Multi Agent Based Dynamic Task Allocation

Arambam James Singh*, Poulami Dalapati, and Animesh Dutta

Multi Agent and Distributed Computing Lab, Department of Information Technology, National Institute of Technology Durgapur, India
{jamesastrick,dalapati89,animeshrec}@gmail.com

Abstract. Multi-Agent system, which is relatively a recent term, can be viewed as a sub-field of Distributed AI. In order to construct and solve complex real world problems like task allocation in an organization, a number of agents can perform work cooperatively and collaboratively. To achieve the maximum system utility through the proper task allocation among agents, there is a notion of dynamic team formation. In this paper we focus on task allocation mechanism to agents with dependencies among various tasks to agents either individually or by forming a team whenever needed according to their capability to do a particular task. We propose few algorithms which dynamically filter the capable agents for performing task and hence allocating them with the help of synergy value and their individual utility. We also simulate the task allocation mechanism done by our proposed algorithms and show the performance of the system by defining some metrics like, agent utility, team utility, system utility.

Keywords: Multi Agent System, Synergy, Dynamic Team Formation, Task Allocation.

1 Introduction

As we see in the human society, in order to organize an event, we need a group of individuals who perform the task associated with the event. For effective completion of this event, the compatibility among the individuals in the group is also important to consider. We extend this notion to an agent based system.

As the field of multi agent system is advancing, this event organization problem could be an emerging trend, in this regard. Here the event can be described as a goal that is to be achieved or performed with the help of an organization or rather an agency. An event may consists of several tasks which are either dependent or independent. So, event organization means successful completion of all the tasks in an efficient way. In this scenario, the allocation of tasks among agents is a major challenge as there are various agents with various capabilities which may or may not be matched with the required characteristics to complete the task. So, here comes the concept of team formation with compatible and capable agents. There is a common term *Synergy* [1] which is generally used in

* Corresponding author.

G. Jezic et al. (eds.), *Agent and Multi-Agent Systems: Technologies and Applications*, Advances in Intelligent Systems and Computing 296,
DOI: 10.1007/978-3-319-07650-8_18,

team formation in human society as well as in multi agent system. It means how well the group members in a team can work together.

Prior researches in agent system and task allocation [2] [3] have dealt with single agent's capability and mostly the task allocation through team formation [1] in undependable environment [4], i.e. all the tasks and subtasks are independent of each other, though there are some algorithms which supports dynamic task allocation scenario [5]. But here we have tried to introduce specifically the event organization through task allocation in the atomic level after decomposition of tasks into subtasks [6] to number of agents either individually or in a team, where some tasks are dependent on other, in terms of priority, as well as for efficient completion of tasks to maximize the overall system utility.

The structure of the paper is organized as follows: In section 2 we dicuss about some previous works in related domain, Section 3 present scope of our work. Section 4 & 5, model our proposed system where we define some metrics. Section 6 highlights some problems as well as its solution following the algorithms of these solutions. In section 7 we discuss about our experiments and results along with its practical application and finally section 8 concludes.

2 Related Work

In MAS(Multi Agent System) many prior works have been focused on the agents as homogeneous environment [7], taking into account that all the agents have same identical capabilities and there are no dependencies in the environment. Various algorithms are proposed on the basis of number of practical problems in this domain where all the agents are continuously adopting environmental inputs and accordingly forming teams to handle problems. But being in heterogeneous environment [2] agents have different capabilities and some of them have the goal to maximize system's utility where others are dedicated to their own work. There are various task allocation algorithm [4] [8] [9] [10] [3] which allocate tasks to agents in optimized way. Yichuan Jiang *et.al.* [4], in their paper, have given the solution of task allocation in undependable multi agent environment. They proposed the idea of deceptive agents who cannot provide desired resources and take more time in seeking those and also the idea of contractor agents. They have also provided a solution in this regard by giving the guarantee of access of dependable resources and also minimizing access time. Sherief Abdallah *et.al.* [8] come up with a generalized semi-MDP($SMDP$) model for efficient representation. They have referred a term *mediation* for decomposition of tasks and also for picking up the right agent negotiating with other agents. In adhoc system [11] all the capable agents collaborate with other agents to complete a task. With the increasing number of autonomous agents, the need of their effective interaction is also growing gradually. Agent has different perspective of the world. So, to achieve a common goal agents should have capability to adopt new team-mates and learn to cooperate dynamically as a part of the adhoc team [12] [13]. An important aspect of adhoc team formation is the idea of leading team-mates which influence a team to alter their decision in the notion

of maximizing system utility. In some recent paper, Katie Genter *et.al.* [12] have mentioned the idea of flocking agents where the main challenge is that whether more than one agent can lead the team to a desired objective and if yes, what would be the policy to do that efficiently as there will be no explicit control over the behavior of the flocking agents, where influence over the adhoc agents is implicit. Agents compute their best action based on their most recent observations of their team-mates' actions. The forming of team [1] highly depends on the composition or the combination of the team members whether they are compatible enough to work in a group to achieve a goal or not. And for checking compatibility between them a team synergy is used in multi agent system. The synergy of two agents can be formulated and calculated from synergy graph where distance between them signifies how well they can work together in a team.

3 Scope of the Work

The work done so far this domain is concerned with tasks assignment problems i.e. assigning a set of tasks to a set of agents in MAS. But there are very few remarkable mechanism, which deal with the task allocation dynamically considering some disastrous condition [14] [15]. There is another issue like optimized allocation of tasks here [8] [9]. An automated agent based system should address all these practical phenomena in an efficient way. Here we address such issues and also take some practical scenario to implement the same. In our proposed technique, the agency perform tasks either by forming team or by decomposing tasks into atomic subtasks according to the need. We also simulate our work to achieve better performance than previous works.

4 System Model

The task of event organization can be formally modeled by some parameters. An event can be described as a goal that is to be achieved and the agency should perform the event in such a manner that the allocation of tasks to the agency will be the optimal allocation to maximize the total system utility. For the sake of clarity here the whole system can be defined by two attributes,

$$Sys = <\text{E,A}>$$

where E is the *event* which can be represented as a task graph [16] and A is the agency which will perform the tasks to complete the event.
An *agency* is composed of number of agents as,

$$A = \{A_i | i \in [1, n]\}$$

Here we define some terms which we use in modeling the task allocation problem.

- *Attributes* : Attributes are some properties which should be available in the characteristics set of tasks and capability set of agents so that tasks can be allocated to agents or in the other way agents can perform some task successfully.
  $C = \{C_i | i \in [1, k]\}$

- *Characteristics or Attributes of a task* : Characteristics of a task T_i which is denoted by char(T_i) represents some attributes of the individual task which are required in agents' capability set in order to complete the task.
  $char(T_i) \subseteq C$
- *Capability of an agent* : Capability of an gent A_i is the set of skills or attributes an agent must have to perform tasks and is denoted by cap(A_i).
  $cap(A_i) \subseteq C$
- *Priority of a task* : Priority of any task can be described in terms of dependencies i.e if T_j is dependent on T_i then T_i needs to be executed first. So T_i has the higher priority than T_j. In our scenario we have represented priority as
  $P = \{P_i | i \in [1, l]\}$
  Please note that more than one task can have same priority and these tasks can be executed concurrently provided system have enough agents.
  A task T_i can only be assigned to an agent A_i iff A_i has the capability to perform the task T_i provided A_i is available at that time instant. i.e. $char(\ T_i\ ) \cap cap(\ A_i\ ) = char(\ T_i\ )$. If we are not able to find a single agent who possesses all the attributes required to complete the task, then the task T_i can either be decomposed into further subtasks as,
  $T_i = \{\ T_{ij} | j \in [1, k]\}$
  or, T_i can be solved by forming a team of agents whose collective capability set matches to the required attributes set of T_i, i.e $char(\ T_i\ ) = \forall i,\ A_i \in T_m \cup cap(\ A_i\ )$, where T_m means team of agents.
- *Synergy between agents* : In every agent community, there is a synergy graph associated with it. Synergy graph provides the relationship among agents. Mathematically, A synergy graph S_{syn} is a connected graph, $S_{syn}=(V_s, E_s)$,
  $V_s = \{\ A_i : A_i \in \text{A}\ \}$
  $E_s = \{\ E_{ij}$, where E_{ij} represents the compatibility between A_i and $A_j\ \}$
  $S_{syn}(A_i, A_j) = 1\ /\ d(A_i, A_j)$, $i \neq j$ and $\forall$i,j$\in [1, n]$
  $d(A_i, A_j)$= Distance or Compatibility between any two agents A_i, A_j in S_{syn}.
  For example, In Fig. 1 we have a synergy graph.
  $S_{syn}(A_1, A_2) = 1\ /\ d(A_1, A_2) = 1/1 = 1$
  $S_{syn}(A_1, A_3) = 1\ /\ d(A_1, A_3) = 1/2 = 0.5$
  The synergy values among agents may change overtime as in when the synergy network or graph change.

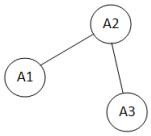

Fig. 1. Synergy (Source: [1], Sec 3.1)

5 Metric Definition

Here we propose some of the metrics to analyse the whole system of task allocation to agents.

- *Agent Utility* : Utility of an agent A_j is defined w.r.t a task T_i , i.e. how well an agent can execute that task, which will depend on how much capability the agent has for that task. So mathematically can be defined as :

$$U(T_i, A_j) = \frac{|char(T_i) \cap cap(A_i)|}{|cap(A_i)|}$$

- *Team Utility* : Utility of a team can be defined as, how Effectively members of the team, perform the task in a cooperative manner.
  Mathematically,
  U_t= U(T_m)+S_{syn} (T_m)
  where,
  $U(T_m)$=$\Sigma_{\forall j, j \in [1,n]}$ $U(T_i, A_j)$,
  $S_{syn}(T_m)$=$\Sigma_{\forall i,j,A_i,A_j \in T_m}$ $Syn(A_i, A_j)$, $T_i \in E$ and $A_i \in A$
- *System Efficiency* : In our scenario efficiency of the system can be described from various perspectives as, we allocate a task to a single agent if the agent has all the necessary capabilities, we also allocate task to a team of agents if we are not able to find a single agent which has all the necessary capabilities. Now here, even after using various techniques to allocate tasks, one question remains as to how much we are able to execute those tasks which can be executed concurrently i.e tasks which are in same priority. So, if our system is able to execute all the concurrent tasks concurrently then it means we have no task in waiting state and thus it should account for the efficiency of the system.
  Efficiency can be mathematically represented as: $Eff = \sum_{\forall i, P_i \in P} (\frac{\lambda}{|same-prio|})$
  where,
  $same - prio$: Set of tasks with same priority,
  λ : Total Number of Concurrently Executed Tasks in same-prio

6 Problem Formulation

In order to decide which task should be allocated first and which agent should do the task so that the maximum system utility or the efficiency will be achieved, we introduce the notion of priority value P for each task, i.e. all the task T_i has a priority value from the set $P = \{ P_i \mid i \in [1,l] \}$, where $P_i > P_j$ if i<j, means task T_i with priority value P_j can only be allocated to an agent $A_j \in$ A iff task T_i with priority value P_i is completed, i.e. T_j dependent on T_i or T_i has the higher priority to complete the event efficiently.

Problem 1: *If more than one task have same priority value, then there may arise an ambiguity that in which order tasks should be performed.*
When number of tasks have the same priority value, i.e.

$P(\ T_i\ ) = P_i$, $P(\ T_j\ ) = P_j$ and $P_i = P_j$

where $P(\ T_i\ )$ = the priority value of the task T_i, $T_i \in$ E, then agent can pick any one of them as the tasks have no dependencies. But agent should calculate the utility value for the order of picking a single task so that the total utility of the system be the maximum. And if there are sufficient number of capable agents are available for doing all those tasks, then the tasks can be done parallely.
i.e. for any task T_i and T_j, (i $\neq$ j), if $P_i = P_j$, then those tasks can be done concurrently provided,

$char(T_i) \cap cap(A_i) = char(T_i)$ and $char(T_j) \cap cap(A_j) = char(T_j)$

Problem 2: *If there are number of tasks T_i with certain characteristics and number of available agents with all the capability to perform a task or with partial capability, then which agent(s) should do which task so that the system utility get maximized.*
There are two possible cases in this scenario.
Case 1 : When there are more than one agents with same capability set, then any agent can perform the task. But only one agent will do the task.
$char(\ T_i\ ) \cap cap(\ A_i\ ) = char(\ T_i\ )$ and $char(\ T_i\ ) \cap cap(\ A_j\ ) = char(\ T_i\ )$
where $T_i \in$ E and A_i,$A_j \in$ A.

To maximize the total system efficiency every agent will calculate their individual utility value to perform a particular task. Among all the capable agents who has the maximum utility to complete the task, will perform that one.

$$allocate(T_i, A_i) = 1, \text{ iff } (T_i, A_i) = max\{U(T_i, A_i)\}$$

Case 2 : Among all the available agents no single agent has all the required capabilities or attributes to perform any particular task completely, i.e. $char(T_i) \cap cap(A_i) \neq char(T_i)$, where $T_i \in E$ and $A_i \in A$. So, in this case either the tasks should be decomposed into number of subtasks T_{ij}.

$$T_i = \{\ T_{ij}\ |j \in [1, k]\ \}$$

or some teams of agents are to be formed in order to complete the whole task.

$$\forall i,\ A_i \in T_m \cup cap(\ A_i\ ),$$

where T_m means team of agents.

Now, whether to form any team or to divide a task further into subtasks can be decided using the synergy graph S_{syn} of the available agents. Here, to take the decision of team formation, we have selected a threshold value for the synergy T_h between agents. If for any two agents in the synergy graph, $\{S_{syn}(A_i, A_j) \forall i, j \in [1, n]\} < T_h$, then the agents are not compatible to work together in a team. In this situation decomposition of task T_i into subtasks T_{ij} is must. After decomposition if the agent A_i has the full capability to perform T_{ij} and also it has the maximum utility value then directly allocate the task to that agent.
$(char(T_{ij}) = cap(A_i)) \bigwedge (U(T_{ij}, A_i) = max\{U(T_i, A_i)\}) = TRUE$,
then $allocate(T_{ij}, A_i) = 1$

Again, if the attributes required to complete any particular task match to the united capability set of number of available agents and the calculated synergy value between agents are greater or equal to the threshold value, then they are compatible to work together and hence to complete any particular task a team of those agents can be formed without any decomposition of task T_i.
$(char(T_i) = (cap(A_i)) \cup cap(A_j)))\forall i, j \in [1, n]) \bigwedge$
$(\{S_{syn}(A_i, A_j)\forall i, j \in [1, n]\} > T_h) = TRUE$,
then, $T_m = \{A_i, A_j\}, \forall A_i, A_j \in A$ and $allocate(T_i, T_m) = 1$

Algorithm 1. Selects Agents with all the characteristics of a tasks, kept in full[] and also select agents with atleast one characteristic and kept in partial[]

full[] */Array containing agents with all the capabilities of any task*
partial[] */Array containing agents with partial capability*

```
1: k=0, l=0
2: for all tasks T_i, T_i ∈ E do
3:    for all agents A_j, A_j ∈ A do
4:       if char( T_i ) ∩ cap( A_j ) == char( T_i ) then
5:          T_i.full[k++] ← A_j
6:       end if
7:       if char(T_i) ∩ cap( A_j ) ⊂ char( T_i ) then
8:          T_i.partial[l++] ← A_j
9:       end if
10:   end for
11: end for
```

7 Experiments and Results

In our system we use three metrics Agent Utility, Team Utility and System Efficiency. System performance is measured using the expression of System efficiency given in section 5. Efficiency is the ratio of total number of concurrently executed task out of total number of task present in same-prio. Now in order for a task to execute we need to find the agent or team of agents. Finding agent or team of agents is done using the metrics Agent Utility and Team Utility. So, ultimately all the three metrics are accounted for System Performance.

Now to simulate our proposed methodology to organize an event, consisting of tasks, i.e. to efficiently allocate the tasks to an appropriate agent or to a team of agents and check how effectively our system responds, we consider an event E which may consists of various number of tasks. We run our algorithm for different number of tasks an event can have also for different priorities(3,6,9) of the task set. Fig. 2. represents the Efficiency Graph 1. Here we fix number of agents in the agency to 8 and test the efficiency for number of tasks varying from 5 to 20. We found an effective result as we can see from the Efficiency graph 1 the average efficiency is 85.13%.

Algorithm 2. Filters the partial[] column of each task, i.e find the best combination or a team of agents which has maximum synergy value and also their collective capabilities are sufficient enough to perform the task

```
1: for i = 1 to | E | do
2:    Evaluate(T_i.partial[])
3: end for
1: Evaluate(T_i.partial[])
2: Group = φ, U_max = 0
3: for j = 1 to T_i.partial.length do
4:    for k = j to T_i.partial.length do
5:       if char(T_i) == char(Group) ∪ cap( T_i.partial[k]) then
6:          Group ← T_i.partial[k]
7:          U ← 1/|Group| Σ_{∀a_i,a_j∈Group∧i≠j} Sy(a_i, a_j)
8:          break
9:       end if
10:      if cap(T_i.partial[k])∉(cap(Group)∩ char(T_i)) then
11:         Group ← T_i.partial[k]
12:      end if
13:   end for
14: end for
```

In second experiment, we fix the number of tasks of the event E to 10 and run our algorithm for different number of agents in the agency A varying from 5 to 19, also for different priorities(3,6,9) of tasks set. Fig. 3. represents Efficiency Graph 2. Here also we obtained a positive result with average efficiency 83.68%.

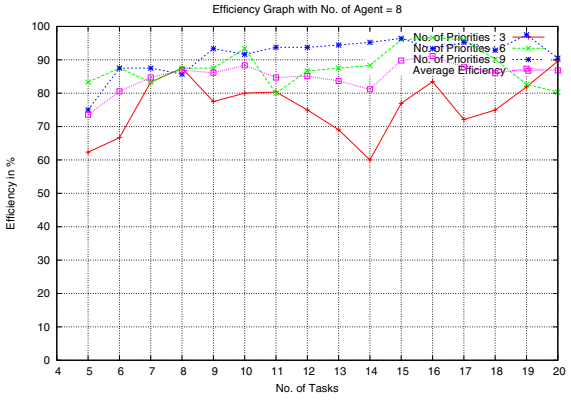

Fig. 2. Efficiency Graph 1 (Source: Experiments and Results)

Algorithm 3. Allocates tasks to a agent or to a team of agents based on priority and then tasks are also executed

if $A_i \notin T_k$.partial[],$\forall T_k \in$ E $\wedge$ k $\neq$ j **then**
  Allocate(T_j, A_i)
end if
if $A_i \in T_k$.partial[] and $A_i \in T_i.full[]$,$\forall T_k$,$T_j \in$ E $\wedge$ k $\neq$ j **then**
  Allocate(T_k, A_i)
end if
for all task $T_i \in E$, T_i.full[] $\neq \phi$ and T_i.partial $\neq \phi$ **do**
  if Utility($T_i.Group$) $< U_{threshold}$ **then**
    decompose(T_i)
  else
    form-team(T_i)
  end if
end for
for all task $T_k \in E$, T_k.full[] $\neq \phi$ **do**
  for all agent $A_i \in$ A $\wedge$ $A_i \in T_i.full[]$ **do**
    $A_i.Utility = |char(T_k) \cap cap(A_i)|$ / $|cap(A_i)|$
  end for
end for
for all priorities $P_i \in$ P, i=1 to P.length **do**
  Same-Prio $\leftarrow$ Task(P_i) {//$Task(P_i)$ *returns tasks with same priority* P_i}
  Exclusive-full = ϕ ; Flag-full = True;
  for all tasks T_i, $T_i \in Same - Prio$ **do**
    if Exclusive-full $\cap$ max($A_i.Utility$, $A_i \in T_i.full[]$) == ϕ **then**
      Exclusive-full $\leftarrow$ max($A_i.Utility$, $A_i \in T_i.full[]$)
    else
      Flag-full $\leftarrow$ False
      Break
    end if
  end for
  if Flag-full == True **then**
    for all tasks T_i, $T_i \in Same - Prio$ **do**
      Allocate(T_i,max($A_i.Utility$, $A_i \in T_i.full[]$))
      Execute(T_i)
    end for
  end if
  Exclusive-partial = ϕ
  Flag-partial = True
  for all tasks T_i, $T_i \in Same - Prio$ **do**
    if Exclusive-partial $\cap$ $T_i.partial[]$) == ϕ **then**
      Exclusive-partial $\leftarrow$ $T_i.partial[]$)
    else
      Flag-partial $\leftarrow$ False; Break
    end if
  end for
  if Flag-full == False *and* Flag-partial == True **then**
    for all tasks T_i, $T_i \in Same - Prio$ **do**
      Allocate(T_i,$T_i.Group$); Execute(T_i);
    end for
  end if
  if Flag-full == False *and* Flag-partial == False **then**
    for all tasks, $T_i \in Same - Prio$ **do**
      if $A_k \in \cap_{\forall i, T_i \in Same-Prio} T_i$.full[] and $A_l \notin T_j$.full[],$\forall T_j \in Same - Prio$ and $cap(A_k) \cap cap(A_l) == cap(A_k)$ **then**
        Replace(A_k, A_l)
      end if
      if $A_k \in \cap_{\forall i, T_i \in Same-Prio} T_i$.partial[] and $A_l \notin T_j$.partial[],$\forall T_j \in Same - Prio$ and $cap(A_k) \cap cap(A_l) == cap(A_k)$ **then**
        Replace(A_k, A_l)
      end if
      Execute(T_i)
    end for
  end if
end for

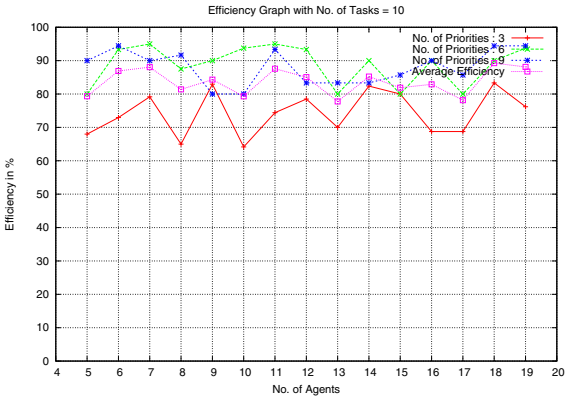

Fig. 3. Efficiency Graph 2 (Source: Experiments and Results)

7.1 Practical Application to RoboCup Rescue

We have tested our system for a practical scenario of disaster rescue, the scenario chosen is that of a Robocup Rescue Problem [14] [15].

In a city after a disaster like an earthquake suffers from devastation. There are chaos everywhere, road blocks, Fires, people stucked in bulidings, seriously injured, urgent need of medical attentions, failure of communication infrastructures.

Now in order to save the civilians the rescue teams need to perform various tasks. The rescue agents that we have considered are : Ambulance Agents, Police Agents and Fire-Fighter Agents. Each of these agents are different, they perform different tasks. But there are few tasks that any of those agents can also perform.

Followings are the tasks that we have considered for our system :

- Task 1 : Providing First Aid to Injured civilians and taking them to near by hospitals
  Agent Required : Ambulance Agents

- Task 2 : To extinguish fire if any
  Agent Required : Fire Fighter Agent or Police Agents

- Task 3 : To save stucked civilians from a multi storey buildings
  Agent Required : Fire Fighter Agent or Police Agent

- Task 4 : To evacuate civilians from the affected areas
  Agent Required : Police Agent or Fire Fighters Agent

- Task 5 : To clear the blocked roads
  Agent Required : Police Agent

– Task 6 : Coordinating civilians
  Agent Required : Police Agent

Our aim here is to rescue all the civilians, which means completion of all the 6 tasks. This further raises the question of task allocation among the agents. Here we want all the tasks to be performed in an efficient manner. A task can be performed by any agent if it has all the necessary capabilities or characteristics. Here it is obvious that though the instances of tasks are predefined, the total number of tasks are much more than the number of available agents in practical. The priority and dependencies between those works change over time. As an example, say at a point of time rescue team rescue an injured people and decide to send them by ambulance to the hospitals. In the mean time another injured person comes into notice whose condition is worse. So, we have to send this people to the hospitals more urgently than the previous rescued people. And as per the dependencies are concerned, such scenario may happen that untill the road blockage is removed, the ambulance cannot move forward.

We calculate the efficiency of the system for this scenario using the system utility formula given in Section 5 and we got an efficiency of 73.25%.

8 Conclusions

In this paper we present three algorithms for dynamic task allocation, where all the tasks have different priorities which determine the execution order of the tasks. We able to achieve approximately 85.13% of efficiency by varying number of tasks while number of agents are fixed. Again, solution finds 83.68% of efficiency when number of agents in agency are varying.

Here we only propose the mechanism of an agent based dynamic task allocation with interdependencies and priorities among tasks. But there can be another important issue of *fault tolerance*, i.e. when any number of capable agents for doing any particular task fail. So, the future direction of our work is to extend this mechanism by developing another dynamic algorithm which takes into account this practical issue, i.e. how other agents can manage to do the task in absence of its previously allocated capable agents.

References

1. Liemhetcharat, S., Veloso, M.: Modeling and Learning Synergy for Team Formation with Heterogeneous Agents. In: Conitzer, Winikoff, Padgham, van der Hoek (eds.) Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012), Valencia, Spain (2012)
2. Anders, G., Hinrichs, C., Siefert, F., Behrmann, P., Reif, W., Sonnenschein, M.: On the Influence of Inter-Agent Variation on Multi-Agent Algorithms Solving a Dynamic Task Allocation Problem under Uncertainty. In: IEEE Sixth International Conference on Self-Adaptive and Self-Organizing Systems, pp. 29–38 (2012)

3. Shehory, O., Kraus, S.: Methods for task allocation via agent coalition formation. Artificial Intelligence 101, 165–200 (1998)
4. Jiang, Y., Zhouand, Y., Wang, W.: Task Allocation for Undependable Multiagent Systems in Social Networks. Advan. IEEE Transaction on Parallel and Distributed Systems 24(8), 1671–1681 (2013)
5. Macarthur, K.S., Stranders, R., Ramchurn, S.D., Jennings, N.R.: A Distributed Anytime Algorithm for Dynamic Task Allocation in Multi-Agent Systems. In: Association for the Advancement of Artificial Intelligence (2011)
6. Skowron, A., Nguyen, H.S.: Task Decomposition Problem in Multi-Agent System. Artificial Intelligence 101
7. Albrecht, S.V., Ramamoorthy, S.: Comparative Evaluation of MAL Algorithms in a Diverse Set of Ad Hoc Team Problems. In: Conitzer, Winikoff, Padgham, van der Hoek (eds.) Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012), Valencia, Spain (2012)
8. Abdallah, S., Lesser, V.: Modeling Task Allocation Using a Decision Theoretic Model. National Science Foundation Engineering Research Centers Program
9. Weerdt, M., Zhang, Y., Klos, T.: Multiagent task allocation in social networks. Autonomous Agent Multi-Agent System 25, 46–86 (2012), doi:10.1007/s10458-011-9168-3.
10. Jiang, L., Zhan, R.: An Autonomous Task Allocation for Multi-robot System. Journal of Computational Information Systems 7: 11 25, 3747–3753 (2011)
11. Barrett, S., Stone, P.: An Analysis Framework for Ad Hoc Teamwork Tasks. In: Conitzer, Winikoff, Padgham, van der Hoek (eds.) Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012), Valencia, Spain (2012)
12. Genter, K., Agmon, N., Ston, P.: Ad Hoc Teamwork for Leading a Floc. In: Ito, Jonker, Gini, Shehory (eds.) Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013). Saint Paul, Minnesota (2013)
13. Agmon, N., Stone, P.: Leading Ad Hoc Agents in Joint Action Settings with Multiple Teammates. In: Conitzer, Winikoff, Padgham, van der Hoek (eds.) Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012), Valencia, Spain (2012)
14. Santos, F.D., Bazzan, A.: Towards efficient multiagent task allocation in the RoboCup Rescue: a biologically-inspired approach, pp. 465–486. Springer (2010)
15. Chapman, A.C., Micillo, R.A., Kota, R., Jennings, N.R.: Decentralised Dynamic Task Allocation:A Practical Game Theoretic Approach. In: Decker, Sichman, Sierra, Castelfranchi (eds.) Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009), Budapest, Hungary (2009)
16. Upadhyay, P., Acharya, S., Dutta, A.: Task Petri Nets for Agent based computing. INFOCOMP Journal of Computer Science (2013)