2013

# Decentralised task allocation under space, time and communication constraints for disaster domain

Xing Su
*University of Wollongong*, xs702@uowmail.edu.au

Minjie Zhang
*University of Wollongong*, minjie@uow.edu.au

Quan Bai
*Auckland University of Technology*, quan@uow.edu.au

## Publication Details

Su, X., Zhang, M. & Bai, Q. (2013). Decentralised task allocation under space, time and communication constraints for disaster domain. In Q. Bai, T. Ito, M. Zhang, F. Ren & X. Tang (Eds.), The International Workshop on Smart Simulation and Modelling for Complex Systems (SSMCS 2013) (pp. 25-37). China: IJCAI.

# Decentralised task allocation under space, time and communication constraints for disaster domain

**Abstract**
The coordination of dynamic task allocation based on available resources is very challenging in disaster domains under time, space and communication constraints. In addition, it is also very hard or even impossible to achieve tasks allocation by using a centralised manner with the global knowledge of the working environment. This paper presents a novel decentralised coordination approach for dynamic task allocation by considering space, time and communication constraints in a disaster domain, and workloads and priorities of different tasks. In this approach, a group formation mechanism is proposed to help agents with limited communication range to achieve effective task allocation in a group through cooperation. The overall task allocation is achieved through distributed coordination in each dynamic group without a central control mechanism to reflect real life situations in the disaster domain. The experiment results show that the proposed approach outperforms other decentralised approaches, in disaster domains under space, time and communication constrains.

**Keywords**
constraints, disaster, domain, allocation, under, space, decentralised, time, task, communication

**Disciplines**
Engineering | Science and Technology Studies

**Publication Details**
Su, X., Zhang, M. & Bai, Q. (2013). Decentralised task allocation under space, time and communication constraints for disaster domain. In Q. Bai, T. Ito, M. Zhang, F. Ren & X. Tang (Eds.), The International Workshop on Smart Simulation and Modelling for Complex Systems (SSMCS 2013) (pp. 25-37). China: IJCAI.

# Decentralised Task Allocation under Space, Time and Communication Constraints for Disaster Domain

Xing Su[1], Minjie Zhang[1], and Quan Bai[2]

[1] University of Wollongong, Australia
xs702@uowmail.edu.au, minjie@uow.edu.au
[2] Auckland University of Technology, New Zealand
quan.bai@aut.ac.nz

**Abstract.** The coordination of dynamic task allocation based on available resources is very challenging in disaster domains under time, space and communication constraints. In addition, it is also very hard or even impossible to achieve tasks allocation by using a centralised manner with the global knowledge of the working environment. This paper presents a novel decentralised coordination approach for dynamic task allocation by considering space, time and communication constraints in a disaster domain, and workloads and priorities of different tasks. In this approach, a group formation mechanism is proposed to help agents with limited communication range to achieve effective task allocation in a group through cooperation. The overall task allocation is achieved through distributed coordination in each dynamic group without a central control mechanism to reflect real life situations in the disaster domain. The experiment results show that the proposed approach outperforms other decentralised approaches, in disaster domains under space, time and communication constrains.

## 1 Introduction

Nowadays, agent-based coordination for task allocation has been widely applied in many domains such as disaster rescue, space exploration and distributed computing, etc [7], [13], [15]. The main objective of task allocation is to allocate the most suitable resources (usually agents) to different tasks in a rational way. In addition to the main objective, there are different new requirements and constraints, which are requested for the efficient coordination of task allocation in different application domains. This paper focuses task allocation problems in disaster domains.

Normally, a disaster domain has the following major constraints for task allocation. **1) Space constraints**. In a disaster domain, tasks are discovered at different locations. Hence, agents need to move to the location of tasks, which will consume time. **2) Time constraints**. In a disaster domain, the objectives of task allocation include locating and saving survivors in debris, extinguishing a fire before it spreads, etc. Under such circumstances, each task has a deadline and the task should be done before the deadline (i.e. when the survivor is alive or the building is still stand). **3) Communication constraints**. Due to the destruction of local infrastructures and other conditions in a disaster area, rescue agents have difficulties to communicate with each other. **4) Task Urgency**. Different tasks have different degrees of urgencies. The most urgent tasks need to be finished before the deadlines, and the least urgent tasks can be disregarded during task allocation if resources are not sufficient. In many existing approaches [12], [6], the researchers emphasised on finishing tasks as many as possible before the deadline, but did not consider the degree of urgency of tasks. For example, supposing there are two collapsed buildings, one with 10 survivors and the other with 10 tons of goods stored in it. When allocating agents for these two buildings, saving the 10 survivors would take precedence over rescuing 10 tons of goods if the available resources were not sufficient to do both. The degree of urgency of each task is obviously a key issue that should be considered.

To handle the above constraints and issues, various models, mechanisms and approaches have been proposed to assist coordinating task allocation in a disaster domain each having its own approach [9], [14], [8], [3]. A number of centralised approaches [6], [12] have been developed to coordinate task allocation in a diaster domain. The centralised approaches can ensure an optimal allocation solution if a central controller can have global knowledge of overall tasks and agents in the domain. However, in most disaster domains, it is hard for a central controller to have that kind of knowledge due to the dynamics of the agents and tasks, as well as time and communication constraints.

To overcome the limitations of centralised approaches, some decentralised approaches [4], [2] have been developed to deal with the coordination of task allocation in a diaster domain. One of the famous models is the F-max-sum algorithm proposed by Ramchurn et al. [11]. The F-max-sum algorithm requires message passing between agents to achieve task allocation. If communication quality and range are limited, it is hard for agents to get enough messages from each other to get accurate results so the F-max-sum algorithm cannot work well. In addition, the F-max-sum algorithm does not consider the different urgencies of tasks. Furthermore, in most of the current research, the communication constraints only refer to the number of messages passing between agents and few of them consider the communication range of agents.

In order to meet the challenges of task allocation in disaster domains, a novel decentralised coordination approach for dynamic task allocation is proposed in this paper. Our approach employs a token passing mechanism to assist neighbouring agents to communicate with each other under communication constraints and to form groups in a decentralised manner. The contributions of the proposed approach are that (1) our approach considers space, time and communication constraints to reflect the real situation in a disaster domain; (2) our approach considers the degree of urgency and the workload of each task as well as the dynamic nature of a disaster environment so as to meet the conditions of the environment; (3) in our approach, a novel group formation mechanism is developed to help agent to form groups despite communication limits; and (4) a comprehensive utility function is designed to help the coordinator to find the best task allocation for the group.

The rest of paper is organized as follows. The problem is formulated and definitions are given in Section 2. The basic procedure of our approach is introduced in Section 3. The experiment and analysis are given in Section 4. Some related work about task allocation is introduced and analyzed in Section 5. The paper is concluded in Section 6.

## 2 Problem Description and Definition

In general, agent-based task allocation models the problem as there is a set of tasks and a set of agents. A task set contains $m$ tasks, which can be described as $\{T_1, T_2, T_3, ...,$

$T_m\}$, where $T_i$ represents the $i^{th}$ task and $1 \leq i \leq m$. An agent set contains $n$ agents, which can be denoted as $\{A_1$, $A_2$, $A_3$, ..., $A_n\}$, where $A_j$ represents the $j^{th}$ task and $1 \leq j \leq n$.

In our approach, we give the following definitions to describe the problem in details.
**Definition 1:** A *Task* ($T_i$) can be defined as a six-tuple $T_i$=<$TNo$, $DLine_i$, $WLoad_i$, $Loc_i$, $Emg_i$, $TSta_i$ >, where $TNo$ is the ID of task $T_i$, $DLine_i$ is the deadline of $T_i$ (e.g. the time point until which the survivor can remain alive or a building remain standing), where $Dline_i \in [0, \infty]$, $WLoad_i$ is the workload of $T_i$, which represents how many work units must be done in order for $T_i$ to be completed, $Loc_i$ is the location of $T_i$, $Emg_i$ is the degree of urgency of $T_i$, where $Emg_i \in [1, 10]$, in which 1 and 10 represent the lowest and the highest degrees of urgency, respectively, and $TSta_i$ is the status of $T_i$, as either 'available', 'working', 'finished' or 'expired'.
**Definition 2:** An *Agent* ($A_j$) can be defined as a six-tuple $A_j$=<$ANo$, $Uti_j$, $Loc_j$, $MSp_j$, $Comm_j$, $ASta_j$>, where $ANo$ is the ID of $A_j$, $Uti_j$ is the working efficiency of $A_j$. $Uti_j$ represents how many units of workload that $A_j$ can perform per time unit. $Loc_j$ is the current location of $A_j$, $MSp_j$ represents the moving speed of $A_j$, which represents how far $A_j$ can move per time unit, $Comm_j$ is the communication and scan range of $A_j$, and $ASta_j$ is the status of $A_j$, which can be ether 'available' or 'working'.

The main objective of weighted task allocation is to find an arrangement of agents to tasks to maximize the weight sum of total finished tasks, which is described as:

$$Objective = argmax \sum_{i=1}^{m} Finish(T_i) \times WLoad_i \times Emg_i, \tag{1}$$

where $Finish(T_i)$ is a Boolean-valued function, if $T_i$ is finished, $Finish(T_i) = 1$, otherwise $Finish(T_i) = 0$.

Apart from the main objective, the cost of agents for finishing tasks should be minimized and this includes 1) minimizing the traveling time and distance of agents for finishing tasks and 2) minimizing the working time for agents to finish tasks.

As mentioned before, we employed a token passing mechanism to help agent efficiently pass information about tasks and agents. There are two types of tokens in our approach: task tokens ($TToken_k$) and agent tokens ($AToken_j$), both of which are created by agents. The two kinds of tokens are defined by Definition 3 and Definition 4, respectively.
**Definition 3:** A *Task Token* ($TToken_k$) can be defined as a six-tuple, $TToken_k$=< $ANo$, $TNo$, $DLine_k$, $WLoad_k$, $Loc_k$, $Emg_k$>, where $ANo$ is the ID of the agent, which creates the $TToken_k$, $TNo$ is the ID of the task represented by $TToken_k$, $DLine_k$ is the deadline of the task, represented by $TToken_k$, $WLoad_k$ is the workload of the task represented by $TToken_k$, $Loc_i$ is the location of the task represented by $TToken_k$ and $Emg_k$ is the degree of the urgency of the task represented by $TToken_k$.
**Definition 4:** An *Agent Token* ($AToken_j$) can be defined as a four-tuple $AToken_j$=< $ANo$, $Uti_j$, $Loc_j$, $MSp_j$>, where $ANo$ is the ID of the agent represented by $AToken_j$, $Uti_j$ is the working efficiency of $A_j$, $Loc_j$ is the current location of $A_j$ and $MSp_j$ represents the moving speed of $A_j$.

An agent $A_j$ can have two different roles, a $coordinator$ or/and a $resource\ provider$ (an agent can be a coordinator, a resource provider or both).
**Definition 5:** A *Coordinator* is an agent, which is in charge of allocating agents to tasks.

**Definition 6:** A *Resource Provider* is an agent, which has the requested resource requested by a task and can finish the task before the deadline .

Since communication range limitation is a common problem in disaster domains, we assume that agents can only discover the information about tasks and communicate with other agents close to its location within a limited communication range.

**Definition 7:** A *Neighbour* of $A_j$ is an agent that can directly communicate with $A_j$. In other words, a neighbour of $A_j$ is an agent within $A_j$'s communication range, i.e. $Comm_j$ (see Definition 2).

## 3 The Basic Processes of Our Approach

Our approach consists of five looped steps, which are 1) task token creation, 2) group formation, 3) token passing, 4) task allocation and 5) allocation solution return. After one loop of five steps, if there are available agents and tasks in the domain, these five steps will be repeated again according to the current location of available tasks and agents.

### 3.1 Task Token Creation

In this step, each available agent $A_j$ (i.e. $ASta_j$ = 'available', see Definition 2) scans the area within its communication range ($Comm_j$). Once, $A_j$ finds an available task $T_i$ (i.e. $TSta_i$ = 'available', see Definition 1), $A_j$ creates a task token, $TToken_k{=}{<}A_j, T_i, DLine_i, WLoad_i, Loc_i, Emg_i{>}$ (see Definition 3) for $T_i$. Meanwhile, $A_j$ also creates an agent token $AToken_j$ (see Definition 4) for itself.

### 3.2 Group Formation

After task tokens are created, agents will form groups and are allocated tasks within groups. Due to communication range constraints, agents can only communicate with their neighbours (see Definition 7). In order to make decisions for task allocation based on available resources as much as possible, the group formation mechanism is designed to connect as many agents as possible within the domain. The objective of the group formation mechanism is to link agents with their neighbouers and form a tree, in which the coordinator of the group is the root node. In the mechanism, we defined three 'neighbour related' variables for each agent, including, the parent agent (PA), the coordinator (C) and the number of neighbors of the coordinator (NNC). For example, for an agent $A_j$, the three variables of $A_j$ is denoted as $A_j$.PA, $A_j$.C and $A_j$.NNC, respectively. The group formation mechanism is conducted based on an iteration process.

The group formation mechanism is described by Algorithm 1. At the beginning of the iteration, the three variables of each agent (i.e. $A_j$) are initialised as follow: $A_j$.PA is set to $A_j$, $A_j$.C is set to $A_j$ and $A_j$.NNC is set to the number of neighbours of $A_j$ (since $A_j$ is the coordinator of itself) (see line 2). Each agent (e.g. $A_j$) repeats the following three steps. **Step 1:** $A_j$ finds an agent (e.g. $A_u$) which has the highest value of the variable NNC (i.e. $A_u$.NNC) among neighbours of $A_j$ (including $A_j$ itself) (see line 5); **Step 2:** the variables of $A_j$ ($A_j$.PA, $A_j$.C and $A_j$.NNC) are updated as follow: $A_j$.PA is set to $A_u$, $A_l$.C is set to $A_u$.C and $A_j$.NNC is set to $A_u$.NNC (see line 6); **Step 3:** After above two steps, each neighbour of $A_j$ (e.g. $A_l$), compares its coordinator ($A_l$.C) with the coordinator of $A_j$ ($A_j$.C). If an different coordinator has been found ($A_l$.C $\neq$ $A_j$.C), the variables of $A_l$ ($A_l$.PA, $A_l$.C and $A_l$.NNC) are updated as follow: $A_l$.PA is set to $A_j$, $A_l$.C is set to $A_j$.C and $A_l$.NNC is set to $A_j$.NNC (see lines 8 to

9). The above three steps (lines 4 to 9) will be repeated by each agent until no further updating for three variables of any agent.

**Algorithm 1:** Group Formation

```
1  for each Agent $A_j$ do
2      $A_j$.PA← $A_j$; $A_j$.C← $A_j$; $A_j$.NNC← number of neighbours of $A_j$
3  end
4  for each Agent $A_j$ do
5      Get $A_u$ from $A_j$ and its neighbours, where $A_u$.NNC is Maximum
6      $A_j$.PA← $A_u$; $A_j$.C← $A_u$.C; $A_j$.NNC← $A_u$.NNC
7      for each neighbours of $A_j$:$A_l$ do
8          if $A_l.C \neq A_j.C$ then
9              $A_l$.PA← $A_j$; $A_l$.C← $A_j$.C; $A_l$.NNC← $A_j$.NNC
10         end
11     end
12 end
```

## 3.3 Token Passing

In this step, each agent $A_j$ first gets task tokens (see Definition 3) and agent tokens (see Definition 4) (created in the task token creation step) from its child agents. Different agents in the same group can create task tokens for the same task. For example, $A_u$ and $A_{u+1}$ are child agents of $A_j$ and they create task tokens $TToken_k$=<$A_u$, $T_i$, $DLine_i$, $WLoad_i$, $Loc_i$, $Emg_i$> and $TToken_{k+1}$=<$A_{u+1}$, $T_i$, $DLine_i$, $WLoad_i$, $Loc_i$, $Emg_i$> for the same task $T_i$. It is can be seen that except for the first item i.e. the agent who created the token, the two task tokens created by $A_u$ and $A_{u+1}$ are exactly the same. In this situation, $A_j$ combines the two tokens together ($TToken_{k+1}$ is abandoned and only $TToken_k$ is kept). After this, $A_j$ passes the combined task token/tokens and agent tokens that $A_j$ received and $AToken_j$ to the parent agent of $A_j$. Finally, all of task tokens and agent tokens are passed to the coordinator of the group.

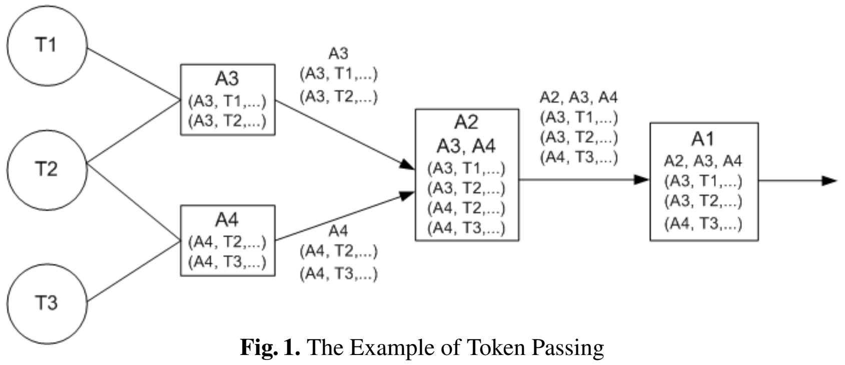

**Fig. 1.** The Example of Token Passing

Fig. 1 shows an example of token passing. There are four agents: $A_1$, $A_2$, $A_3$ and $A_4$, in the example. $A_1$ is the parent agent of $A_2$ ($A_2$.PA=$A_1$), $A_2$ is the parent agent of $A_3$ and $A_4$ ($A_3$.PA=$A_2$ and $A_4$.PA=$A_2$ ). There are three tasks in the example: $T_1$, $T_2$ and $T_3$, $T_1$ and $T_2$ can be directly found by $A_3$ and $T_2$ and $T_3$ can be directly found by $A_4$. After the task token creation step, $A_3$ creates two task tokens (see, Definition 3) for

$T_1$ and $T_2$, which can be represented as '($A_3$, $T_1$, ...)' and '($A_3$, $T_2$, ...)', respectively, while $A_4$ also creates two task tokens (see, Definition 3) for $T_2$ and $T_3$, which can be represented as '($A_4$, $T_2$,... )' and '($A_4$, $T_3$, ...)', respectively. Then, the task tokens with the agent tokens (see, Definition 4) of $A_3$ and $A_4$ are passed to $A_2$. After checking the task tokens getting from $A_3$ and $A_4$, $A_2$ discovers that task tokens '($A_3$, $T_2$, ...)' and '($A_4$, $T_2$, ...)' have been created for the same task by two different agents. Therefore, $A_2$ combines the task tokens '($A_3$, $T_2$,... )' and '($A_4$, $T_2$,... )' to '($A_3$, $T_2$, ...)'. After combination of task tokens, $A_2$ passes its agent tokens ($A_2$, $A_3$ and $A_4$) and the combined task token ('($A_3$, $T_2$, ...)') and its other task tokens ('($A_3$, $T_1$, ...)'and '($A_4$, $T_3$, ...)') to $A_1$.

## 3.4 Task allocation

This step is the core of the approach. In this step, a coordinator tries to find the best allocation solution for the group according to available tasks and agents in the group. In our approach, the allocation solution of a group is represented by an n-tuple $Alloc =< ANo_1 \rightarrow TNo_1, ANo_2 \rightarrow TNo_2, ..., ANo_n \rightarrow TNo_n >$, where, $ANo_j$ is the ID of the $j^{th}$ agent of the group and $TNo_j$ is the ID of the task to which the $j^{th}$ agent of the group is allocated. If the $j^{th}$ agent of the group is not allocated to any task, the $TNo_j$ is $\varnothing$. Since finding the best allocation solution is an NP-hard problem (the proof process can be found in [12]), our approach employs heuristics. The heuristics for finding the best allocation solution can be implemented in two sub-steps 1) useless allocation elimination and 2) utility calculation

**Useless Allocation Elimination**

Based on the combinatorics, the number of combination for a group with $m$ number of tasks and $n$ number of agents is $m^{n+1}$. Among these allocations, there are allocations where the allocated agents cannot be sure to finish the task in time (before deadline). For example, in a group, one of allocation solution is $Alloc_a =< A_1 \rightarrow T_1, A_2 \rightarrow T_2, A_3 \rightarrow T_1 >$. After calculation, if we find that $A_1$ and $A_3$ cannot finish $T_1$ according to the status of $A_1$, $A_2$ and $T_1$, this allocation solution is useless and should be eliminated to make the following calculation more efficient. The process of this elimination can be described as follows. First, the coordinator gets $m^{n+1}$ allocation solutions of the group. Second, for each task $T_i$ existing in the allocation solution $Alloc$, we calculate the maximum workload of agents allocated to $T_i$, which can perform before the deadline of $T_i$. If an allocation solution exists for task $T_i$ which cannot be finished by allocated agents before deadline, the $Alloc_a$ is eliminated.

**Utility Calculation**

In this sub-step, the coordinator calculates the utility of each useful allocation solution and chooses the allocation solution with the highest utility value as the best allocation solution for the group. To judge an allocation solution depends ont only on how many tasks that can be finished, but also on the benefits received from finishing the tasks and the cost of finishing the tasks. In our approach, the utility of an allocation solution $Alloc_a$ is affected by two factors, which are (1) the benefits received from finishing the task and (2) the cost of finishing the tasks.

With the two factors above, the utility of an allocation solution $Alloc_a$ can be calculated by the following formula.

$$Utlity_{Alloc_a} = Q_{benefit} - Q_{cost}. \quad (2)$$

The benefits received from finishing tasks involves two factors: 1) the degree of urgency of the task and 2) the time saved by finishing the task. The benefits of an allocation solution $Alloc$ is the sum of the benefits of all finished tasks in $Alloc_a$, which is calculated as follows.

$$Q_{benefit} = \sum_{t=0}^{n} Emg_i \cdot (WLoad_i + \frac{(DLine_i - FTime_i) \times WLoad_i}{DLine_i}), \tag{3}$$

where $Emg_i$, $WLoad_i$, $DLine_i$ are the degree of urgency, the workload and the deadline of $T_i$ in $Alloc_a$ and $FTime_i$ is the predicted finishing time of $T_i$ in $Alloc_a$.

The cost for finishing tasks in our approach includes 1) the traveling time of each agent moving from its current location to the location of the its allocated task and 2) the time each agent spends on finishing the allocated task. Therefore, the cost of an allocation solution $Alloc_a$ is the sum of the cost of agents in $Alloc_a$, the cost of traveling and of finishing tasks, which can be calculated by Equation 4 as follows.

$$Q_{cost} = \sum_{t=0}^{m}((FTime_i - CurTime) \times Uti_j), \tag{4}$$

where $Uti_j$ is the working efficiency of $A_j$, $FTime_i$ is the predicted finishing time of $T_i$ in $Alloc_a$, which $A_j$ is allocated to and $CurTime$ is the current time when the coordinator calculates the utility function of $Alloc_a$. $FTime_i$ can be calculated as follows:

$$FTime_i = \frac{WLoad_i + (CurTime - StaTime) \times \sum_{j=1}^{n} Uti_j + \sum_{j=1}^{n} Time(T_i, A_j) \times Uti_j}{\sum_{j=1}^{n} Uti_j}, \tag{5}$$

where $Wload_i$ is the workload of $T_i$ in $Alloc_a$, $CurTime$ is the current time point when the coordinator calculates the utility function of $Alloc_a$,$StaTime$ is the start time of the system begin to work, $Uti_j$ is the $j^{th}$ agent, which is allocated to $T_i$ in $Alloc_a$ and $Time(T_i, A_j)$ is the traveling time from the current location of $A_j$ to the location of $T_i$.

### 3.5 Allocation Return

In this step, the coordinator sends the chosen allocation solution to the agents in the group. Once an agent received an allocation, it will do the following things. First, the tokens of allocated tasks (which it can definitely finish) is eliminated from the agent token pool. Second, the allocation is passed to the neighbours of the agent and the status of the agent is changed from 'available' to 'working'. Finally, the agent moves to the location of the task and starts working on it. When the first agent reaches the location of a task, it changes the status of the task from 'available' to 'working'. After finished the task, all of agents, which worked on the task, change their status from 'working' to 'available' and the status of the task is also changed from 'working' to 'finished'.

## 4 Experiment and Analysis

Two experiments are conducted to evaluate the performance of our approach. Experiment 1 is to evaluate the performance of our approach among agents with different ranges of communication. The benchmark of Experiment 1 is the F-max-sum algorithm proposed by Ramdchurn et al. [11]. The F-max-sum is a decentralised task allocation mechanism. Experiment 2 is to evaluate how the degree of urgency of tasks impact on the performance of our approach. Two experiments are demonstrated and analysed in detail in the following two subsections, respectively.

## 4.1 Experiment 1

The purpose of this experiment is to test the impact of the different communication ranges on the performance of the proposed approach.

**Experimental Settings**

In Experiment 1, 100 tasks (in which 50 tasks already exist at the beginning of the experiment and 50 new tasks are discovered between 0 to 100 time units) and 10 agents with different initial locations in a $50 \times 50$ areas are employed. Since the degree of urgency of each task is not being tested in Experiment 1, the degree of urgency of all of tasks is set to 1. In addition, all tasks have a workload of between 10 to 60 work units and must be finished before the deadline between 5 to 100 time units. Agents can only finish 1 work unit or move 1 distance per time unit. The settings of Experiment 1 are shown in Table 1.

| Name | Value | Name | Value | Name | Value |
|---|---|---|---|---|---|
| Number of tasks | 50 | Deadline of tasks | $5 \sim 100$ | Number of agents | 10 |
| Number of new tasks | 50 | Workload of tasks | $10 \sim 60$ | Work Efficiency | 1 |
| Urgent Degree | 1 | Area size | $50 \times 50$ | | |

**Table 1.** The Settings in Experiment 1

In this experiment, we use three scenarios to represent three different kinds of communication ranges, shown in Table 2.

| Scenario | Communication range (unit) | Communication capacity |
|---|---|---|
| 1 | 10 | isolated |
| 2 | 20 | limited |
| 3 | 50 | full |

**Table 2.** Three Scenarios in Experiment 1

This setting tries to simulate tree different communication conditions, i.e. ‘isolated’, ‘limited’ and ‘full’ communication. The two approaches (i.e. the proposed approach and the benchmark approach) use their utility functions to choose that which tasks should be finished and which tasks should be abandoned. In addition, the Manhattan distance [1] is employed to calculate the distance between two locations in the experiment. The group formation mechanism is affected by three factors: the number of agents in the domain, the communication range of agents and the size of the domain. From the definition of the Manhattan distance, in an $m \times m$ size domain, if the communication range of agent $A_j$ is $c$, the possibility that another agent is the neighbour of $A_j$ is calculated as follows.

$$Possibility = \frac{2 \cdot c^2 + 2 \cdot c + 1}{m \cdot m} \tag{6}$$

With increasing communication range or agent numbers and decreasing domain size, more agents can find neighbours and form groups.

Since the F-max-sum does not consider the communication range, we assume that agents in both approaches consider communication range. In most situations, agents need to cooperate to complete tasks. The two approaches are compared with three different communication ranges between agents 10, 20 and 50 in the three scenarios, respectively. In Scenario 1 (10 communication range), according to Equation 6, in an $50 \times 50$ area, the possibility of two agents being neighbour of each other is 8.84%, which is an extreme “lonely” setting for 10 agents. In Scenario 2 (20 communication range), the possibility of two agents being neighbour of each other is 33.6%. This is a relatively good setting for 10 agents, because there is increased possibility for agents to form groups and there will also be some agents that cannot connect with other agents. In Scenario 3 (50 communication range), there is no communication range constraints between agents and all agents can communicate with each other directly.

**Experimental Results and Analysis**

The experimental results for three scenarios are shown in Fig. 2., 3. and 4., respectively. The X-axis of three figures represents the consumed time units. The Y-axis of three figures indicates the number of tasks that have been finished.

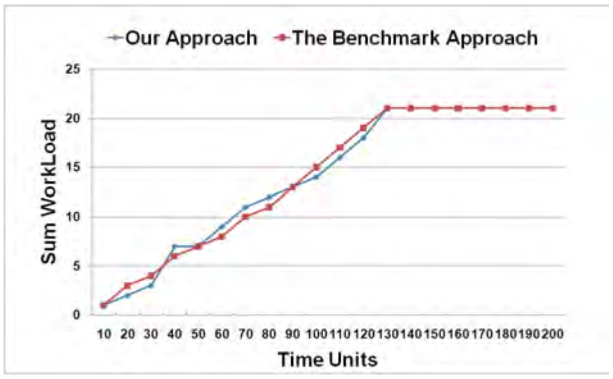

**Fig. 2.** Experimental Result in Scenario 1

**Scenario 1:** Since the communication range for each agent is only 10, in this scenario, it is hard for agents to work cooperatively (i.e. each agent is isolated). All of the agents in both approaches work independently. From Fig. 2., we can see that there are nearly the same finished workloads between the two approaches.

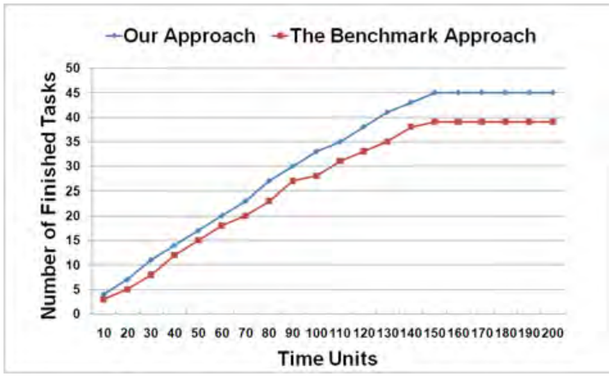

**Fig. 3.** Experimental Result in Scenario 2

**Scenario 2:** The communication range for each agent is 20, in the scenario, and according to Equation 6, the probability for two agents in the domain to be neighbours is around 33%, so the communication capacity is limited. From Fig. 3., we can see that our approach has better performance than the benchmark approach. That is because when agents form groups, the coordinator in our approach can collect the task and agent information from the members of the group and find the most suitable allocation solution for the group. However, the benchmark approach is limited by communication range and agents can not have full knowledge for all of tasks. Hence, agents can not offer the comprehensive messages to other agents and this affects the decision making of agents to achieve good performance in the domain.

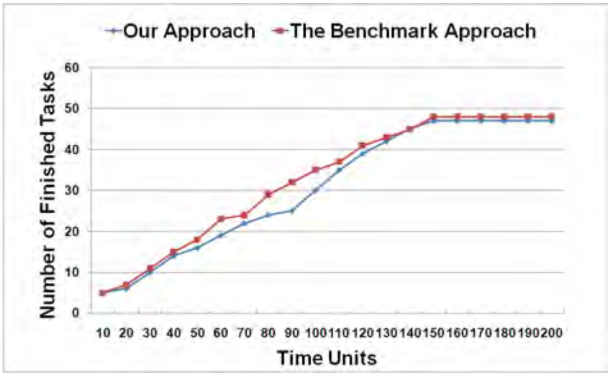

**Fig. 4.** Experimental Result in Scenario 3

**Scenario 3:** The communication range of this scenario is 50 in a $50 \times 50$ size area, so an agent can have a global view of all tasks and other agents in the area. Agents in the domain can fully communication with any other agents. Fig. 4. demonstrates that our approach is equal to a centralised approach and the F-max-sum algorithm can also achieve the highest efficiency with the full communication capacity, so the performance of the two approaches is nearly the same.

From the experimental results produced from three scenarios, it can be seen that the communication range has a bigger impact on the F-max-sum algorithm for task allocation than that of in our approach. The reason is that the F-max-sum algorithm requests good communication capacity between agents, so decision making for each agent in an area is limited if an agent can not offer comprehensive information of itself to other agents. In our approach, since the group formation algorithm can help the coordinator to find much information in a domain, the coordinator can reduce the impact of the limited communication range on task allocation and make the most suitable decision for the group. Therefore, our approach performs better than the benchmark approach when there is limited communication range.

### 4.2 Experiment 2

The purpose of this experiment is to test the impact of different degrees of urgencies of individual tasks on task allocation.

**Experiment settings**

In Experiment 2, 100 tasks and 10 agents with different initial locations in a $50 \times 50$ domains are employed. All of the tasks have workloads between 10 to 60 work units, which must be finished before the deadline between 5 to 100 time units. Agents in the domain can only finish 1 work unit or move 1 distance per time unit and have a fixed communication range of 20. To evaluate the impact of degree of urgency, our approach is conducted twice. First time, the degree of urgency of all tasks is 5, which indicates that there is no difference between the degree of urgency of all agents, while in the second time, the degree of urgency of all tasks varied between 1 and 9 with normal distribution. The settings of the experiment are shown in Table 3.

| Name | Value | Name | Value | Name | Value |
|---|---|---|---|---|---|
| Number of tasks | 100 | Workload of tasks | $10 \sim 60$ | Work Efficiency | 1 |
| Urgent Degree | $1 \sim 9$ | Area size | $50 \times 50$ | Communication range | 20 |
| Deadline of tasks | $5 \sim 100$ | Number of agents | 10 | | |

**Table 3.** The Settings in Experiment 2

**Experimental Results and Analysis**

The experimental results of Experiment 2 are shown in Fig. 5 and 6. In Fig. 5, the X-axis is the consumed time units, while Y-axis is the sum of the task workloads that have been finished. In Fig. 6, the X-axis is the consumed time units, while Y-axis is the sum of weighted workloads of the tasks that have been finished.

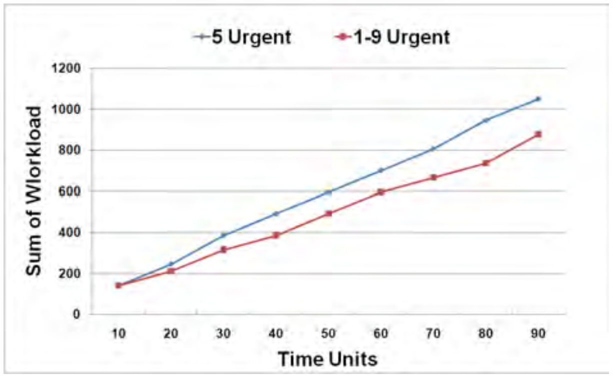

**Fig. 5.** Sum Workload

From Fig. 5., it can be seen that the proposed approach can finish more workload, when the degree of urgency of the tasks are similar. This is because when the degrees of urgency are similar for all tasks, the coordinator allocates agents to tasks only according to the workload of each task and the distance between agents and tasks.

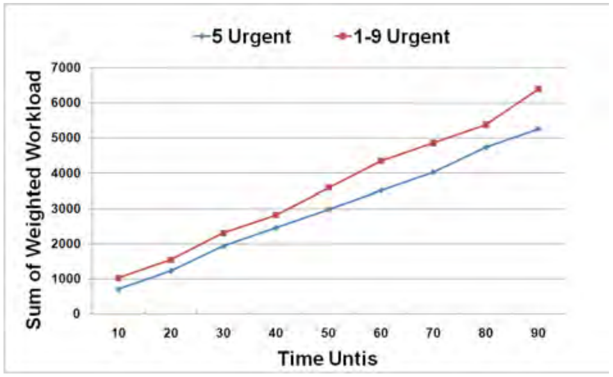

**Fig. 6.** Sum Weighted Workload

Form Fig. 6., we can see that the proposed approach can finish more weighted workload, when the degree of urgency of the tasks is different. The reason for this situation is that the proposed approach consider the degree of regency of tasks in its utility calculation function.

From Fig. 5 and 6, we can see that if we differentiate the degrees of urgency of tasks in the domain, the proposed approach tries to allocate the more urgent tasks before the less urgent tasks, which is reasonable.

### 4.3 Summary

From Experiment 1 and 2, we can conclude that (1) the communication range of agents and the degrees of urgencies of tasks have great impact on decision making for task allocation, particularly in a diaster domain, because the limited communication range reduces the local view of an agent; and (2) using the equivalent degree of urgency for all tasks can bury the importance of tasks.

Our approach uses group formation mechanism to expand the local view of agents and differentiates tasks by employing the concept of the degree of urgency of each task during task allocation. The experimental results indicate that our approach has better performance than the famous approach (F-max-sum) under limited communication range. In addition, using the weighted workload concept can help agents to finish urgent tasks under time constraints.

## 5 Related Work

Ramamritham et al. proposed using distributed coordination to handle task allocation with deadline and resource requirements in [10]. Their approach solves common task allocation problem based on the classification of tasks. The weakness of their approach is to omit the space and communication constraints in the process of task allocation. Furthermore, in a disaster domain, most tasks are critical and it is rare that there are tasks of other categories. Therefore, the approach proposed by Ramamritham et al. is hard to be applied to a disaster domain. Our approach considers the space and communication constraints and can also deal with tasks with different degrees of urgencies during task allocation in a diaster domain.

Some researchers such as Katirya and Jennings have used mixed integer linear programming (MILP) [5] to get the optimal allocation solution in a disaster domain in [12], [6]. The MILP process can guarantee an optimal solution for task allocation if the coordinator can have a global view of the domain. However, the MILP has two main weaknesses. First, in a common disaster domain, it is hard for a coordinator to have a global view due to communication constraint. Second, the MILP can only deal with problems with fixed numbers of tasks and agents. These features make their solutions hard to fit the highly dynamic natures of a disaster domain. Third, the computation complexity of the MILP is high. Our approach can solve dynamic task allocation problems in a decentralised manner.

In recent years, the DARPA coordinators program is a popular environment for the coordination of task allocation, which has a complicated hierarchy task structure. The difference between the DARPA coordinators program and ours can be summarized as follows. 1) The tasks in the DARPA coordinators program do not have clear deadlines. 2) Most proposed models [9], [14], [8], [2] dealing with the DARPA coordinators program do not consider either space or communication constraints. The approach proposed in this paper considers the coordination problem with both space and communication constraints.

## 6 Conclusion and Future Work

In this paper, a novel decentralised coordination approach for task allocation with space, time and communication constraints is proposed. The approach uses a comprehensive utility function to enable a coordinator to find the best solution for task allocation in the group by considering degree of urgency of tasks, workload and traveling time of agents to the location of a task under multiple constraints. A group formation mechanism is set up in order to help agents to form a group within limited communication range due to the destruction of infrastructures in a diaster domain. In addition, the degree of urgency of each task and dynamic features of a such domain are covered by our approach in order to meet the requirements of a disaster domain. The experimental results have shown that our approach can provide a more effective way to meet the challenges of task allocation in a disaster domain than other existing approaches. In the future, we will work on task allocation problems in more complex environments, e.g., agents can only have spars interaction.

## References

1. Taxicab geometry. `https://en.wikipedia.org/wiki/Centroid`.
2. L. Barbulescu, Z. B. Rubinstein, S. F. Smith, and T. L. Zimmerman. Distributed coordination of mobile agent teams: the advantage of planning ahead. In *AAMAS '10*, pages 1331–1338, Richland, SC, 2010.
3. A. C. Chapman, R. A. Micillo, R. Kota, and N. R. Jennings. Decentralised dynamic task allocation: a practical game: theoretic approach. In *AAMAS '09*, volume 2, 2009.
4. A. Farinelli, A. Rogers, A. Petcu, and N. R. Jennings. Decentralised coordination of low-power embedded devices using the max-sum algorithm. In *AAMAS '08*, volume 2, pages 639–646, Estoril, Portugal, 2008.
5. G. J. Gordon, S. A. Hong, and M. Dudík. First-order mixed integer linear programming. In *UAI '09*, pages 213–222, Arlington, Virginia, United States, 2009. AUAI Press.
6. M. Koes, I. Nourbakhsh, and K. Sycara. Heterogeneous multirobot coordination with spatial and temporal constraints. In *AAAI'05*, pages 1292–1297. AAAI Press, 2005.
7. V. Lesser. Cooperative multiagent systems: A personal view of the state of the art. *IEEE Transactions on Knowledge and Data Engineering*, 11:133–142, 1999.
8. R. Maheswaran, P. Szekely, M. Becker, S. Fitzpatrick, G. Gati, J. Jin, R. Neches, N. Noori, C. Rogers, R. Sanchez, K. Smyth, and C. V. Buskirk. Look where you can see: Predictability & criticality metrics for coordination in complex environments. In *AAMAS '08*, 2008.
9. D. J. Musliner and R. P. Goldman. Coordinated plan management using multiagent mdps. In *AAAI '06*, pages 73–80. AAAI Press, 2006.
10. K. Ramamritham, J.A. Stankovic, and W. Zhao. Distributed scheduling of tasks with deadlines and resource requirements. *Computers, IEEE Transactions on*, (8), 1989.
11. S. D. Ramchurn, A. Farinelli, K. S. Macarthur, and N. R. Jennings. Decentralized coordination in robocup rescue. *Comput. J.*, 53(9):1447–1461, November 2010.
12. S. D. Ramchurn, M. Polukarov, A. Farinelli, C. Truong, and N. R. Jennings. Coalition formation with spatial and temporal constraints. In *AAMAS 2010*, pages 1181–1188, 2010.
13. F. Ren and K.M. Sim. Adaptive conceding strategies for automated trading agents in dynamic, open markets. *Decision Support Systems*, 48(2):331–341, 2009.
14. S. F. Smith, A. Gallagher, and T. Zimmerman. Distributed management of flexible times schedules. In *AAMAS '07*, pages 74:1–74:8, New York, NY, USA, 2007. ACM.
15. M. Vinyals and M. Pujol. Divide-and-coordinate: Dcops by agreement. In *AAMAS '10*, pages 149–156, 2010.