Task Schedulable Problem and Maximum Scheduling Problem in a Multi-agent System

Bin Li, Xiaowei Zhang, Jun Wu, Junwu Zhu
School of Information Engineering
Yangzhou University
Yangzhou, China
Email: lb@yzu.edu.cn, zxw9289zxw@163.com, j_wu@vip.sohu.net, jdkr@163.com

***Abstract*—Tasks scheduling is a key problem in multi-agent system, traditional tasks scheduling methods can't be applied to new application areas of the MAS such as emergency system. In order to apply the Agent method to these new areas, a multi-agent system model is built in this paper, and corresponding task schedulable problem and maximum scheduling problem are defined based on this multi-agent system model. Task schedulable problem is modeled using flow network, and it is proved that maximum flow algorithm can be used to solve such problem, which means the problem can be solved in polynomial time. Furthermore, by analyzing the flow network model, a necessary and sufficient condition which can be used to determine whether tasks can be scheduled is gained and proved. Three approximation algorithms have been proposed to solve the maximum scheduling problem. The experiment results show that all above algorithms can get pretty solutions in solving maximum scheduling problem, and the approximation ratio for optimal solution of these approximation algorithms are all larger than or equal to 0.5 even though the resource ratio is very low.**

***Index Terms*—task scheduling, task schedulable problem, maximum task scheduling problem, MAS, flow network, NP complete**

I. INTRODUCTION

Multi-Agent system (MAS) is composed of a number of autonomous units named agent which can interact with each other to improve the problem-solving ability of the overall system which greater than simple addition of problem-solving ability of single agent. MAS is an open system, and the change of external environment may continually generate new tasks. However, single Agent has limited capability to get solution about information and problem. At this time, multiple Agents are needed to complete these tasks by collaboration. Therefore, task scheduling problem is a key problem in MAS. An effective task scheduling method is necessary for MAS to accomplish the tasks with high efficiency. Recent years have seen a lot of work on task scheduling methods, which can be broadly classified as follow: (1) task scheduling based on auction protocol [1, 2]; (2) Coalition formation methods [3, 4]; (4) Distributed task scheduling methods [5, 6]; (5) Decision theory based methods [7, 8]; (6) Manisterski et al. [9] discusses the possibilities of achieving efficient allocations in both cooperative and non-cooperative settings. They propose a centralized algorithm to find the optimal solution.

All of above works have improved the efficiency and have enriched the theory of MAS. However, these works can't be used in many new applications of MAS such as Emergency system [10] etc. The reasons are: (1) in these applications all agents must mandatory executing tasks allocated by distributor, and the agent's local goal must not conflict with the distributor's global goal. However, above task scheduling methods such as auction based ones suppose that the agents are selfish, and have high degree of autonomy, so they would not execute the tasks if the distributor doesn't offer payment they expected. (2) Situation is urgency and tasks should be accomplished as soon as possible in above applications. However, task allocation of most above works is decentralized, and the task scheduling process is carried out by negotiating between agents, and the task scheduling process will be uncertainty and time-consuming, which cannot satisfy the requirements of timely task allocation in urgency situation of above application of MAS.

In order to make MAS suitable for above applications, we construct a MAS model based on the characteristics of the Emergency system, and then define task schedulable problem and maximum task scheduling problem in section 2. In section 3, we prove that task schedulable problem can be solved in polynomial time. Furthermore, we construct flow network based on the instance of task schedulable problem to show that the maximum flow algorithm can be used to solve the problem. We discuss the maximum task scheduling problem in section 4, we prove the problem to be NP complete by reducing X3C problem to it in polynomial time. In section 6, we propose three approximation algorithm (named Increased Greedy Algorithm, IGA; Decreased Greedy Algorithm, DGA; Iterative Minimum Cost Algorithm, IMCA) to solve the maximum task scheduling problem. We also do experiment to verify the performance of above algorithms in this section, and conclude our work at the last section.

II. PROBLEM DESCRIPTION

A. Multi-agent System Model

In this section, we design a MAS model according to characteristics of emergency system, C3I system etc. As

Project number: 60903130, BK2007074, BK2009698, BK2009699.
Corresponding author: Bin Li, Yangzhou University, Yangzhou, China. Email: lb@yzu.edu.cn.

© 2011 ACADEMY PUBLISHER
doi:10.4304/jsw.6.11.2225-2231

we can see from figure 1, the model includes a task manager, task interface layer, task schedule layer, and resource layer. Task manager is responsible for monitoring the overall system, and intervene the system when necessary who can receives the tasks from customers, and translates them into a certain format the system can understand.

The task interface layer comprised of task matching model, task template library. Task template library is used to store task templates which is a role set that complete corresponding tasks, and a task matching module which responsible for searching task templates in template library according to the task arrive at the system. The advantage of the mechanism is simple, fast and efficient, which can prettily instead of traditional task decomposition process and satisfy requirements of emergency systems.

Task scheduling layer includes resource discovery module, scheduling module, and role set. The role is equivalent to a sub-task which can be used to restraint the agent's actions. For example, if an agent plays the role of fire-fighting, it can only fight fire, and can not carry the wounded and other rescue tasks. Resource discovery module can find agents required to accomplish tasks in accordance with relationships between roles and agents, and send the information about these agents to the scheduling module. When the scheduling module received the information, it can easily schedule agents to execute tasks arrive at the system by scheduling algorithm. The algorithm and role enable the system to accomplish tasks effectively and possesses strong exception handling capability.

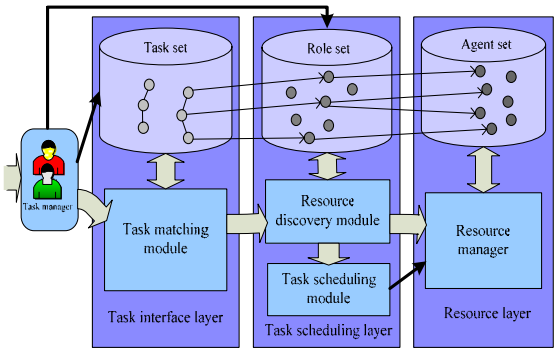

Figure 1. Multi-agent system mode

Resource layer includes a resource manager, and all agents that form the multi-agent system. Resource manager monitor the agent's action in system, and register the name, function, utilities, and role-agent relation information about agent when it entering the system, and delete these information when agent is in failure or exiting the system. Agents are ultimate executor of tasks, which encapsulate part of resources necessary to accomplish tasks. The agent has characteristics of autonomy, interactively, and initiative etc. In this system, the quality of task scheduling strategy determines the overall system's performance. Therefore, we define task schedulable problem based on agent's preferences in next section, and further discuss the problem below.

B. Problem definition

There is a set of agents: $A = \{ a_1, \ldots, a_m \}$, each of which encapsulate parts of needed resources to complete tasks. Let $R = \{ r_1, \ldots, r_l \}$ denote the collection of the role the agents can play. Each agent $a \in A$ can plays a number of roles in R, on the contrary, a role can also be played by a number of agents, which is defined by role-agent relations: $Re \subset R \times A$. Suppose a set of tasks $T = \{t_1, t_2, \ldots, t_n\}$ arrives at the system. Each task mapping is then represented by a tuple $< t, Re' >$, where $t \in T$ is the task, $Re' \subset Re$. The exact assignment of tasks to agents is defined by a task scheduling. Task scheduling generally includes two situations, the one is that all tasks in T can be accomplished by MAS, and the other is that only parts of the tasks in T can be accomplished by MAS of which we consider how to make the existing Agent resources to accomplish tasks of the system as much as possible, which known as the maximum scheduling problem. Formal definitions of these two problems are defined as follow:

DEFINITION 1. (TASK SCHEDULABLE PROBLEM) Given a set of tasks $T = \{t_1, \ldots, t_n\}$, a set of agents $A = \{ a_1, \ldots, a_m \}$, and a set of role $R = \{ r_1, \ldots, r_l \}$ the agents can play, $t_i \in T$ ($i = 1,2, \ldots, m$) is a multi-set on R, then task schedulable problem is a mapping φ: $R \rightarrow A$. which must satisfy the following constraints:

(1) If task t can be accomplished, every role related to t must have corresponding agents to play, that is for each r in multi-set t, there is a function $\varphi(r) = a \Rightarrow (r, a) \in Re, r \in R, a \in A$.

(2) The agent be scheduled can at most play a role at some time point, i.e. $\forall a \in A. | \{ r \in R : \varphi(r) = a) \} | \leq 1$

(3) All roles related to every task have corresponding agents to play, that is $\forall t \in T. \forall r \in R. \exists a \in A. S(r) = a$

DEFINITION 2. (MAXIMUM SCHEDULING PROBLEM, MAX-S) Given a set of tasks $T = \{t_1, \ldots, t_n\}$, a set of agents $A = \{ a_1, \ldots, a_m \}$, and a set of role $R = \{ r_1, \ldots, r_l \}$ the agents can play, $t_i \in T$ ($i = 1,2, \ldots, m$) is a multi-set on R, then the maximum task scheduling $\max_{T' \subseteq T} | T' |$ *s.t.* there exists a mapping $S : R \rightarrow A$ which satisfy following constraints:

(1) $S(r) = a \Rightarrow (r,a) \in \mathrm{Re}, r \in R, a \in A$,

(2) $\forall t \in T'. \forall r \in t. \exists a. S(r) = a$,

(3) $\forall a \in A. | \{ r \in R : S(r) = a \} | \leq 1$.

III. TASK SCHEDULABLE PROBLEM IN MAS

A. Construct flownetwork model

We construct flow network model in accordance with an instance I of task schedulable problem, and then determine whether the tasks can be scheduled on the flow network, the construction algorithm is as follows:

© 2011 ACADEMY PUBLISHER

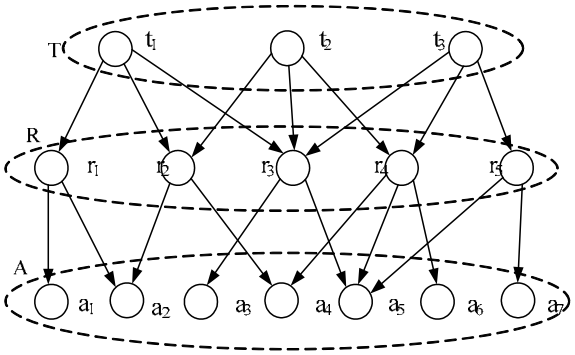

Figure 2. Directed graph of the problem

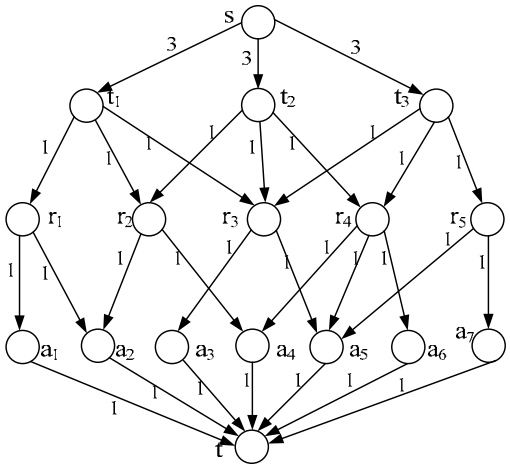

Figure 3. Flow network model of the problem

Algorithm 1. Construct Flow Network

1. Create direct graph (see figure 2), create directed graph $G = (V, E)$ in accordance with an instance of task scheduling problem, where the vertex set $V = T \cup R \cup A$ and the edge set $E = Ms \cup Re$, in which $Tr \subset T \times R$. and $Re \subset R \times A$.
2. Construct flow network (see figure 3), create the flow network model N as below:
   (a) Add source node s and sink node t to graph G.
   (b) For each task node t_i (i = 1, 2, …, n) in T, create an edge (s, t_i) from s to this node with capacity equal to out-degree of t_i (e.g. $c(s, t_i) = d_{out}(t_i)$)
   (c) For each agent node a_j (j = 1, 2, …, m) in A, create an edge from a_j to sink node t with capacity 1.
   (d) For each edge (r, a) in Re, $r \in R$, $a \in A$, and for each edge (t, r), $t \in T$ and $r \in R$, set the capacity be 1.

B. Solution of Task Schedulable Problem

As we will use different method to solve the tasks schedulable problem and maximum task scheduling problem, then we must determine whether tasks can be scheduled before we assigned tasks to agents of MAS.

THEOREM 1. Given a set of tasks $T = \{t_1, \ldots, t_n\}$, a set of agents $A = \{ a_1, \ldots, a_m \}$, and a set of role $R = \{ r_1, \ldots, r_l \}$ the agents can play, $t_i \in T$ (i = 1,2, …, m) is a multi-set on R, role-agent relations represented by $Re \subset R \times A$, then tasks are schedulable if and only if the maximum flow of the corresponding flow network N is equal to summation of out degree of all task nodes, which represented as follow:

$$|f| = \sum_{t_i \in T} d_{out}(t_i) . \quad (1)$$

PROOF. Suppose that all tasks in T can be accomplished by agents in A, which implies that we can find enough agents to play all roles relate to each task vertex $t_i \in T$ ($i = 1,2,\cdots,n$), then in corresponding flow network N, the flow value of each edge from vertex $t_i \in T$ to $r_j \in R$ ($j = 1,2,\cdots,l$) is 1. For the capacity constraints, note that $f(t_i, r_j) \le 1$ for all $t_i \in T$ and $r_j \in R$ ($j = 1,2,\cdots,l$). Intuitively we can see that the overall flow value of flow network N is maximum, and the value $|f| = \sum_{t_i \in T} d_{out}(t_i)$. On the contrary, if we know that $|f| = \sum_{t_i \in T} d_{out}(t_i)$ in corresponding flow network N, then the flow value of each edge (t_i, r_j) ($t_i \in T, r_j \in R$) must be 1. Thus from the flow-conservation property, we can know that each role vertex has related to an agent, which means there have enough agents to accomplish all tasks.

Theorem 1 shows that the task schedulable problem of such MAS model can be solved by the maximum flow algorithm[11], the current maximum flow algorithm complexity is $O(n^3)$, so the problem can be solved in polynomial time. By further analysis of the problem, we can get the following conclusions:

DEFINITION 3. There is a direct graph $G = (V, E)$, let $S \subseteq V$ be any vertex set of G, $H = \{(s,v) \mid s \in S, v \in V\}$ be the arc set of which arcs are all sourced by vertex in S, and $N(S)$ be the terminal vertex set of H (if $(s,v) \in H$, then $v \in N(S)$), then we say that $N(S)$ is neighbour set of S.

THEOREM 2. Given a set of tasks $T = \{t_1, \ldots, t_n\}$, a set of agents $A = \{ a_1, \ldots, a_m \}$, and a set of role $R = \{ r_1, \ldots, r_l \}$ the agents can play, $t_i \in T$ (i = 1,2, …, m) is a multi-set on R, role-agent relations represented by $Re \subset R \times A$, then tasks are schedulable if and only if for any role vertex set $S \subseteq R$ in corresponding direct graph G, there has

$$\sum_{r \in S} d_{in}(r) \le |N(S)| \quad (2)$$

in which $d_{in}(r)$ represents the in-degree of role vertex r.

PROOF. Let N be the corresponding flow network model of an instance of the problem (see figure 4), and let $S \subset V$ be any vertex set which satisfy that source vertex $s \in S$ and sink vertex $t \in \overline{S}$, and set $S_T = S \cap T, S_R = S \cap R, S_A = S \cap A$.

It is simply to know, from the maximum flow minimum cut set theorem, that all tasks in T can be accomplished by the system if and only if for any cut of

© 2011 ACADEMY PUBLISHER

N, the capacity is larger than or equal to $\sum_{t_i \in T} d_{out}(t_i)$, represented by

$$capacity(S, \bar{S}) \geq \sum_{t_i \in T} d_{out}(t_i) \quad (3)$$

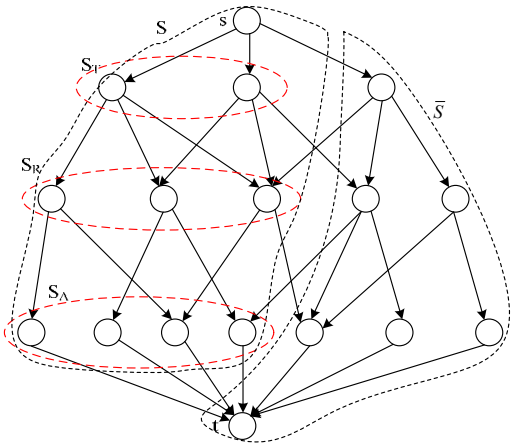

Figure4. The cut of the flow network *N*

We first show that if the formula (2) is satisfied, then tasks in T can be scheduled. Suppose that for any vertex set $S \subseteq R$ in Graph G satisfy the inequality $\sum_{r \in S} d_{in}(r) \leq |N(S)|$. Thus for any cut $(S, \bar{S})$ that:

$$c(S,\bar{S}) = \sum_{t_i \in T \setminus S} c(s,t_i) + \sum_{h \in (S_T, R \setminus S_R)} c(h) + \sum_{h \in (S_R, A \setminus S_A)} c(h) + \sum_{a_j \in S_A} c(a_j,t)$$

$$\geq \sum_{t_i \in T \setminus S_T} d_{out}(t_i) + \sum_{t_i \in S_T} (d_{out}(t_i) - \sum_{h \in (S_T, S_R)} c(h)) + |N(S_R) \cap \bar{S}| + |S_A|$$

$$\geq \sum_{t_i \in T \setminus S_T} d_{out}(t_i) + \sum_{t_i \in S_T} d_{out}(t_i) - \sum_{h \in (S_T, S_R)} c(h) + |N(S_R)|$$

(because $N(S_R) \subseteq S_A \cup (N(S_R) \cap \bar{S})$)

$$\geq \sum_{t_i \in T} d_{out}(t_i) - \sum_{h \in (S_T, S_R)} c(h) + \sum_{r \in S_R} d_{in}(r)$$

(by inequality (2))

$$\geq \sum_{t_i \in T} d_{out}(t_i) \text{ (because } \sum_{r \in S_R} d_{in}(r) \geq \sum_{h \in (S_T, S_R)} c(h) \text{)}$$

Thus, all tasks can be accomplished by MAS from the formula (3).

To prove the converse, suppose that all tasks in *T* can be accomplished. By the formula (3), for any cut $(S, \bar{S})$ of flow network *N*, there has $c(S, \bar{S}) \geq \Sigma_{t_i \in T} d_{out}(t_i)$, and if there a vertex set $S_R \subseteq R$, $S_T \subseteq$ T and $S_A \subseteq$ A which satisfy that $N(S_T)=S_R$, $S_A=N(S_R)$, let $S=S_T \cup S_R \cup S_A \cup \{s\}$, then $c(S, \bar{S}) = \Sigma_{t_i \in T \setminus S_T} d_{out}(t_i) + |S_A| = \Sigma_{t_i \in T \setminus S_T} d_{out}(t_i) + |N(S_R)|$. Thus $\Sigma_{t_i \in T \setminus S_T} d_{out}(t_i) + |N(S_R)| \geq \Sigma_{t_i \in T} d_{out}(t_i)$ which can get $|N(S_R)| \geq \Sigma_{t_i \in S_T} d_{out}(t_i)$. Observe that $N(S_T) = S_R$, we have $|N(S_R)| \geq \Sigma_{r \in S_R} d_{in}(r)$, which completes the proof.

IV. MAXIMUM TASK SCHEDULING PROBLEM IN MAS

We know that maximum flow algorithm can be used to solve task schedulable problem, but whether it is suitable for maximum scheduling problem? Unfortunately, we have prove that maximum scheduling problem is NP complete in our previous work [12].

As we know that MAX-S problem is NP complete, hence the problem cannot be solved in polynomial time through brute force algorithm. That is to say, the brute force method can not meet the urgency requirements of the emergency systems. To deal with the problem of task scheduling in above MAS model, we design three approximation algorithm to solve the maximum task scheduling problem. We will introduce Increased Greedy Algorithm, Decreased Greedy Algorithm and Revised Minimum Cost Algorithm in next three consecutive sections.

A. Increased Greedy Algorithm

The idea of increased greedy algorithm is as follows. In each iteration, the algorithm select the task that require least resources from the set *T*, and find appropriate agents to play roles of the task. If all roles of the task can find corresponding agent to play, the algorithm execute the task. Otherwise, the algorithm does not allocate any resource to it. The algorithm is ended with all task in *T* have been scanned by the algorithm.

Algorithm 1: IGA (Increased Greedy Algorithm)

While T≠∅, repeat the following process:

1. Search the task $t \in T$ that require least resource. For each role of the task:
   (a) If there has agents can play it, then select the most suitable agent.
   (b) Otherwise, delete the task from T, and skip to the top of the procedure.
2. If all roles of the task can find enough agent to play, then move the task from T to accomplished task set T', and delete all resources it has consumed.
3. When all tasks have been scanned, exit the while loop.

In each iteration, the IGA algorithm select the eligible task from T, and search suitable agents from A to play all role of the task. Obviously, the time complexity of the IGA algorithm is $O(nmk)$, in which *n* is the number of task, *m* is number of role, and *k* is number of agent.

B. Decreased Greedy Algorithm

The basic idea of DGA is as follows. The algorithm firstly construct the flow network in accordance with the instance of MAX-S problem, and use the maximum flow algorithm to test whether tasks can be scheduled. If all tasks can be accomplished, then the output is immediately gained. Otherwise, delete the most unimportant or largest resource consumption task from *T*. Repeat the above procedure until all tasks have been scanned, see details of the algorithm in algorithm 2.

Algorithm 2: DGA (Decreased Greedy Algorithm)

While T≠∅, repeat the following process:

1. Constructing flow network model $t \in T$ through Algorithm 1.
2. Test whether tasks can be scheduled.

© 2011 ACADEMY PUBLISHER

(a) If tasks can be scheduled, then output the solution and end the procedure.
(b) Otherwise, search the most unimportant or largest resource consumption task $t \in T$ and delete it from T.

The DGA algorithm call maximum flow network algorithm in each iteration, and the time complexity of maximum flow network algorithm is $O(n^3)$. It is simply to see that the time complexity of the DGA algorithm is $O(n(n+m+k)^3)$, in which n is the number of task, m is number of role, and k is number of agent.

C. Iterative Min-Cost Flow Algorithm

The IMCA algorithm firstly ordered tasks in accordance with the importance of tasks, such as $t_1 \prec t_2 \prec ... \prec t_n$. The order relation represents the priority of tasks when select task to execute, which means tasks must accomplished in accordance the order relation. Then the algorithm construct flow network with cost in accordance with above order relation. When assigning cost value to edges of the flow network, the task priority is higher, the cost of edge is smaller. Finally, the algorithm iteratively call the successive approximation algorithm[13, 14] on the flow network until the solution is gained. The details of IMCA is as algorithm 3.

Algorithm 3. Iterative Min-Cost Algorithm (IMCA)

Description: $t_1 \prec t_2 \prec ... \prec t_n$ represents the order relation of tasks，and cost value is interger.

1. Create directed graph $G = (V, E)$ in accordance with an instance of MAX-S problem, where the vertex set $V = T \cup R \cup A$ and the edge set $E = Tr \cup Re$, in which $Tr \subset T \times R$. and $Re \subset R \times A$.
2. Create the flow network model N as below
   (a) Add source node s and sink node t to graph G.
   (b) For each task node t_i (i = 1, 2, …, n) in T, create an edge (s, t_i) from s to this node with capacity equal to out-degree of t_i (e.g. $c(s, t_i) = d_{out}(t_i)$), and cost value is from 1 to n according to order relation.
   (c) For each agent node a_j (j = 1, 2, …, m) in A, create an edge from a_j to sink node t with capacity value is 1, and cost value is 0.
3. For each edge (r, a) in Re ($r \in R$, $a \in A$), set the capacity to be 1 and cost be 0.
4. Create an edge directly from t to s with unlimited capacity and zero cost.
5. While T≠∅, repeat the following process:
   (a) Call approximation algorithm to gain minimum cost of the flow network
   (b) If $|f| \geq \sum_{t_i \in T} capacity(s, t_i)$, then output the solution and exit the while loop.
   (c) Otherwise, scanning T in ascending order of costs. For task t_i, if $f(s, t_i) \geq capacity(s, t_i)$, then scanning the next task t_{i+1}. If $f(s, t_i) < capacity(s, t_i)$, then delete t_i from T ($T \leftarrow T - t_i$) and re-construct flow network model.

The time complexity of successive approximation algorithm is $O(v^3 \log(vC))$, in which C is maximum cost value and v is the vertex number of flow network N. From the above section, we know that cost value on edges (s, t_i) increased linearly, and IMCA algorithm will call n times successive approximation function in the worst case. Therefore, the time complexity of IMCA algorithm is $O(n(n+m+k)^3 \log(n(n+m+k)))$.

V. EXPERIMENTS

We implemented the increased greedy algorithm(IGA), the decreased greedy algorithm (DGA), and the iterative min-cost flow algorithm(IMCA) in C++, and tested them on a Windows PC. The purpose of these experiments is to study the performance of the algorithms in different problem settings using different instances of the problem. The performance measurements are the solution quality and computation time, where the solution quality (SQ) is computed as follows. When the number of tasks is small, we compare the output of the above algorithms with the optimal solution, but if it is not feasible to compute the optimal solution, we use the task completion rate to compare the solution quality. In the following, we describe the setup of all experiments, and present the results.

A. Eexperiment Settings

We consider several kinds of instance of the problem and several experimental environments. We now describe the different settings used in our experiments.

Setting 1. In order to be able to compare the output of the algorithms with the optimal solution, the scale of problem is relatively small. The number of tasks is 10, the number of roles is 10, and the number of agents varies from 5 to 40. During the experimenting, the average resource requirements remain unchanged, and the role-play ability of agent is maintained around 3.

Setting 2. The setting is used to evaluate the computation time of the algorithm implementing on large scale problems. The number of the vertex varies from 170 to 1290, and the number of tasks varies from 50 to 400, and the average outdegree of the task vertex and agent vertex remain around 4. The ratio between the number of task and agent is 1/2.

B. Experiment Results

The first experiment implements on experimental setting 1. We would like to see how the algorithms behave when the resource ratio varied gradually.

© 2011 ACADEMY PUBLISHER

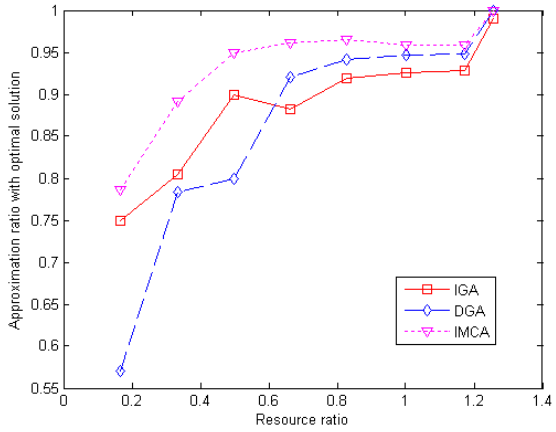

Figure 5. Approximation ratio of the algorithms

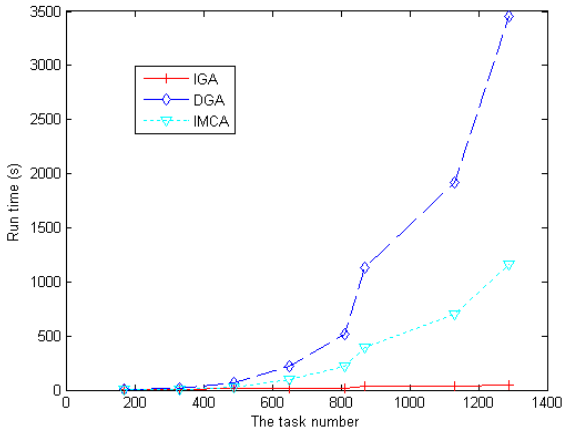

Figure4. Run time of the approximation algorithms

In Figure 5 we see how the quality of the IGA algorithm, DGA algorithm, and the IMCA algorithm depends on the resource ratio. Remarkably, for higher resource ratios, the quality of the algorithms are better. When the resource ratio grows above 1.2, the solution of the algorithms are almost optimal solution. The optimal ratio of all algorithms are larger than 0.55, and the solution quality of the IMCA is best, which larger than IGA and DGA algorithm. By experiments, we also find that when the resource ratio is relative lower, the solution of DGA is far worse than other algorithms. But when the resource ratio increased, the solution quality of the algorithms are more closer, and the solution quality of DGA even better than IGA when the resource ratio larger than 0.66. Besides, we find that when the role-play capability of agents are weak, the outputs of DGA are volatile.

The another experiment is to test the time performance of above algorithms. We would like to see the run time of algorithms when the scale of the MAX-S problem increased.

We can obviously see from the figure 6 that the run time of IGA is far less than DGA and IMCA algorithm. Therefore, the IGA algorithm is more suitable than DGA and IMCA algorithm when used in large scale task scheduling, especially when the MAS used in emergency systems that need strong real time requirements. Besides, we also find that the run time of IMCA is less than DGA which conflicts with the time complexity of the two algorithm mentioned in above section. But by further analysis, we find that the number of push operation of IMCA is much more less than DGA in each iteration, hence the practical run time of IMCA is less than DGA.

VI. COCLUSIONS

In this paper, we studied the task scheduling problem based on some new application of MAS such as emergency system, C3I commander system etc. We believe that it has a great amount of potential for these applications because of its advantages. Firstly, we built a multi-agent system model according to characteristics of above applications. Then we define and discuss task schedulable problem and maximum task scheduling problem based on above MAS model. For task schedulable problem, we firstly constructed a flow network corresponds to instance of task problem, and then test whether task can be scheduled on this flow network model. We conclude that the task schedulable problem can be solved in polynomial time. Thirdly, we proposed three efficiently approximation algorithm to solve maximum task scheduling problem. Finally, we do experiments to test both time performance and solution quality of above algorithms, the experiment results show that IGA algorithm is most suitable for MAX-S problem, especially when the MAS model used in emergency systems.

There are many interesting extensions to our current work. In this paper, we only test the performance of approximation algorithm by experiments. In our future work, we would also like to analyze the algorithm's optimality theoretically. Besides, we will discuss optimal task scheduling problem with utility, which aims to maximize the utility of the system. Another interesting topic for further work is the addition of QoS information among the agents. This may further help to select more suitable agents to improve efficiency of the system when scheduling.

ACKNOWLEDGMENT

This paper is supported by the National Science Foundation of China under Grant No. 60903130, and the Natural Science Foundation of the Jiangsu Province of China under Grant No. BK2007074, BK2009698, BK2009699.

REFERENCES

[1] O. Shehory and S. Kraus, "Methods for task allocation via agent coalition formation", Artificial Intelligence, Vol. 101, May, 1998, pp. 165-200

[2] H. L. Choi, L. Brunet, and J. P. How, "Consensus-based decentralized auctions for robust task allocation," IEEE Trans. on Robotics. 2008.

[3] D. Sarne1 and S. Kraus, "Solving the auction-based task allocation problem in an open environment", Proc of Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial

© 2011 ACADEMY PUBLISHER

Intelligence Conference, MIT Press, May, 2005, pp. 164-169.
[4] S. Aknine and O. Shehory, "A feasible and practical coalition formation mechanism leveraging compromise and task relationships", Proc of IEEE/WIC/ACM International Conference on Intelligent Agent Technology, IEEE Computer Society, Jul. 2006, pp. 436-439.
[5] M. Weerdt, Y. Q. Zhang and T. B. Klos, "Distributed task allocation in social networks", Proc of International Conference on Autonomous Agents and Multiagent Systems, ACM press, May, 2007, pp. 488-495.
[6] S. Berman, A. Halasz, M. A. Hsieh, V. Kumar, "Optimized Stochastic Policies for Task Allocation in Swarms of Robots**,** IEEE Transactions on Robotics", 2009, 25(4): 829-838.
[7] S. Abdallah and V. Lesser, "Modeling task allocation using a decision theoretic model", Proc of International Conference on Autonomous Agents and Multiagent Systems, ACM press, Jul. 2005, pp. 719-726.
[8] S. Papantonopoulos, G. Salvendy, "Analytic Cognitive Task Allocation: a decision model for cognitive task allocation", Theoretical Issues in Ergonomics Science, 2008 , 9(2): 155-185.
[9] E. Manisterski, E. David, S. Kraus and N. R. Jennings, "Forming efficient agent groups for completing complex tasks", Proc of International Conference on Autonomous Agents and Multiagent Systems, ACM press, May, 2006, pp. 257-264.
[10] H. Kitano, S. Tadokoro. RoboCup Rescue: a grand challenge for multi-agent systems. AI Magazine, 2001, 22(1): 39-52
[11] A. V. Goldberg, *É*va Tardos, R. E. Tarjan. "Network flow algorithms", In Bernhard Korte L*ászló* Lov*ász*, Hans J*ü*rgen Pr*ö*mel, and Alexander Schrijver, Springer-Verlag, 1990, pp. 101–164
[12] Z. Xiaowei. "Research on Task Scheduling problems for MAS based Emergency System", Master degree thesis, 2009.
[13] A. V. Goldberg, R. E. Tarjan. Finding minimum-cost circulations by successive approximation**,** Mathematics of Operations Research, 1990, 15(2): 430-466
[14] A. V. Goldberg, R. E. Tarjan **,** An efficient implementation of a scaling minimum-cost flow algorithm, Journal of Algorithms, 1997, 22(1): 1-29

Bin Li was born in Yangzhou, Jiangsu Province, China, in 1965. He received the Ph.D. degree in computer application technology from Nanjing University of Aeronautics & Astronautics, Jiangsu, China in 2001.

Currently, he is a Professor of Yangzhou University and conducts research in the areas of multi-agent system, artificial intelligence and service oriented computing.

Xiaowei Zhang was born in Yugan, Jiangxi Province, China, in 1982, M. S.. His main research interests include multi-agent system and artificial intelligence.

Jun Wu was born in Yangzhou, Jiangsu Province, China, in 1970. He received the Ph.D. degree in computer application technology from Southeast University, Jiangsu, China in 2005.

Currently, he is a Associate Professor of Yangzhou University and conducts research in the areas of computer network and formal method.

Junwu Zhu was born in Yangzhou, Jiangsu Province, China, in 1972. He received the Ph.D. degree in computer application technology from Nanjing University of Aeronautics & Astronautics, Jiangsu, China in 2008.

Currently, he is an Associate Professor of Yangzhou University and conducts research in the areas of Ontology and artificial intelligence.

© 2011 ACADEMY PUBLISHER