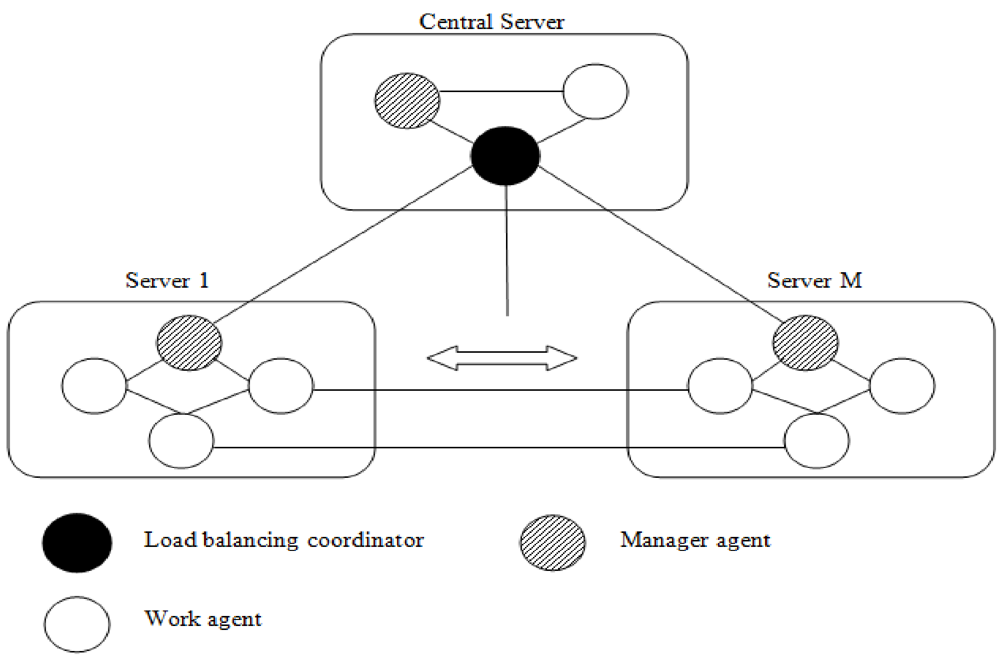

Figure.1. Communication architecture between agents

When the manager agent on a server detects that server overloaded, it will send time synchronization request for implement load balancing algorithm to the central server. The request is sent from manager agent to coordinator agent within the central server. If time synchronization established for other server, overloaded server must wait, otherwise the message will be sent by central server to all servers and implement time synchronization.
Assume multi-agent systems have M server that n agents are among those distributed. The load capacity of all servers is the same that divided among agents resident in server equally. Agents within the same server and two different servers can communicate with each other for cooperation and coordination.
The agents communication cost within the same server (intra-server interaction) is much less than two different servers communication cost. Three objectives that followed to improve the system performance are:

• Agent load distribution among servers
• Reducing the communication cost between servers
• Reducing load balancing algorithms execution time

At First, the load distribution among servers is required for use of all available resources in the system. The agents use the limited resources to perform tasks of server where they reside, if the server overloaded, it causes reduction in system efficiency, and need to transfer the overload to the server that has extra capacity. Load transfer can be done through agent migration to the other servers.

In the load-balancing algorithm, agents that are associated together must be put in the same server for reduction in communication costs, since the agents' communications, cost in same server is less than the cost of communication between two different servers. The Proposed algorithm use parallel computing. The manager agent in overload server performs small piece of code and other pieces of code is executed in parallel by the Manager agent at another server to reduce the execution time. Also, partial information about the agent communication requires for execution time reduction.

In this work we have try to achieve all three goals stated in the above. We define three policies in order to implement load balancing algorithm. The first policy is selecting the optimal agent set for migration. The optimal agent set selection for each server is based on the number of communication each agent within overloaded server has with the destination server. The agents have most communication with the destination server and least communication whit the overloaded server, are added to the optimal agent set.

The second policy is selecting the destination server. The server has the most communication with the optimal agent set and remaining capacity must be chosen as destination server.

Third policy is the migration of optimal agent set to the destination server. The agent set migrates as a group together to the destination server for reducing execution time of Algorithm.

III. Proposed load balancing algorithm in distributed multi-agent system

In this section, first we define the variables used for agent, agent load, server, server load, the threshold. All variables are shown in Table I, and then the proposed load-balancing algorithm is described.

Table I
Notations

S_j	Server j (j = 1,2,….,m)
a_{ij}	agent i in server j (i = 1,2,3,…, $n_j = \sum i$, $N = \sum_{j=1}^{M} n_j$)
L_{s_j}	computation load of S_j
$L_{a_{ij}}$	computation load of a_{ij}
L_{ps_j}	computation load acceptance capability of server S_j
$C_{a_{n_i},a_{n_j}}$	communication count between a_{n_i} and a_{n_j} ($a_{n_i} \in S_i$, $a_{n_j} \in S_j$, i,j ≤ M)
C_{s_o,s_j}	communication count between S_o and S_j (o,j ≤ M ^ o≠j)
C_{a_{io},s_j}	communication count between a_{io} and S_j (o,j ≤ M ^ o≠j)
C_{m_{oj},s_j}	communication count between m_{oj} and S_j (o,j ≤ M ^ o≠j)
$C_{m_{oj},s_o-m_{oj}}$	communication count between m_{oj} and $s_o - m_{oj}$ (o,j ≤ M ^ o≠j)
Th	Threshold of computation load of servers

IV. Definitions

Agents in each server are performed together into thread. $a_{ij}a_{ij}$ is agent i in server j. load $L_{s_j}L_{s_j}$ is load of server $s_j\ s_j$ at time t. Let us assume that agent loads within the same server is equal, also achieved through the following equation:

$$L_{a_{ij}}(t) = \frac{L_{s_j}(t)}{n_j} \tag{1}$$

n is the number of agents in server $s_j s_j$.
The load threshold at a server is a server computing capability. The load threshold shown as Th, that determines whit power of system or average workload of the overall system. There are two statuses for each server:
$L_{s_j}L_{s_j} > Th$ server $s_j s_j$ is overloaded.
$L_{s_j}L_{s_j} < Th$ server $s_j s_j$ is not overloaded.
Agents are connected together within the same server or different servers for cooperation and coordination. $C_{a_{io},s_j}C_{a_{io},s_j}$ is the number of agent's i communication in server $s_o s_o$ whit server $s_j s_j$ that is calculated using the following equation:

$$\mathrm{C}_{\mathrm{a}_{io},s_j} = \sum_{l=1}^{n_j} C_{a_{io},a_{lj}} \tag{2}$$

$C_{s_o,s_j}C_{s_o,s_j}$ is the number of connections between server $s_o s_o$ and server $s_j s_j$ at time t and determines as following:

$$C_{s_o,s_j} = \sum_{i=1}^{n_o} C_{a_{io},s_j} \tag{3}$$

B. Proposed load balancing algorithm

The proposed load balancing algorithm perform based on three polices explained in the previous section. The optimal agent set will be determined for each server, and then servers perform calculations using Equation 4.

$$C_{m_{oj},s_j} - C_{m_{oj},s_o - m_{oj}} + a * L_{ps_j} \tag{4}$$

$C_{m_{oj},s_j}C_{m_{oj},s_j}$ is the number of server's communications, $s_j s_j$, with optimal agent set $m_{oj}m_{oj}$ chosen for it. $L_{ps_j}L_{ps_j}$ is server load capacity and coefficient α is the relative importance of acceptance. Then the overloaded server selects the highest credit server as the destination server. And then optimal agent set for server is migrated to it. Protocol of load balancing process is shown in Figure 2.

The Implementation stages of the algorithm are as follows:

Step 1: When the MA within a server sees the server load is greater than the load threshold then it sends coordinate request to coordinator agent at the central server. If time synchronization didn't perform for another server, the central server send time synchronization massage to the overloaded server and other server, otherwise overloaded server must wait.

Step 2: when the MA in $s_o s_o$ receives the time synchronization message from central server, it broadcasts a message contain (ID of $s_o s_o$, the overload, the agent load and partial information about agent communication).

Step 3: The MA in $s_j s_j$ receives message from MA in $s_o s_o$, if $L_{s_j} \geq L_{s_j} \geq Th$, it send "not accept requests" to MA in $s_o s_o$. If $L_{s_j} L_{s_j}$<Th, it starts the procedure for selecting the optimal agent set.

Step 4: The capacity of the server $s_j s_j$ compared with $L_{migrate} L_{migrate}$ to determine number of agents that can be migrated to server $s_j s_j$.

$$\left.\begin{array}{l} L_{ps_j} \langle L_{migrate} \; n_{migrate} = \dfrac{L_{ps_j}}{L_{a_o}} \\[2em] L_{ps_j} \rangle L_{migrate} \; n_{migrate} = \dfrac{L_{migrate}}{L_{a_o}} \end{array}\right\{ \qquad (5)$$

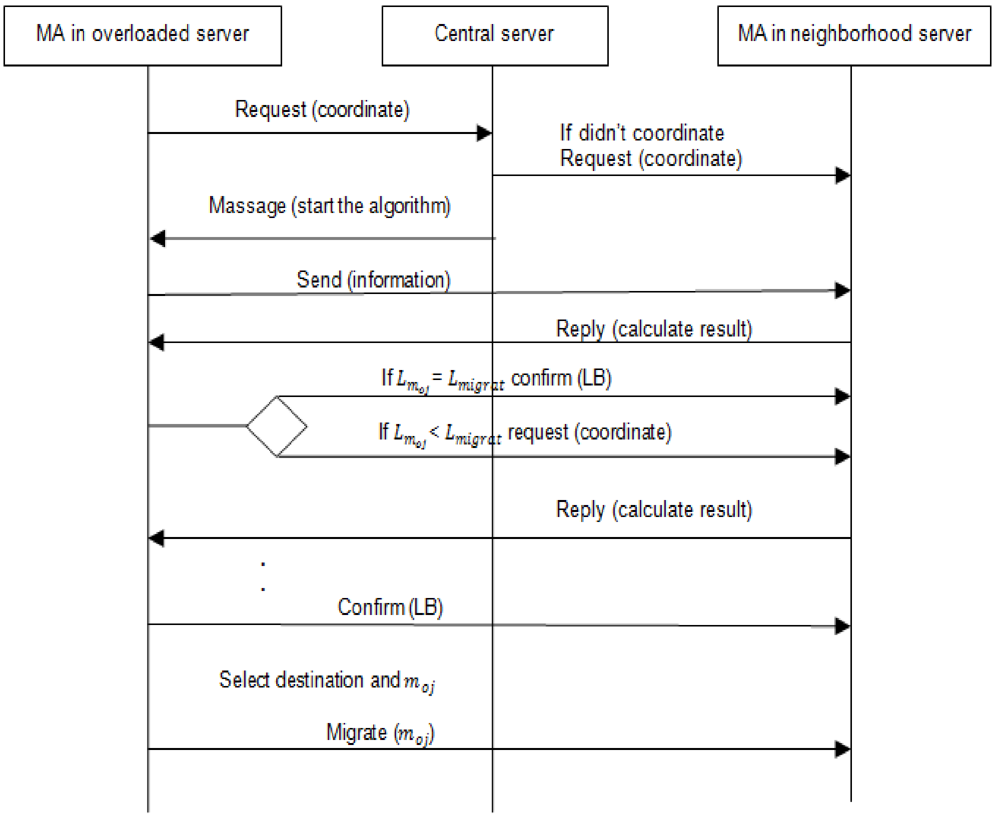

Figure.2. Protocol of load balancing process.

Step 5: At this stage, according to the number of immigration agent $s_j s_j$, The agents have the most communication with $s_j s_j$ and least communication whit $s_o s_o$ are added to the optimal agent set $m_{oj} m_{oj}$. This means that the credit is given to each agent based on the number of communications. The agent have the most credit is selected for migration to $s_j s_j$.

Step 6: the MA in $s_j s_j$ according to the amount of credit selected agents for migration and also the load capacity, calculates credit of the server through Equation 4.

Step 7: the MA in $s_j s_j$ sends message to the MA in s_o s_o contain the value calculated using equation (4) and the optimal agent set.

Step 8: The MA in s_o s_o selects destination server s_j s_j with max (C_{m_{oj},s_j} C_{m_{oj},s_j} - $C_{m_{oj},s_o - m_{oj}} C_{m_{oj},s_o - m_{oj}}$ + $\propto * \propto *$ $L_{ps_j} L_{ps_j}$) and the optimal agent set of $s_j s_j$ as immigration agents.

If $L_{ps_j} L_{ps_j} >$ $L_{migrate} L_{migrate}$, the optimal agent set migrate from $s_o s_o$ to $s_j s_j$ as a group, otherwise the MA in $s_o s_o$ broadcast the message contain (ID of $s_o s_o$, the rest overload, the agent load and new partial information about agent communication) and the algorithm repeats of step 3 until all load is transferred. At end The MA in $s_o s_o$ begin immigration procedure.

V. Optimized load balancing algorithm scheduling

In this section analyze the algorithm process and optimized load balancing scheduling. Finally, analyze the time complexity of the proposed algorithm.

VI. Analysis of optimized load balancing scheduling

To reduce the execution time of load balancing algorithm, parallel computing, partial information about the agent communication and effect of load capacity at selecting destination server is used. Parallel computing means performing optimal agent set selection for each server at same server simultaneously, So, overloaded server performs a piece of code for finding destination server. Having partial information about each agent's communication within overloaded server (ID of server and ID of agent) credit determination time for each agent for migration is reduced, Credit of each agent according to the number of communication with overloaded server and the number of communication with destination server. In DAOSAPI algorithm, overloaded server sends undirected graph of the intra-server agent communication to other servers, so search time of optimal agent set was increased, but in the proposed algorithm, Overloaded server sends partial information about agent communication to other servers, so search time of optimal agent set is decreased. Also the destination server capacity affects on Select it, so to prevent repeat algorithm for the transmission of all overload.
Two Works has been done to reduce the execution time of algorithm. The algorithm has two important pieces of code. One of them executes on the overloaded server and another execute as parallel computing on other servers. So the MA in all servers performs a small part of the code in same time to reduce the execution time.
On the other hand, the optimal agent set is selected and as a group to migrate to destination server so the execution time of algorithm will be reduce than any individual agent. Also the destination server capacity affects on Select it, so to prevent repeat algorithm for the transmission of all overload. Pseudo code of proposed algorithm is shown in Table II.

B. Analysis of time complexity of the proposed algorithm

The proposed algorithm has two important pieces of code that a piece of code is implemented in overloaded server and other piece of code is implemented on other servers at same time. The piece of code on overloaded server is selection of the destination server for migration agents and the time complexity of it is O (M). The second piece of code is implemented on other servers for selects optimal agent set. The time complexity of second piece is O (NC). (N is the average number agents per server and C is the average number communication per agent). The time complexity of the proposed algorithm is O (MNC). The time complexity of SLB algorithms in (Miyata and Ishida, 2007) is O (NC|A1|) (N is the number migrated agents, C is the average number of agents which an agent interacts with and |A1| is the number of agent in the overloaded server) and the time complexity of DAOSAPI algorithm in (Long et al., 2011) is O (M) (M is the number of server and N is the number of agent per server). The proposed algorithm compared with SLB and DAOSAPI, it significantly reduces the time complexity and can respond to the dynamic load balancing more quickly. Although the proposed algorithm has a low communication delay, it has no significant impact on its performance.

VII. Experimental results

This section describes simulation to test our technique. And results obtained from the proposed algorithm with DAOSAPI and SLB compared. This simulation was implemented in C# and executed on a machine with 2.67GHz CPU and 4GB of memory.

VIII. Simulation Settings

We have considered simulation settings as follows: a mass Multi-agent system is composed of several servers in a network. Initially certain number of agents to servers allocated. The agent communication pattern is random and the maximum number of communication defined for agents. The load distributed among servers randomly and the load of per server distributed between all agents within this server equally. The Computation threshold of servers is equal Average computational load of all servers.
First the distribution network of servers, which the number of agents per server was set to 100 and the number of servers, was set to 10, 20, 50, and 100. Average computation time of algorithm is shown in Table 3.
Table 4 shows the average computation time of the simulation in which the number of servers was set to 10 and the number of agents per server was set to 100, 200, 500, and 1000.

Table II
The pseudo code of the proposed algorithm

Code segment in the overloaded server	Code segment in the other servers
Send a request time synchronization to the main server	Receive the message from the main server for time synchronization
if receive the message of main server **then** start algorithm	
send a message with (ID, $L_{a_{io}}$, L$_{\text{migrate}}$, and partial information of the agent communication) to the MA in the other servers	Receive the message from the MA in the overloaded server s$_0$
	if $L_{s_j} <$ Th **then**
	Calculate L_{ps_j} and $n_{j_{migrate}}$
	for $a_{io} \epsilon S_o$ **do**
	if $m_{oj}.count \le n_{j_{migrate}}$ **then**
	calculate C_{a_{io},s_j} and $C_{a_{io},s_o-a_{io}}$
	Select a_{io} into m_{oj} whit max (C_{a_{io},s_j} - $C_{a_{io},s_o-a_{io}}$)
	End if
Receive the message from MA in the other servers	End for
	End if
Select m_{oj} and S_j whit max (C_{m_{oj},s_j} - $C_{m_{oj},s_o-m_{oj}}$ $+ \propto * L_{ps_j}$)	Send a message whit m_{oj} and C_{m_{oj},s_j} - $C_{m_{oj},s_o-m_{oj}}$ and L_{ps_j} to the MA in the overloaded server s$_0$
if $L_{m_{oj}} <$ L$_{\text{migrate}}$ **then**	
Send a message whit (ID, L$_{\text{migrate}}$ - $L_{m_{oj}}$, $L_{a_{io}}$ and partial information of the rest agent communication) to the MA in the other servers	
…………	

IX. Simulation results analysis

When there are 100 agents on average in the servers, the proposed algorithm takes less time than DAOSAPI in the process of load balancing. However, when the number of agents is increased to 500 or more, the proposed algorithm takes much less time than SLB and DAOSAPI, because it makes use of distributed parallel computing. Also the proposed algorithm takes less time than DAOSPI, since it uses partial information about the agent communication and load capacity of destination server.

Tables III and IV show the average computation time of the proposed algorithm, SLB and DAOSAPI algorithms.

In the massively multi-agent system when the number of agents within each server increases maybe the number of connections per agent increases. Therefore, the effect of increasing number of agent communication is investigated on algorithm execution time. Paper (Miyata and Ishida, 2007) defined six neighbors for each agent. That is, each agent interacts with its neighbor. Here's the number of servers was set to 10 and the number of agents per server was set to 1000 and the number of agent communication was set to 6, 20, 40, 60, 80 and 100.

As seen in figure 3, when the number of agent communication increases, the execution time of the proposed algorithm increases very low.

Table III
Comparison of execution time between SLB and Proposed

100	50	20	10	Number of server
0.21	0.08	0.03	0.01	SLB (s)
0.046	0.03	0.02	0.01	Proposed (s)

Table IV
Comparison of execution time between SLB, DAOSAPI and Proposed versus the number of agents per server

1000	500	200	100	Number of agent per server
0.42	0.11	0.03	0.01	SLB (s)
0.078	0.062	0.047	0.031	DAOSAPI (s)
0.046	0.031	0.015	0.01	Proposed (s)

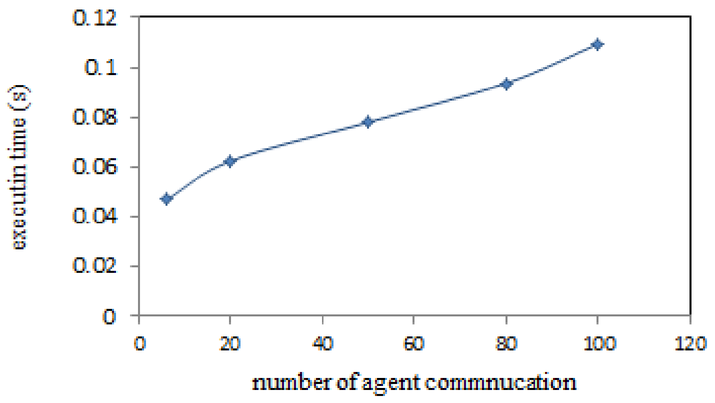

Figure 3: execution time of proposed algorithm versus the number of communication per agents.

X. Conclusions

In this paper, we studied the distributed multi-agent system; we remarked that load balancing in distributed system has special significance. To reduce time and communication costs for optimized scheduling a load balancing algorithm is required. In the cases reviewed, search time of migration agent set increased. We introduce the Optimized Load Balancing Scheduling Algorithm in Distributed Multi-Agent Systems that it reduced execution time and cost. The proposed algorithm improve performance of the system and execution time using parallel computing, but also solve the bottleneck and failure point problems. This algorithm needs to know partial information of agent communication for selecting optimal agent set. So search time and execution time of algorithm will decrease. The algorithm performance was tested by simulation. Since the proposed algorithm use partial information of communication agent for selecting optimal agent set and load capacity to choose target server, the simulation results show that proposed algorithm reduces execution time of load balancing algorithm compared to SLB and DAOSAPI.

References

Chen J.J, Su S.W (2003). Agent Gateway: a communication tool for multi-agent systems, *Inf. Sci*, Vol(150), pp 153–164.

Chen R.S and Tu M.R (2009). Development of an agent-based system for manufacturing control and coordination with ontology and RFID technology, *Expert Systems with Applications*, Vol(36), pp 7581–7593.

Chow K.P and Kwo Y.K (2002). On load balancing for distributed multiagent computing, *IEEE Transactions on Parallel and Distributed Systems*, Vol(13), pp 787–801.

Deng Y.F, Peierls R.F and Rivera C (2000). An adaptive load balancing method for parallel molecular dynamics simulations, *Journal of Computational Physics*, Vol(161), pp 250–263.

Gao Y, Shang Z and Kokossis A (2009). Agent-based intelligent system development for decision supportin chemical process industry, *Expert Systems with Applications*, Vol(36), pp 11099–11107.

Ilaria B, Giuseppe C and Giuseppe S (2008). Development of a multi-agent model for production scheduling in innovative flexible manufacturing system. *Manufacturing Systems and Technologies for the New Frontier, The 41st CIRP Conference on Manufacturing Systems*, pp 279–283.

Jang M.W and Agha G (2004). Adaptive agent allocation for massively multi-agent applications, *International Workshop on Massively Multi-Agent Systems*, pp 25–39.

Li S.L and Li J.Z (2010). AgentsInternational: integration of multiple agents, simulation, knowledge bases and fuzzy logic for international marketing decision making, *Expert Systems with Applications*, Vol(37), pp 2580–2587.

Li S.L (2007). AgentStra: an Internet-based multi-agent intelligent system for strategic decision-making, *Expert Systems with Applications*, Vol(33), pp 565–
571.

Long Q, Lin J and Sun Z (2011). Agent scheduling model for adaptive dynamic load balancingin agent-based distributed simulations. *Simulation Modelling Practice and Theory*, Vol(19), pp 1021–1034.

Miyata N and Ishida T (2007). Community-based load balancing for massively multi-agent systems, *International Workshop on Coordination and Controlin Massively Multi-Agent Systems*, pp 28–42.

Sadeh N.M, Hildum D.W, Kjenstad D and Tseng A (2001). MASCOT: an agent-based architecture for dynamic supply chain creation and coordination in the internet economy, *Production Planning & Control*, Vol(12), pp 212–223.

Sancin U, Dobravc M and Dolšak B (2010). Human cognition as an intelligent decision support system for plastic products' design, *Expert Systems with Applications*, Vol(37), pp 7227–7233.

Santos M.F, Portela F, Vilas-Boas M, Machado J, Abelha A and Neves J (2009). for Intelligent Decision Support in Intensive Information Architecture Medicine, *Issue 5*, Vol(8), pp 810-819.

Sergio I, Eduardo M and Arantza I (2008). Using cooperative mobile agents to monitor distributed and dynamic environments. *Inf. Sci*,. Vol(178). pp 2105–2127.

Shin S.Y, Lee H.C, Song S.K and Youn, H.Y (2009). A load balancing scheme for multi-agent systems based on agent state and load condition, *International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, pp 121–127.

Vigueras G, Lozano M, Orduna J.M and Grimaldo, F (2010). A comparative study of partitioning methods for crowd simulations, *Applied Soft Computing*, Vol(10), pp 225–235.

Zhang D.L, Jiang C.J and Li S (2009). A fast adaptive load balancing method for parallel particle-based simulations, *Simulation Modelling Practice and Theory*, Vol(17), pp 1032–1042.