Intelligent Algorithm for Assignment of Agents to Human Strategy in Centralized Multi-agent Coordination

Reza Nourjou[a], Stephen F. Smith[b], Michinori Hatayama[a], Pedro Szekely[c]
[a] Informatics Graduate School and Disaster Prevention Research Institute, Kyoto University, Japan
Email: {nourjour, hatayama}@imdr.dpri.kyoto-u.ac.jp
[b] The Robotics Institute, Carnegie Mellon University, USA
Email: sfs@cs.cmu.edu
[c] Information Sciences Institute, University of Southern California, USA
Email: pszekely@isi.edu

Abstract— **Problem: Multi-agent coordination is an important issue in the domain of disaster emergency response operations where a team of agents (field units or robots) aims to achieve a joint objective. The responsibility of the Incident Commander (IC) is to (I) specify an effective strategy composed of a number of threads (a set of prioritized sub-problems), (II) appropriately assign/allocate agents to these threads as a strategic decision, and (III) release agents in a timely manner from the assigned threads to adapt a strategic decision to a new situation. Objective: The purpose of this paper is to present an intelligent algorithm that assists a human in multi-agent coordination by providing two key functions: 1) automatically calculate and present a set of feasible alternatives for selecting a choice as a strategic decision in a definite time, and 2) autonomously and in a timely manner identify a subset of assigned agents that should be released from their threads in order to refine a strategic decision. Method: This algorithm expands a decision tree from a state node in which a thread (or several threads) has received a set of new agents from either the IC or a higher thread. Each thread is associated with one level of a decision tree with a number of nodes. A thread calculates a set of efficient coalitions using all the available agents and generates a new node for each coalition to show what agents are allocated to the thread and what agents are released into a lower thread. In real-time, this algorithm continuously observes and monitors the task environment to identify a subset of the assigned agents that cannot provide efficient capabilities for their threads and should be released for assignment to other threads. Results: To gather further insight, this paper applied this algorithm for team coordination to a simulated search & rescue scenario in an earthquake disaster-affected area where the team's goal was to rescue trapped people distributed in five operational zones. The result was an infinite set of alternative scenarios for a human-defined strategy. The calculated alternatives were presented to the IC for selection according to his intuition or for delegation to the system to determine an optimal strategy.**

Index Terms— **Agent assignment problem, incident commander, multi-agent coordination, human strategy, intelligent algorithm, crisis response, action planning**

Manuscript received December 05, 2013; revised December 25, 2013; accepted February 27, 2013.

I. Introduction

A review of past disasters (natural or human-made) shows that the coordination of disaster emergency/crisis response operations is a significant and essential issue. A responder team such as the INSARAG (International Search and Rescue Advisory Group) [7] that is composed of an IC (Incident Commander) and several field/operational units, called *agents* in this paper, is faced with the problem of effectively performing spatially dispersed tasks under evolving execution circumstances in a manner that achieves a joint objective in a minimum time. Coordination is the act of managing interdependencies among activities performed to achieve a goal [11]. To maximize the global result, five types of coordination are considered to be managed within the team: 1) task dependencies, 2) action dependencies, 3) redundant actions, 4) information sharing, and 5) agent allocation [15]. Inefficient coordination can result in idle agents, conflict among actions, or redundant activities, and consequently, operations can require a very long duration to be completed. Effective coordination is both difficult and challenging because of the characteristics of this domain including uncertainty in the outcome of the actions of agents, incomplete task information, time pressure, limited resources, and task flow [4].

Coordinating a team of agents is a critical issue because the aim of the team is to achieve a joint objective. Centralized multi-agent coordination is the primary role of the IC. In this role, the IC must make three important decisions: 1) select a global objective and specify a strategy composed of a number of threads that are a set of prioritized sub-problems, 2) strategically allocate agents to these threads, 3) identify an appropriate time to adapt the strategic decision to a new situation by identifying a subset of assigned agents that should be released from their threads and be sent to other threads [13], [14]. Human decisions define the macro behavior of the team because they do not explicitly specify micro actions that should be performed by the agents. A strategic decision specifies a domain of agent actions to maximize the global

doi:10.4304/jsw.9.10.2586-2597

results.

The calculation of the feasible alternatives for facilitating a strategic decision is an essential process. These alternatives enable the IC to understand what options are available at a specific time. Furthermore, in a real-time situation, identifying the proper time to revise a strategic decision is a difficult and challenging problem. These two issues, however, are critical for efficient coordination. It is a demanding problem for a human because of the complexity of the crisis emergency response process. Moreover, a fully automated system cannot provide a complete solution for this problem and is not considered a reliable substitute for a human.

Consequently, it is both important and necessary to develop an ideal approach based on a mixed-initiative solution in which an intelligent system assists the IC in the multi-agent coordination and collaborates with him in the decision making process. This paper focuses on two sub-problems and the proposed issues that address them: 1) how to assist the IC and collaborate with a human in an appropriate assignment of agents to threads, and 2) how to support a human in the adaption of a thread assignment to a new critical situation in a timely manner.

Many works have been developed for multi-agent coordination especially for planning, scheduling, tasks assignment, action coordination, decision making, and optimization. Unfortunately they do not provide a proper solution for the problem addressed by this paper. The deficiencies can be categorized as follows:

- They aim to fully specify the micro-actions of the agents. It is impossible, however, for an IC to make all the decisions and define the detailed actions for agents who are located in uncertain and dynamic environments. Agents, human or robot, select and perform the best action based on their perception of the local environment using detailed information.
- They are automated systems that do not involve a human in the process. The main obstacle is scale. It is currently unfeasible for a fully automated system to effectively determine all future possibilities that may arise during the execution of tasks in a complex environment.

The objective of this paper is to present an autonomous algorithm. The two most important benefits to an IC of the application of this algorithm in a centralized multi-agent coordination are: 1) automated calculation of a set of feasible alternatives for selecting the strategic decision at a definite time, and 2) autonomous adaption of a strategic decision at the proper time by identifying a subset of agents who should be released from a subset of threads.

This paper is organized as follows. For further insight, Section II provides our motivation for this paper by introducing a simulated urban search & rescue scenario in which the IC of a team of four agents is faced with an agent coordination problem. Section III reviews some related works. The problem is stated in more detail in Section IV. The approach and proposed algorithm are presented in Section V. Section VI discusses the implementation and evaluation process. Section VII presents the result and we conclude our paper in Section VII.

II. Motivation

Let us imagine that an earthquake has occurred in the urban area displayed in Fig. 1. Urban search and rescue (USAR), as a major function of the disaster emergency response operations, aims to rescue the greatest number of people trapped under the debris of the damaged buildings in the shortest period of time. To rescue a victim in a certain spatial location, a sequence of three dependent location-based tasks must be accomplished. Each type of task requires a set of definite capabilities and a considerable amount of time. Table I lists these task types and their capability requirements. We ignore the deadline for tasks in this paper.

Figure 1: GIS map of spatial distribution of location-based temporal macro tasks in five operational zones (road segments) of disaster-affected area and location of disaster response team in the initial state

TABLE I.: Three Task Types of Problem Domain

Task-Type	Δt (minute)	Capability Requirements *C0*	*C1*	*C2*
T0	5	1	0	0
T1	20	0	1	0
T2	60	0	0	1

Capabilities Description:
C0: Reconnaissance
C1: Search
C2: Light Rescue

Task Types Description:
T0: Reconnaissance
T1: Search
T2: Light Rescue

A team of four agents has been assigned to the five operational areas that are presented by the five road segments displayed in Fig. 1. Actions that an agent can do are presented by capabilities that this agent possesses. An agent can have different capabilities from those required by the tasks. In a specific duration, an agent can execute an action with a specific speed. Table II lists the characteristics of the agents. It is assumed the agents are either *free* (or idle) and are located in the incident command post.

The shortest distances among each of the six road segments are calculated by GIS and are presented in Table III. To perform spatially distributed tasks, agents

TABLE II.: Capabilities/Abilities of Four Agents

Field Unit ID	Action Speed	Number of Capabilities *C0*	*C1*	*C2*
a0	2	1	0	0
a2	1	1	0	0
	2	0	1	0
a6	1	0	1	0
	2	0	0	1
a7	1	0	1	0
	2	0	0	1

are required to move from one location to another through the road network with a moving speed equal to 20. To simplify the problem, these variables do not change over time. These data are used by the IC in the decision-making process. This table and related information are provided and are updated by relevant teams or organizations whose activities are to clear road blockages.

TABLE III.: Shortest Paths (given in meters) among Road Segments in the Geographic Area Visualized in Fig. 1

	s2	**s1**	**s3**	**s4**	**s5**	**s6**
s2	0	225	447	764	364	625
s1	225	0	370	687	418	548
s3	447	370	0	343	452	221
s4	764	687	343	0	618	224
s5	364	418	452	618	0	476
s6	625	548	221	224	476	0

A set of LoTeM tasks (Location-based Temporal Macro tasks) are associated to each road segment. A LoTeM task is an aggregate task of all the tasks with the same task type that are spatially located within a geographic area such as a road segment or city block [13]. Table IV shows that twelve LoTeM tasks are located in the five segments at time 0. The information regarding these tasks forms the overall picture that the IC observes from the task environment. For example, the 10th LoTeM task states that in the proximity of road segment s5, five light rescue tasks are estimated to be revealed and six tasks are available and ready to be started. It is also estimated that if the ninth LoTeM task (2 Not Yet Enabled, 5 Enabled) is completed, five light rescue tasks will be revealed in the same road segment. All of this information is presented to the IC to provide a situational awareness of the state of the environment in a timely manner.

TABLE IV.: Set of Location-based Temporal Macro Tasks Associated to Five Road Segments Displayed in Fig. 1

No.	Location (Road S.)	Task-Type	Not Yet Enabled Amount	Enabled Amount
1	s1	T0	0	25
2	s1	T2	0	4
3	s2	T1	0	5
4	s2	T2	10	5
5	s3	T0	0	15
6	s3	T1	8	0
7	s3	T2	2	8
8	s5	T0	0	10
9	s5	T1	2	5
10	s5	T2	5	6
11	s4	T1	0	18
12	s4	T2	10	2

The first step in the coordination of agents by the IC is to specify a strategy. The joint objective that the IC selects for the team is to rescue all the victims distributed in the five operational areas. Table V presents the human strategy composed of three threads to achieve the defined goal. Thread 1 states that the first priority of the team is to perform reconnaissance and search operations in three geographic areas {s3, s4, s5}. Any *appropriate* subset (sufficient coalition) of four agents {a0, a2, a6, a7} can be assigned to this thread to accomplish the tasks that are spatially distributed within these areas. The human strategy has also defined agent a2 for three threads. To specify a strategy, the IC considers two important facts: 1) agent availability, and 2) enabled tasks. Agents shared among threads and task dependencies among the threads make threads either completely or partially interdependent. For example, the search LoTeM task of the first thread enables/reveals the light rescue LoTeM task of the second thread.

TABLE V.: Example of Human Strategy with Three Threads Specified by the IC for Centralized Multi-agent Coordination

Thread Id	Sub-Location	Sub-Goal (Task Type)	Sub-Team
1	s3, s4, s5	T0, T1	a0, a2, a6, a7
2	s3, s4, s5	T2	a2, a6, a7
3	s1, s2	T0, T1, T2	a0, a2, a6, a7

The second step in multi-agent coordinating is to execute the human strategy by assignment (or allocation) of the agents to the threads. We assume that the IC has sent four free agents to thread 1. To distribute the agents among the threads, the IC applies two simple methods: 1) select a sufficient subset of available agents for a thread for which this subset provides a maximum amount of capabilities, and 2) select a sufficient subset of available agents for a thread for which this subset provides a minimum yet sufficient amount of capabilities. The result is a set of two alternatives for selecting a choice as the final strategic decision as shown in Table VI. A choice is then made, for example, the first alternative is selected by the IC, and its information is disseminated. This decision constrains all four agent to thread 1, implying that the agents are allowed to only perform reconnaissance or search tasks distributed in {s3, s4, s5}.

In this step, the IC addresses the following questions:

- What alternatives are available for making a choice?
- How is a set of feasible alternatives calculated?
- When are these alternatives calculated?

TABLE VI.: Two Types of Assignment of Agents (AoA) to the Human Strategy as Two Alternatives for Making a Choice at Time 0

No.	Method	AoA to Thread 1	AoA to Thread 2	AoA to Thread 3
1	max	a0, a2, a6, a7		
2	min	a2	a6	a0, a7

During operations, the IC continuously monitors the real-time state of the task environment to identify a correct time to adapt the current strategic decision. This requires

the IC to continually observe the real-time and updated information of the complete environment and identify a subset of assigned agents that should be released from their threads and be assigned to subsequent threads. This results in the re-running of the strategic decision making procedure. The IC must address the following questions:

- Is this the right time to revise/adjust the strategic decision?
- What agents should leave what threads?
- What threads should receive what released agents?

III. Literature Review

The problem addressed by this paper has been thoroughly studied and a data model has been designed for formulating the problem [13]. The design of a GIS-based intelligent software system, called GICoordinator, has been proposed for this problem and the requirements of this system have been defined [14]. These works provide essential information for this paper to design and develop the proper algorithm.

The incident commander system is a disaster-management tool based on a series of rational bureaucratic principles for disaster response [2]. The basic system objectives and plans are established at or near the top of the hierarchy and used as the basis for decisions and behaviors at lower levels. FEMA provides a set of useful manuals and guidelines regarding practices. However, it does not explicitly define the design requirements and algorithms for preparing incident action plans.

STaC is a valid approach that addresses strategic planning issues [10]. There are two inefficiencies, however, in this approach. First, STaC is based on C_TAEMS and is not perfect for modeling this type of problem [13]. Moreover, STaC selects and automatically assigns a set of agents that provides the threads with the minimum capabilities. This decision making approach can sometimes be wrong as it is sometimes best to have all the useful agents assigned to one thread.

There are other works related to the team task-assignment problem that are applicable to an IC. A spatio-temporal task allocation algorithm for human groups is proposed for rescue operations management [19]. The goal of the RoboCupRescue simulation project is to build a simulator of rescue teams (fire-fighting, police, ambulance) acting in large urban disasters to minimize damages caused by earthquakes such as buried civilians, building fires, and blocked roads [8]. Unfortunately, these approaches are automated systems that do not involve the human in the planning cycle. Furthermore, decisions made by these approaches specify agent actions fully and explicitly.

There are related automated mixed approaches that include the participation of human teams. SpDI2A (Spatially Distributed Intelligent Assistant Agents) includes human activity to form a human-agent team. Human-agent teams aim to distribute search and rescue tasks among themselves and assign tasks to the proper teams [12]. The goal of the COORDINATORS program is to create distributed intelligent software systems that help field units adapt their mission plans as the situation around them changes and influences their plans. A single COORDINATOR is partnered with each tactical unit to collaborate and coordinate with the other tactical units to optimize necessary mission changes [1], [9]. Unfortunately, these systems address tactical decisions that are not applicable to the strategic planning problems of incident commanders.

DEFACTO incorporates artificial intelligence, 3D visualization, and human-interaction reasoning into a unique high-fidelity system for training incident commanders. A key aspect focuses on adjustable autonomy that refers to an agent's ability to dynamically change his own autonomy and possibly transfer control of a decision to another human or agent [18]. Adjustable autonomy is different from the strategic planning problems that an incident commander must solve.

IV. Problem Statement

To gather further insight to the problem, Fig. 2 presents the structure of the problem faced by this paper. Required data and desirable results are shows in this figure. This section is dedicated to the description of these elements.

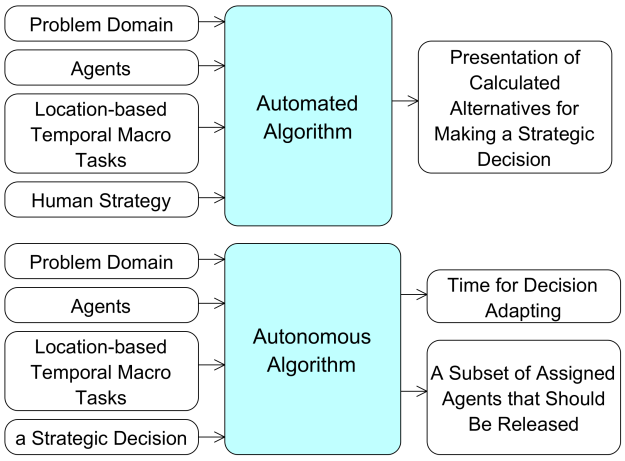

Figure 2: Structure of the problem

A. The Problem Domain

The domain of emergency/crisis response is concerned with reducing the number of fatalities in the first few days after a disaster (natural or human-made). USAR (Urban Search and Rescue) has a significant role in this domain. USAR operations involves four task types: (1) reconnaissance and assessment by collecting information on the extent of the earthquake damage, (2) search and locate victims trapped in collapsed structures, (3) extract and rescue trapped victims, and (4) transport and dispatch injured survivors to hospitals and shelters. Rescue tasks can be classified into three categories: light, medium, and heavy, according to their capability requirements.

Teams are often organized hierarchically. This is especially true for command and control organizations in military and emergency response. A disaster response team

such as a firefighting team or INSARAG has a hierarchical structure composed of an incident commander at the top level of the team and agents in the lower nodes of the hierarchy.

B. Agents

In real domains, a team of agents, field units or robots, is faced with the problem of performing geographically dispersed tasks under evolving execution circumstances in a manner that achieves a high-level objective in a minimum time. These agents are spatially distributed in a geographic environment, have different capabilities, move from one location to another, perceive their local environment, execute strategic decisions made by their incident commander (IC), have partial information of the state of the environment, make full decisions regarding their own actions, coordinate their actions with each other, cooperate with each other, perform various tasks, and report to the operations center. In addition, there is an uncertainty in action duration and outcome. Because he has a global overview of the state of the environment, the role of the IC is to coordinate and control the agents.

Tasks to be performed are spatially located in geographic objects (buildings, road segments, city blocks, or zones), require one capability or several synchronous capabilities, entail various dependencies, have dynamic and temporal quantities, and take considerable time to be completed. Team members work with incomplete and uncertain task information (spatial distribution, quantity, and duration of tasks) and new tasks may be revealed by the actions of the agents or discovered by them in time and space.

C. Location-based Temporal Macro Tasks (LoTeM)

The information in LoTeM tasks forms a global picture of the task environment for the incident commander. A LoTeM task is an aggregate task that combines a subset of tasks that have two criteria: 1) identified tasks are from the same task type, and 2) tasks are spatially contained within a specific geographic object (or adjacent to a road segment) to which this LoTeM task is located. LoTeM tasks are encoded with geographic information.

At a specific time, a LoTeM task contains two variables: 1) an ”Enabled” number of tasks, and 2) a ”Not Yet Enabled” number of tasks. The Enabled variable states the number of available tasks observed, discovered, or revealed within an associated geographic object. Agents can only perform tasks that have become Enabled. The Not Yet Enabled number states how many homogeneous tasks are estimated to be revealed or discovered in the future. These two numbers are dynamic and uncertain quantities that may vary over time because agents can complete enabled tasks while other agents may reveal other tasks. These numbers provide an estimation of the total duration and total capabilities required to perform this task type in a geographic area.

There are interdependencies between LoTeM tasks associated with a specific location. For example, a reconnaissance LoTeM task (number of Enabled plus Not Yet Enabled) can enable the Not Yet Enabled part of the LoTeM search task within the same geographic area.

D. Human Strategy

The workflow of the multi-agent coordination by the IC is presented in Fig. 3. The first step is to define a strategy. Strategy specification enables an IC, as a human, to express and encode his intuition and initiative for the multi-agent coordination and action planning to achieve a global objective. A human strategy partitions and decomposes a complex problem into a finite set of prioritized sub-problems [10], [13]. A human strategy specification is composed of a set of threads. A **thread** contains a unique ranking, sub-team (a subset of agents), sub-goal (a subset of task types), and sub-location (a subset of geographic areas). Human strategy will allow agents to engage in several threads.

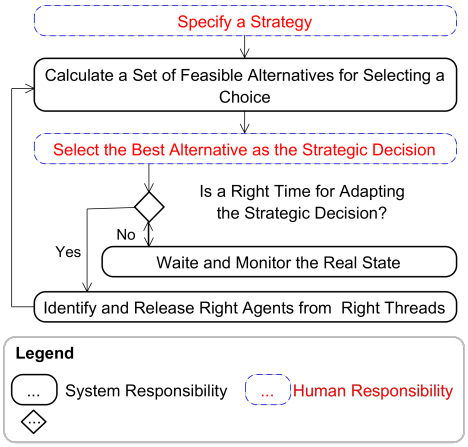

Figure 3: Work flow of multi-agent coordination by the IC

E. Feasible Alternatives

The execution of a strategy is the strategic decision-making process. It is the problem of assignment/allocation of *available* agents to threads at a specific time. A strategic decision states what agents are assigned to what threads and constrains the actions of the agents to the threads. An agent is allocated to only one thread. A thread receives new agents, called available agents, from two sources: 1) from a higher thread that has released them, or 2) from a human who has directly entered them into the thread.

A thread, as an autonomous entity, can select a sufficient subset of available agents and send unrequired agents to a lower thread. There may be a number of these subsets. Therefore, a set of feasible alternative scenarios may exist that present different distributions of the agents among the threads. A feasible alternative can be selected as a strategic decision.

Agents flow from a higher thread to a lower thread through a human strategy. For example, the human strategy specified in Table V includes three threads as three simple sub-problems. First, human strategy execution enters/releases agent a2 to thread 1. Then, this agent is assigned to thread 2 after it is released by thread 1. Finally, this agent will be assigned to thread 3. It takes time for this agent to reach thread 3.

F. Time for Decision Adapting

An agent assigned to a certain thread is retained by this thread *until* this thread releases this agent into the next thread as an idle/available agent. The IC must adjust and adapt a strategic decision to the new state of the environment in a timely manner. It is necessary to detect a correct time for releasing the proper agents from a thread to facilitate making a new strategic decision at a new time.

G. Importance of the Problem

The calculation of feasible alternatives and adapting of a strategic decision are two important responsibilities of an IC in a multi-agent coordination. They are time-consuming processes. They become more difficult and challenging for the IC when the human strategy contains several interdependent threads that require the IC to continuously revise and re-make strategic decisions under rapidly dynamic situations and time pressure. Consequently, this paper focuses on these two significant issues and states two research questions:

- How to develop an automated algorithm that can calculate feasible alternatives for making a choice whenever threads receive new agents?
- How to develop an autonomous algorithm that can monitor the state of the environment to identify what agent (or set of agents) should be released from what thread (or several threads) within a strategic decision at a proper time?

V. ALGORITHM

Our proposal is an automated/autonomous algorithm that assists an IC in a centralized multi-agent coordination. The responsibilities of this algorithm are presented in Fig. 3. The proposed workflow shows a mix-initiative approach in which humans and this intelligent algorithm collaborate on solving a complex problem. This algorithm includes two sub-algorithms whose pseudo code is presented in Algorithm 1 and Algorithm 2.

Algorithm 1 is an automated algorithm that calculates a finite set of feasible alternatives for making a strategic decision. It expands a decision tree from a node in which a thread (or several threads) has received new agents from the IC or from a higher thread. Thread *n* of the strategy is associated with level *n* of the decision tree from which it receives, from a higher thread, a number of nodes that are in fact leaf nodes of the tree. This thread expands the tree from all the received nodes and generates new nodes for each received node. Each node expresses what agents have been assigned to higher threads and what agents have been released into this thread. Finally, new nodes generated by this thread are disseminated to the lower threads.

Data: n :an entity of the "stateNode" class of the data model.
Data: S :an entity of the "strategy" class.
Data: D :the problem domain.
Data: p :type of selection method.
Result: N :a set of entities of the "stateNode" class that are final feasible alternatives.

for $i \leftarrow 1$ **to** $|S.Threads|$ **do**
  $t \leftarrow S.Threads[i]$;
  $ta \leftarrow n.ThreadAssignments\_node[i]$;
  $ta.MacroTasks\_ofThread \longleftarrow$
  $f\_Calculate\_MacroTasks(n.Segments\_node, t, D)$;
end
$Na \longleftarrow \emptyset$;
$Nb \longleftarrow \emptyset$;
$Nb \longleftarrow Nb \cup \{n\}$;
for $i \leftarrow 1$ **to** $|S.Threads|$ **do**
  $t \leftarrow S.Threads[i]$;
  for $nb \in Nb$ **do**
    $ta \leftarrow nb.ThreadAssignments\_node[i]$;
    $A1 \longleftarrow f\_Identify\_Agents\_ResidentIn(ta)$;
    $A2 \longleftarrow f\_Identify\_Agents\_ReceivedBy(ta)$;
    for $m0 \in ta.macroTask\_ofThread$ **do**
      $tm0 \leftarrow m0.TemporalMacroTasks.Last()$;
      $tm0.LegalAssignments \longleftarrow$
      $f\_Select\_Efficient\_Agents(m0, t, A1, A2)$;
    end
    $M \longleftarrow ta.macroTask\_ofThread$;
    $C1 \longleftarrow f\_Form\_EfficientCoalitions(M, A1, A2)$;
    $C2 \longleftarrow f\_Purify\_Coalitions(C1, M)$;
    $C3 \longleftarrow f\_Select\_Coalitions(C2, p)$;
    for $j \leftarrow 1$ **to** $|C3|$ **do**
      $Na \longleftarrow$
      $Na \cup \{f\_Generate\_NewNode(C3[j], nb)\}$;
    end
  end
  $Nb \longleftarrow Na$;
  $Na \longleftarrow \emptyset$;
end
$N \longleftarrow Nb$;

Algorithm 1: Automated algorithm for calculation of finite set of feasible alternatives for making a strategic decision

For each node, this thread first extracts LoTeM tasks associated with this thread. Then, it calculates a set of efficient coalitions using all the available agents (previously assigned agents and recently received agents). An efficient coalition is composed of a subset of available agents that can provide all the capabilities required by this thread considering the strategy definition, state of the tasks, capabilities of the agents, and the problem domain. A new node is generated for each coalition to show what agents are allocated to this thread and what agents are going to be released into a lower thread. It is important to remember that the state of the tasks does not change and all decisions taken are related to a snapshot. A node is modeled by the "stateNode" class of the data model presented in Fig. 4.

Algorithm 2 is an autonomous algorithm that continuously monitors the task states to detect the proper time when a subset of assigned agents should be released from their threads. In real-time, this algorithm continuously observes and monitors the task environment that changes

Data: n: an entity of the "stateNode" class of the data model.
Data: S: an entity of the "strategy" class.
Data: D: the problem domain.
Result: $n.ThreadAssignment\_node$: the updated attribute of the $n0$.

repeat
  for $i \leftarrow 1$ **to** $|S.Threads|$ **do**
    $t \leftarrow S.Threads[i]$;
    $ta \leftarrow n.ThreadAssignments\_node[i]$;
    $ta.MacroTasks\_ofThread \longleftarrow f\_Calculate\_MacroTasks(n.Segments\_node, t, D)$;
  end
  $A3 \longleftarrow \emptyset$;
  for $i \leftarrow 1$ **to** $|S.Threads|$ **do**
    $t \leftarrow S.Threads[i]$;
    $ta \leftarrow n.ThreadAssignments\_node[i]$;
    $A1 \longleftarrow ta.AgentIds\_assigned$;
    for $m0 \in ta.macroTask\_ofThread$ **do**
      $tm0 \leftarrow m0.TemporalMacroTasks.Last()$;
      $A1 \longleftarrow A1 - f\_Select\_Efficient\_Agents(m0, t, A1, \{\})$;
    end
    $A3 \longleftarrow A3 \cup A1$;
  end
  if $|A3| = 0$ **then**
    $f\_WaitForaWhile()$;
  end
  else
    $TA \longleftarrow n.ThreadAssignment\_node$;
    $f\_Releas\_Agents(TA, A3)$;
    break;
  end
until *the global objective is achieved*;

Algorithm 2: Autonomous algorithm for adaption of a strategic decision

over time. It aims to identify, in a timely manner, a subset of assigned agents that can no longer provide any efficient capabilities for their threads. This algorithm sends these unrequired agents to a lower prioritized thread in the identified time. A release time indicates that the strategic decision should be refined. This process results in re-executing Algorithm 1.

To develop these algorithms, the first step is to formulate the problem. The SAP data model presents elements of the problem, properties, relationships, and interaction among these elements with regard to the problem data modeling [13]. This data model is used to support the development of the proposed algorithm. Fig. 4 shows a part of the UML class diagram of the data model that is used in this paper.

The following subsections are dedicated to the description of the sub-algorithms.

A. Calculate Macro Tasks

The purpose of Algorithm 3 is to extract the LoTeM tasks that are associated with a definite thread. This algorithm integrates the thread definition with the segments-located macro tasks to select a subset of segment-based macro tasks that are located in the thread. It classifies the selected tasks according to the task types of the problem domain and aggregates tasks of each group to calculate a new macro task that is associated with this thread. Table VII shows the achieved result from the execution of this algorithm.

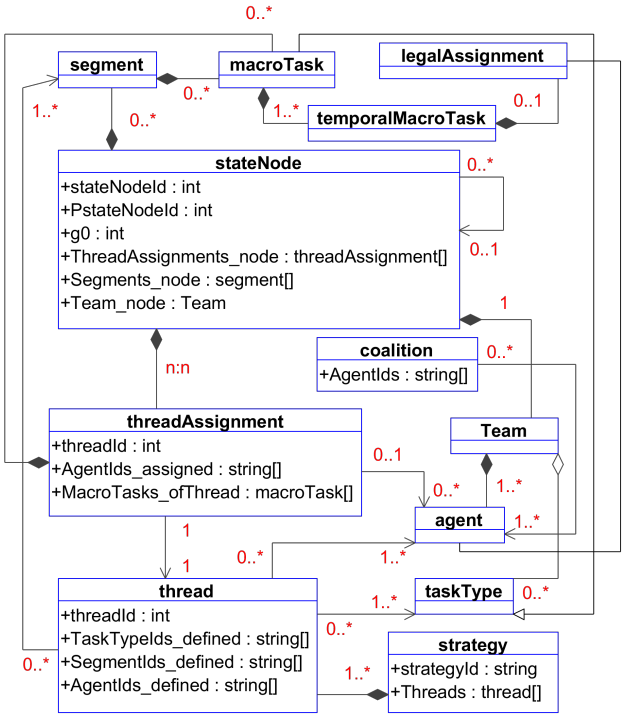

Figure 4: Part of the SAP data model that is used for the problem formulation [13]

B. Select Efficient Agents for a Macro Task

This algorithm aims to identify a subset of available agents that are efficient for a macro task of a specific thread. Two objectives are identified: 1) define whether an available agent is allowed to be assigned to a thread, and 2) calculate a maximum ability that an agent can provide for a macro task. Algorithm 4 shows that this algorithm considers three criteria for the selection of an agent.

Tasks require capabilities to be completed and agents provide capabilities. An agent is efficient for a task if he provides any capabilities required by that task. To simplify this algorithm, we assume that each task type requires a specific capability and agents provide different capabilities. This assumption ignores tasks that may require simultaneous capabilities. This algorithm applies both Enabled and Not Yet Enabled tasks in the computation.

C. Form Efficient Coalitions for a Thread

Algorithm 5 aims to calculate a set of efficient coalitions for a specific thread. An efficient coalition is a subset of efficient agents that can provide all the capabilities required for this thread. An efficient coalition has the ability to perform all the tasks contained by this thread. A coalition is a candidate whose agents will be assigned to the thread. In coalition formation theory, performance of a coalition is a function of the capabilities of the agents and quality of cooperation among the agents.

This algorithm engages all efficient and previously assigned agents in this process. If a coalition is not found, the algorithm selects a coalition that contains all the efficient agents; however, this coalition will not fully meet the capabilities that the thread requires to complete the

n0.Segments_node, t, D
Data: S: a set of "segment" elements.
Data: t: an entity of the "thread" class that presents a thread in the human strategy.
Data: D: a set of "taskType" elements in the problem domain.
Result: M: a set of "macroTask" elements that will be extracted for t.

```
begin
    M ⟵ ∅;
    for d0 ∈ D do
        m0 ← f_Initialize_aNew_macroTask();
        m0.taskType ← d0;
        m0.geoObjectId ← t.threadId;
        m0.TemporalMacroTasks ⟵
        {f_Initialize_aNew_temporalMacroTask()};
        M ⟵ M ∪ {m0};
    end
    for s0 ∈ S do
        for m0 ∈ s0.MacroTasks_ofSegment do
            if m0.taskType.taskTypeId ∈
            t.TaskTypeIds_defined and
            s0.segmentId ∈ t.segmentIds_defined then
                for i ← 1 to |M| do
                    if M[i].taskType = m0.taskType
                    then
                        c0 ←
                        |m0.TemporalMacroTasks|;
                        tm0 ←
                        m0.TemporalMacroTasks[c0];
                        tm2 ←
                        M[i].TemporalMacroTasks[0];
                        tm2.disenabledAmount ←
                        tm2.disenabledAmount +
                        tm0.disenabledAmount;
                        tm2.enabledAmount ←
                        tm2.enabledAmount +
                        tm0.enabledAmount;
                    end
                end
            end
        end
    end
end
```

Algorithm 3: Extract the set of macro tasks contained by a thread.

TABLE VII.: Temporal Macro Tasks associated with the Three Threads in the Simulated USAR Scenario

Location (Thread)	Task-Type	Not Yet Enabled Number	Enabled Number
Thread 1	Reconnaissance	0	25
Thread 1	Search	12	23
Thread 2	Light Rescue	17	16
Thread 3	Reconnaissance	0	25
Thread 3	Search	0	5
Thread 3	Light Rescue	10	9

task. We assume that each macro task requires one type of capability.

D. Purify Coalitions

Algorithm 6 aims to purify a set of coalitions. It is important for the system to find and eliminate redundant coalitions. The role of two coalitions formed by different agents is the same for a thread if they provide the same capabilities for the thread. Redundant coalitions cause a loop problem in search algorithms.

The central part of this algorithm is an ability matrix. This matrix is used to present a matrix of the total capabilities that all the coalitions provide for all the macro

Data: t: an entity of the "thread" class that presents a thread.
Data: $A1$: a subset of agents assigned to t.
Data: $A2$: a subset of agents released/entered into t.
Data: m: an entity of the "macroTask" class.
Result: LA: a subset of entities of the "legalAssignment" class to encode agents selected for the m.

```
LA ⟵ ∅;
tm0 ← m.TemporalMacroTasks.Last();
tm0.LegalAssignments ⟵ ∅;
for a0 ∈ A1 ∪ A2 do
    if a0.agentId ∈ t.AgentIds_defined then
        if tm0.disenabledAmount > 0 or
        tm0.enabledAmount > 0 then
            x0 ← 0;
            for C0 ∈ a0.agentType5.AgentCaps do
                c2 ← m.taskType5.CapTypeIds[0];
                n ← C0.CapTypeIds.CountOf(c2);
                if x0 < n ∗ aC0.speed then
                    x0 ← n ∗ aC0.speed;
                end
            end
            if x0 > 0 then
                lA0 ⟵
                f_Initialize_aNew_legalAssignment();
                lA0.agentId ← agent.agentId;
                lA0.maxAbility ← x0;
                LA ⟵ LA ∪ {lA0};
            end
        end
    end
end
```

Algorithm 4: Select efficient agents for a macro task associated with a thread

tasks. Rows indicate macro tasks and columns indicate coalitions. A number in a cell shows the total capability that the relevant collation can provide for the relevant macro task. Finally, a number representing the weighted total ability is calculated for each collation.

E. Select Coalitions

This simple algorithm aims to choose a number of coalitions from the purified coalitions for a specific thread according to a selection method. We consider two options: 1) sort the coalitions according to their weighted total ability and then select the first and last that represent the minimum and maximum ability, and 2) select all available agents.

F. Generate a New Node

For a certain thread, a selected coalition defines what agents should be assigned to this thread and what agents should leave the thread. A number of coalitions may be calculated and each one presents a feasible alternative for an appropriate assignment of available agents to this thread. A new node is produced for each coalition and agents of this coalition are assigned to the related thread. Unwanted agents are assigned to the next thread. This algorithm updates the "ThreadAssignments_node" attribute of the node.

VI. Implementation and Evaluation

The presented algorithms were implemented in GI-Coordinator. GICoordinator is a spatial intelligent agent

```
Data: M: the set of "macroTask" elements associated with a
      specific thread.
Data: A1: a subset of agents assigned to this thread.
Data: A2: a subset of agents released/entered into this thread.
Result: C: a set of "coalition" elements.
begin
    C ⟵ ∅;
    C0 ⟵ ∅;
    A1 ⟵ A1 ∩ f_Read_EfficientAgents(M);
    A2 ⟵ A2 ∩ f_Read_EfficientAgents(M);
    P ⟵ f_Power(A2);
    for p0 ∈ P do
        p0 ⟵ p0 ∪ A1;
        C0 ⟵ C0 ∪ {p0};
    end
    for c0 ∈ C0 do
        uc0 ⟵ true;
        for m0 ∈ M do
            um0 ⟵ false;
            tm0 ← m0.TemporalMacroTasks.last();
            for lA0 ∈ tm0.LegalAssignments do
                if lA0.agentId ∈ c0 then
                    um0 ⟵ true ;
                    exit for;
                end
            end
            if um0 = false then
                uc0 ⟵ false;
                exit for;
            end
        end
        if uc0 = true then
            C ⟵ C ∪ {c0};
        end
    end
    if C = ∅ then
        c2 ⟵ f_Find_biggest_c0In(C0);
        C ⟵ {c2};
    end
end
```

Algorithm 5: Form efficient coalitions for a thread

```
Data: M: the set of "macroTask" elements.
Data: C: the set of "coalition" elements.
Result: C2: a subset of C that are purified.
begin
    C2 ⟵ C;
    for c0 ← 1 to |C| do
        t0 ← 0;
        for r0 ← 1 to |M| do
            S[r0][c0] ← 0;
            m0 ⟵ M[r0];
            tm0 ← m0.TemporalMacroTasks[0];
            for lA0 ∈ tm0.LegalAssignments do
                if lA0.agentId ∈ C[c0].AgentIds then
                    S[r0][c0] ←
                    S[r0][c0] + lA0.maxAbility;
                end
            end
            t0 ← t0 + S[r0][c0];
        end
        C[c0].totalAbality ← t0 * |C[c0].AgentIds|;
    end
    for c0 ← 1 to |C| do
        for c2 ← c0 + 1 to |C| do
            Eq0 ← true;
            for r0 ← 1 to |M| do
                if S[r0][c0] ≠ S[r0][c2] then
                    Eq0 ← false;
                    exit for
                end
            end
            if Eq0 = true then
                exit for
            end
        end
        if Eq0 = false then
            C2 ⟵ C2 ∪ {C[c0]};
        end
    end
end
```

Algorithm 6: Purify coalitions

that assists the IC and supports human decisions in the coordination of a team of field units. It is used in crisis response management, especially in urban search & rescue operations [14]. This spatial intelligent system supports the IC with intelligent algorithms for action planning and task scheduling for the centralized coordination of a team in a dynamic and spatial environment [15]. It applies a spatial database for geographic and location-based information management and uses GIS functions to support development of these spatial intelligent algorithms [13], [14]. The C#.Net programming language was used to implement the core of GICoordinator. An IC is equipped with a computer that runs an instance of GICoordinator. A simulator that runs multiple instances of the GICoordinator was developed for the evaluation of distributed algorithms for decentralized coordination [16].

To better understand these two algorithms, they were executed for the scenario described in Section II. Furthermore, we tested the automated algorithm on various human strategies and with different numbers of agents to calculate the run time and number of generated nodes. The results are presented in the next section.

VII. Result

A. Automated Calculation of Feasible Alternatives

Figure 5 presents the expanded/generated decision tree from the automated algorithm using a state node in which four free agents have been sent to thread 1 by the IC at time 0 in the described USAR scenario. Table VIII shows the result of this algorithm that represents a collection of ten alternative scenarios presented for the IC to support and assist him in selecting a choice for a strategic agent coordination decision. Ten new nodes were generated and each one presents a different assignment of agents to the threads.

TABLE VIII.: Set of Alternatives Calculated by Automated Algorithm for Making a Strategic Decision at Time 0

Choice No.	Assignment to Thread 1	Assignment to Thread 2	Assignment to Thread 3
1	a2	a7	a0, a6
2	a2	a6, a7	a0
3	a0, a7	a6	a2
4	a2, a7	a6	a0
5	a0, a2	a7	a6
6	a0, a2	a6, a7	
7	a0, a2, a6		a2
8	a2, a6, a7		a0
9	a0, a2, a7	a6	
10	a0, a2, a6, a7		

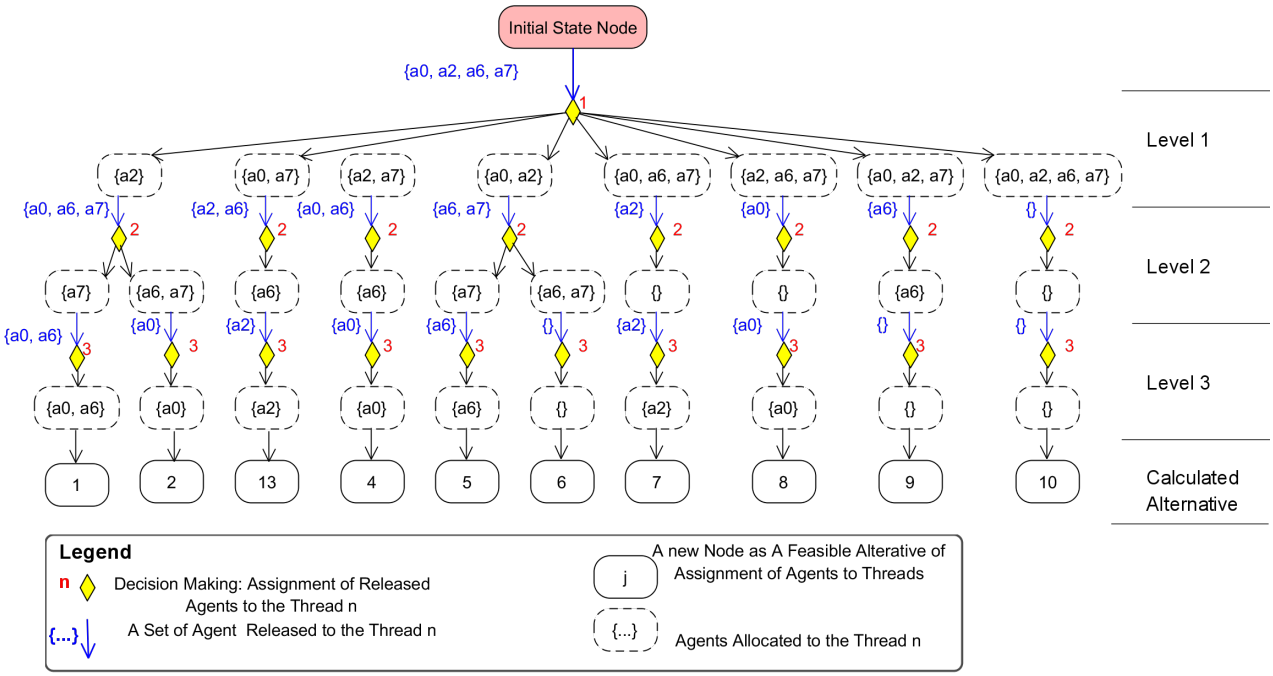

Figure 5: Expand the decision tree from State Node 0 using the automated algorithm

B. Autonomous Adaption of a Strategic Decision

Assume that the sixth alternative was selected by the IC or by an intelligent assistant system. Table IX shows this strategic decision made at time 0. The information regarding this decision was sent to the four agents distributed in the operational area. According to this decision, the actions of agents {a0, a2} are limited to reconnaissance and search tasks spatially located/distributed at the proximity of the three roads {s3, s4, s5}. Furthermore, according to this decision, the actions of agents {a6, a7} will be restricted to light rescue tasks located in the same geographic area. This mission started at time 0.

We have simulated the problem using a suitable approach [15]. An effective algorithm was used to dynamically assign the LoTeM tasks to the agents in the multi-agent scheduling process.

TABLE IX.: First Strategic Decision, the Sixth Alternative at Time 0, for Coordination and Control of Agents

Thread Id	Assigned Agents	Start Time
1	a0, a2	0
2	a6, a7	0
3		0

Furthermore, assume that during the emergency response operations, new data are reported to the operation center from the agents. This center gathers and integrates the data to continuously/temporally provide a situational awareness in a timely manner as an overall picture for the IC. The IC is required to adapt the current strategic decision to the new situation at a proper time. Therefore, he must scan and monitor the state of the environment. At time 96, Table X shows the updated information of the LoTem tasks.

TABLE X.: Updated State of Twelve Segment-based Temporal Macro Tasks at Time 96 using GICoordinator [15].

No.	Location (Road S.)	Task-Type	Not Yet Enabled Amount	Enabled Amount
1	s1	T0	0	25
2	s1	T2	0	4
3	s2	T1	0	5
4	s2	T2	10	5
5	s3	T0	0	*0*
6	s3	T1	*0*	*8*
7	s3	T2	2	*5*
8	s5	T0	0	*0*
9	s5	T1	*0*	*7*
10	s5	T2	5	*3*
11	s4	T1	0	*9*
12	s4	T2	*5*	*7*

The autonomous algorithm revealed that time 96 is the correct time for adaption of the strategic decision. Based on the strategic decision and Table X, there is no available task that agent a0 can perform; other agents must continue their tasks. The result was to release the agent a0 from thread 1 and assign him to thread 2. This action triggers the automated algorithms that re-calculate a feasible alternative as shown in Fig. 6.

C. Evaluation of Efficiency of the Automated Algorithm

To evaluate the running time of the automated algorithm, we executed it with the same tasks with three human strategies and different team sizes. Strategy I includes three threads that are agent independent. Strategy II includes three semi-independent threads. Strategy III is more complex with three threads where all the agents are defined for all the threads.

Table XI shows the results of this test. The result achieved for each scenario includes two parameters: 1)

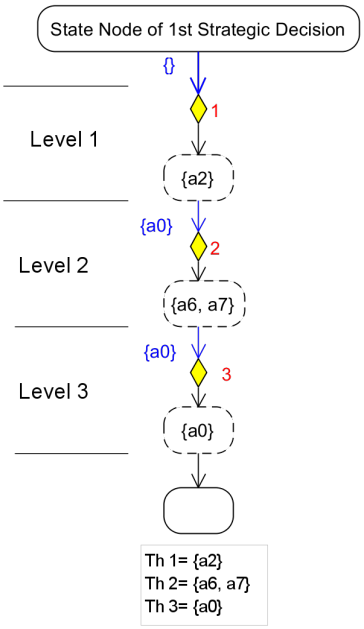

Figure 6: Autonomous adaption of the first strategic decision at time 96 by releasing agent a0 from thread 1 into thread 2 and the automated calculation of a feasible alternative for this adaption decision.

TABLE XI.: Evaluation of Running Time (in milisecond) of the Automated Algorithm

Team Size	Strategy	All Coalitions		Min-Max Coalitions	
		Run Time	Nodes	Run Time	Nodes
4	I	27	1	31	1
8	I	44	16	38	8
12	I	51	81	43	8
4	II	29	2	28	2
8	II	42	28	41	6
12	II	101	171	41	6
4	III	51	10	38	3
8	III	106	172	40	4
12	III	922	1548	117	4

the running time, and 2) number of alternatives that were calculated for this scenario.

..

VIII. Conclusion

This paper presented an intelligent algorithm composed of two sub-algorithms for 1) automated calculation of feasible alternative scenarios for selecting a strategic decision at a specific time, and 2) autonomously detecting the correct time at which this strategic decision should be refined by releasing a set of identified agents from their threads. This algorithm autonomously controls what agents should be released from their thread within a strategic decision. The role of the human is to select an alternative as a strategic decision.

A strategic decision constrains agents to threads; however, it does not assign tasks to agents. A strategic decision controls the domain of activities of the agents.

A coalition, which is a set of agents, is sufficient for a thread if it provides all the capabilities required by all the tasks (Enabled and Not Yet Enabled tasks) located in this thread. A question that arises is: What coalition among the available coalitions can accomplish this thread in the least time? This question will be addressed in a future work.

Many feasible alternatives are found for a case that contains a large team and a complex strategy. The selection of the two coalitions that provide the minimum and maximum capabilities for a thread reduce the run time and number of generated alternatives.

The value of this algorithm is to assist a human and collaborate with him in the calculation of a centralized multi-agent coordination. This issue is essential, especially in a situation where the human strategy defines agents for several threads.

This algorithm can be used in the multi-agent planning problem. An action plan made by the IC is a sequence of strategic decisions that states how a team of agents can achieve the goal.

In the future, we will investigate a search-based algorithm that aims to determine an optimal decision by selecting the best choice among available alternatives.

Acknowledgment

R.N. is grateful for the financial support of GCOE-HSE of Kyoto University that enabled him to be a visiting scholar at the Information Sciences Institute of University of Southern California and the Robotics Institute of the Carnegie Mellon University from December 2011 to November 2012.

References

[1] Barbulescu, Laura, Rubinstein, Zachary B., Smith, Stephen F., and Zimmerman, Terry L. "Distributed coordination of mobile agent teams: the advantage of planning ahead." In Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1, pp. 1331-1338. International Foundation for Autonomous Agents and Multiagent Systems, 2010.

[2] Bigley, Gregory A. and Roberts, Karlene H. "The incident command system: High-reliability organizing for complex and volatile task environments." Academy of Management Journal 44, no. 6 (2001): 1281-1299.

[3] Burstein, Mark H. and McDermott, Drew V. "Issues in the development of human-computer mixed-initiative planning." Advances in Psychology 113 (1996): 285-303.

[4] Chen, Rui, Sharman, Raj, Rao, H. Raghav, and Upadhyaya, Shambhu J. "Coordination in emergency response management." Communications of the ACM 51, no. 5 (2008): 66-73.

[5] Fiedrich, Frank, Gehbauer, Fritz, and Rickers, U. "Optimized resource allocation for emergency response after earthquake disasters." Safety Science 35, no. 1 (2000): 41-57.

[6] Hunger, J. David and Wheelen, Thomas L. Essentials of strategic management. New Jersey: Prentice Hall, 2003.

[7] INSARAG Guidelines and Methodology Manual, United Nations Office for the Coordination of Humanitarian Affairs, 2004, `http://www.usar.nl/upload/docs/insarag_guidelines_july_2006.pdf` (accessed July 2013).

[8] Kitano, Hiroaki and Tadokoro, Satoshi. "Robocup rescue: A grand challenge for multiagent and intelligent systems." AI Magazine 22, no. 1 (2001): 39.

[9] Maheswaran, Rajiv T., Szekely, Pedro, Becker, Marcel, Fitzpatrick, Stephen, Gati, Gergely, Jin, Jing, Neches, Robert et al. "Predictability & criticality metrics for coordination in complex environments." In Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2, pp. 647-654. International Foundation for Autonomous Agents and Multiagent Systems, 2008.
[10] Maheswaran, Rajiv T., Szekely, Pedro, and Sanchez, Romeo. "Automated adaptation of strategic guidance in multiagent coordination." In Agents in Principle, Agents in Practice, pp. 247-262. Springer Berlin Heidelberg, 2011.
[11] Malone, Thomas W. and Crowston, Kevin. "The interdisciplinary study of coordination." ACM Computing Surveys (CSUR) 26, no. 1 (1994): 87-119.
[12] Nourjou, Reza, Hatayama, Michinori, and Tatano, Hirokazu. "Introduction to spatially distributed intelligent assistant agents for coordination of human-agent teams actions." In Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on, pp. 251-258. IEEE, 2011.
[13] Nourjou, Reza, Szekely, Pedro, Hatayama , Michinori, Ghafory-Ashtiany, Mohsen, and Smith, Stephen F. "Data Model of the Strategic Action Planning and Scheduling Problem in a Disaster Response Team." Journal of Disaster Research 9, no. 3 (2014)
[14] Nourjou, Reza, Hatayama, Michinori, Smith, Stephen F., Sadeghi, Atabak, and Szekely, Pedro. "Design of a GIS-based Assistant Software Agent for the Incident Commander to Coordinate Emergency Response Operations." arXiv preprint arXiv: 1401.0282 (2014).
[15] Nourjou, Reza, Smith, Stephen F., Hatayama, Michinori, Okada, Norio, and Szekely, Pedro. "Dynamic Assignment of Geospatial-Temporal Macro Tasks to Agents under Human Strategic Decisions for Centralized Scheduling in Multi-agent Systems." International Journal of Machine Learning and Computing (IJMLC) 4, no. 1 (2014): 39-46.
[16] Nourjou, Reza and Hatayama, Michinori. "Simulation of an Organization of Spatial Intelligent Agents in the Visual C#.NET Framework." International Journal of Computer Theory and Engineering, IJCTE 2014 Vol.6(5), 2014.
[17] Nwana, Hyacinth S., Lee, Lyndon C., and Jennings, Nicholas R. "Coordination in software agent systems." British Telecom Technical Journal 14, no. 4 (1996): 79-88.
[18] Schurr, Nathan, Marecki, Janusz, Lewis, John P., Tambe, Milind, and Scerri, Paul. "The defacto system: Training tool for incident commanders." In AAAI, pp. 1555-1562. 2005.
[19] Vafaeinezhad, Ali Reza, Alesheikh, Ali Asghar, Hamrah, Majid, Nourjou, Reza, and Shad, Rouzbeh. "Using GIS to Develop an Efficient Spatio-temporal Task Allocation Algorithm to Human Groups in an Entirely Dynamic Environment Case Study: Earthquake Rescue Teams." In Computational Science and Its Applications ICCSA 2009, pp. 66-78. Springer Berlin Heidelberg, 2009.

Reza Nourjou was born in Urmia, Iran, in 1979. He received the B.Sc. degree in geomatics engineering and the M.Sc. degree in geographic information systems (GIS) from the K.N.Toosi University of Technology, Iran, in 2002 and 2007, respectively. In 2006, he joined the Risk Management Research Center, International Institute of Earthquake Engineering and Seismology (IIEES), Tehran, Iran as a GIS expert and a research assistant. Since 2010, he has been a Ph.D. candidate in graduate school of Informatics, Kyoto University, Japan.

His current research interests include Multi-agent Systems, Automated Planning and Scheduling, Intelligent Software Agents, Artificial Intelligence, GIS, Geodatabase, Spatial Agent-based Modeling and Simulation, System Design & Analyst. He is a member of Association for the Advancement of Artificial Intelligence.

He was the recipient of the first prize in the GIS competition ranked first in the national student competition, Iran, in 2001. He received the Japanese Government Scholarship from October 2009 to March 2013 for the Ph.D. program. He was a visiting scholar at the Information Sciences Institute of the University of Southern California and the Robotics Institute of Carnegie Mellon University from November 2011 to November 2012.

Michinori Hatayama was born in Osaka, Japan, in 1968. He received the B.S. and M.S. degrees in Control Engineering from Osaka University in 1992 and 1994 respectively, and the Ph.D. degree in Computational Intelligence and Systems Science from Tokyo Institute of Technology in 2000. In 2002 he became an Assistant Professor and from 2004 is an Associate Professor in Disaster Prevention Research Institute, Kyoto University, Kyoto, Japan.

His research carrier in disaster risk management area began from the Great Hanshin-Awaji Earthquake in Kobe in 1995. After that, his group developed an original temporal GIS named DiMSIS and applied it to disaster response activity and disaster risk management in some local governments and regional communities. This software was used in real sites by officials in Kobe (1995), Turkey (1999 and 2000), Niigata (2004) and Tohoku (2011).

His current research interests include ICT based disaster risk assessment system, disaster response system, rescue activity support, and disaster planning support system.