Agent-Based Dynamic Scheduling for Earth-Observing Tasks on Multiple Airships in Emergency

Xiaomin Zhu, *Member, IEEE*, Kwang Mong Sim, *Senior Member, IEEE*, Jianqing Jiang, Jianjiang Wang, Chao Chen, and Zhong Liu

***Abstract*—Airships are becoming promising platforms for Earth observing in emergency (e.g., nature disaster surveillance). Dynamic scheduling plays a very critical role in dealing with emergent tasks. In this paper, we devise a novel agent-based scheduling mechanism. In contrast to the traditional contract net protocol, our mechanism has a bidirectional announcement mechanism, and the collaborative process consists of forward announcements from the perspective of tasks and backward announcements from the perspective of resources to jointly accomplish the scheduling. Additionally, we devise calculation rules of the bidding values in both forward and backward announcements and two heuristics for selecting contractors. Based on the bidirectional announcement mechanism, we propose an agent-based dynamic scheduling (ABDS) strategy for scheduling Earth-observing tasks on multiple airships. The ABDS scheme employs the fair competition principle of a roulette wheel and the dynamic adjustment principle of a buffer pool to solve the problem of local searching and to improve the load balancing of resources. Extensive experiments were carried out to evaluate the performance of ABDS by comparing it with a unidirectional announcement (UA) scheduling algorithm and a genetic algorithm. In addition, the sensitivity of the priority parameter to the system performance is evaluated. Experimental results show that ABDS significantly outperforms the scheduling quality of UA and that it is suitable for the Earth-observing task scheduling on multiple airships in emergency.**

***Index Terms*—Agent-based system, airship, bidirectional announcement mechanism, buffer pool, contract net protocol (CNP), Earth-observing, roulette wheel, scheduling, task allocation.**

I. Introduction

THE airships, which rely on buoyant gases to maintain altitude, are becoming efficient means for Earth-observing. This is partially due to the ever-increasing requirement for atmospheric platforms with the characteristic of long-term presence within a particular altitude range or geographic perimeter [1]. The advantages of the airship for high-altitude/long endurance missions have generated substantial interests in both commercial and military sectors (e.g., aerial exploration, environmental monitoring, and surveillance purposes) [2]. Compared with unmanned aerial vehicles (UAVs), the airships have longer navigation period and can float around a fixed position for a longer duration to observe a target continuously. In comparison with satellites, the airships can be flexibly controlled and have relatively small cost in terms of manufacturing and operations. In addition, since satellites orbit the Earth, targets can only be observed in the satellites' visible scopes, i.e., task execution has inherent visible time window constraints [3]. However, the airships do not have the inescapable time window constraints of satellites. Consequently, the airships play a critical role in a multiplatform Earth-observing system. In particular, Earth-observing in emergency by airships has become a critical measure for getting the firsthand information. For example, when an earthquake occurs, the images of stricken areas can be obtained by airships. More importantly, the images are expected to be acquired within a few minutes or persistently for timely conducting damage assessment and planning rescue policies. Owing to airship's strength with respect to area covering, quick response, and long-term surveillance, airships can efficiently satisfy the Earth-observing requirements in emergency.

Efficient scheduling, in distributed systems, is crucial for achieving high performance of applications, e.g., improving tasks' acceptance rate and reducing tasks' finish time, etc. [4]. The Earth-observing by multiple airships substantially belongs to an application in distributed systems. In the scheduling, tasks submitted by different users will be allocated to airships and, thus, to generate schedule decision satisfying some constraints. It should be noted that the tasks in emergency may massively arrive within a short time interval and often have high quick-response requirements, e.g., if a task cannot be finished within a given time period or deadline, the Earth-observing result might be useless for users. Thus, a task that cannot be finished before its deadline will not be allocated while scheduling. Usually, the airships are equipped with multiple optical sensors for Earth-observing, and the targets for observing are the tasks submitted by users. The scheduling objective is to allocate tasks to airships to satisfy the users' requirements in sensor sort, timing, resolution, and so on.

To date, a wide variety of scheduling algorithms have been developed to provide high performance for distributed systems

Manuscript received June 26, 2013; revised December 24, 2013 and March 17, 2014; accepted May 20, 2014. This work was supported by the National Nature Science Foundation of China under Grant 61104180 and Grant 71271216.

X. Zhu, J. Jiang, C. Chen, and Z. Liu are with the College of Information Systems and Management, National University of Defense Technology, Changsha 410073, China (e-mail: xmzhu@nudt.edu.cn; poyan_soho@sohu.com; chenchao@nudt.edu.cn; liuzhong@nudt.edu.cn).

K. M. Sim is with the School of Computing, University of Kent, Kent CT2 7NF, U.K. (e-mail: prof_sim_2002@yahoo.com).

J. Wang is with the College of Information Systems and Management, National University of Defense Technology, Changsha 410073, China, and also with the Faculty of Economics and Business, Katholieke Universiteit Leuven, 3000 Leuven, Belgium (e-mail: jianjiangwang@nudt.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSYST.2014.2327069

in many fields. However, our study is out of the ordinary. While the majority of the algorithms are extensions or variants of these intelligent algorithms, our work deviates from the algorithms in the literature by designing and implementing a novel scheduling mechanism based on an intelligent agent approach and then develops a corresponding dynamic scheduling algorithm for real-time tasks executed by airships.

Contributions: The major contributions of this work are summarized as follows.

1) We designed a bidirectional announcement mechanism based on an improved contract net protocol (CNP).
2) We constructed a multiobjective constraint satisfaction optimization model.
3) We developed an agent-based dynamic scheduling algorithm or ABDS for independent real-time tasks executed by multiple airships.
4) We investigated fair competition principle of roulette wheel and a dynamic adjustment principle of buffer pool used for our ABDS.
5) We conducted experimental evaluation of ABDS by simulations.

Although in this paper we only focus on demonstrating the effectiveness of our approach in task scheduling for airships, our agent-based scheduling algorithms, announcement mechanisms, and heuristics strategies can be also generalized and applied to other domains involving distributed processing systems in which two or more programmable devices are connected so that information can be exchanged [5], such as grid and cloud.

The rest of this paper is organized as follows. The related work in the literature is summarized in Section II. Section III defines the problem. Section IV formally models the agent-based scheduling problem for Earth-observing tasks in emergency. The ABDS algorithm is described in detail in Section V. Simulation experiments and performance analysis are given in Section VI. Section VII concludes this paper with a summary and future directions.

II. Related Work

Usually, satellites, UAVs, and airships are the main platforms for Earth-observing. Correspondingly, there exist many scheduling strategies or algorithms for Earth-observing tasks on these platforms. Specifically, static satellite scheduling were extensively investigated, including single-satellite scheduling and multiple-satellite scheduling, e.g., [6]–[8]. In recent years, more attention has been drawn toward the dynamic scheduling for Earth-observing tasks on satellites, e.g., [9]–[11]. Regarding the Earth-observing task scheduling on UAVs, the main focus is the collaborative planning on multiple UAVs, e.g., [12]–[14]. For the Earth-observing task scheduling on airships, there are a few investigations about scheduling when airships are spot hovering, e.g., [15] and [16]. Due to the unique feature of airships aforementioned, multiple airships' Earth-observing in a cooperation manner has become a promising research. In this paper, we concentrate on designing a novel scheduling mechanism and corresponding scheduling algorithms and applying them into airships' Earth-observing. In addition, our work is used to solve the scheduling problem for aperiodic, independent, and real-time Earth-observing tasks.

So far as scheduling algorithms are concentrated, they can be classified into three categories: simulation based, artificial intelligence based, and agent based [17]. Meanwhile, the agent-based scheduling approaches are particularly attractive when designing scalable systems due to being inherently decentralized. Agents make decisions based on local interactions, and good agent-based scheduling algorithms allow them to adapt, enabling them to coordinate through self-organization [18]. Moreover, the agent-based scheduling has the ability to effectively handle the dynamic nature in emergency. Agents directly represent physical objects, e.g., machines, tasks, and operators [19]. Hence, the agent-based scheduling can effectively realize dynamic scheduling of Earth-observing tasks on multiple airships in emergency.

One type of agent-based scheduling algorithm is threshold algorithms that are developed from the threshold model of task allocation in insect colonies. The tasks in this model are classified into a set of types. An individual chooses to engage in a task by probabilistically comparing some environmental stimulus for the task with some internal preference for the task's type [18]. For example, Campos *et al.* investigated the dynamic scheduling and division of labors in social insects [20]. Janacik *et al.* proposed an emergent topology control mechanism based on division of labor in ants [21]. Price evaluated the adaptive nature inspired task allocation against decentralized multiagent strategies [22]. However, this kind of intelligence approaches exhibit highly variable computation time; hence, adopting their worst case execution time perhaps results in an unacceptable underutilization [23].

Another successful type of agent-based scheduling algorithm is based on the combination of market-based mechanisms, which have been used in a wide range of applications [24]. The most well-known market-based mechanism used by the multiagent systems is CNP [25], in which groups of individuals employ marketlike approaches, i.e., auction, to decide who realizes these goals, with bids based on the individual's desire and the ability to finish their goals. For example, Parunak attempted to employ distributed dynamic approach in manufacturing by deploying agent-based concept [26]. In this paper, CNP is used for assignment of jobs to machines providing dynamic allocation and natural load balancing. Macchiaroli and Riemma also proposed a price mechanism in CNP to assign parts to resources in flexible manufacturing systems [27]. Sousa and Ramos presented a negotiation protocol based on CNP for scheduling in a holonic manufacturing system [28]. Lau *et al.* suggested an agent-based supply chain model to support distributed scheduling, in which a modified CNP is proposed to enable more information sharing among enterprises [29]. Owliya *et al.* designed a CNP-based peer-to-peer model with specific rules and characteristics for scheduling and task allocation in manufacturing application [30]. Yen and Wu introduced a market-based control mechanism in which self-interested agents initiate or participate in auctions to sell or buy in Internet scheduling environment [31]. In addition, the CNP also has been adopted in job-shop scheduling due to its simplicity and intuitiveness. Meanwhile, job and machine are

represented by manager and contractor in CNP, respectively. For instances, Vancza and Markus applied market mechanism (based on job price) for resource allocation in job-shop scheduling [32]. Baker demonstrated the viability of CNP in a real job-shop environment [33]. In addition, Fischer *et al.* reported a successful application of CNP in transportation domain [34]. Recently, the research in this area has evolved to consider the combination of CNP with Lagrangian relaxation. For instance, Kutanoglu *et al.* addressed the combination issue of auction and Lagrangian relaxation for distributed resource scheduling [35].

In contrast to the aforementioned approaches, we investigated in this paper a novel agent-based scheduling strategy based on CNP to solve the task scheduling issue for airships in emergency. The strategy employs the bidirectional announcement mechanism (i.e., forward announcement and backward announcement), where the tasks and resources are both announcers and bidders. In addition, a roulette wheel and a buffer pool are employed to further improve the scheduling quality.

III. Problem Depictions

The Earth-observing targets are capable of being regarded as tasks needed to be executed from the view of airships. In emergency, the tasks commonly massively and aperiodically arrive during a short time interval with tight deadlines, and their arrival times are not known *a priori*. Since the multiple airships are equipped with different kinds of optical sensors, the users in practice may require specific sensors and resolutions [e.g., charge coupled device (CCD), 0.5 m]. In this paper, a mathematic model based on a multiobjective constraint satisfaction problem is established to solve the scheduling issue.

A. Assumptions and Notation

The scheduling problem investigated in this paper is based on the following assumptions.

1) The Earth-observing targets are all static point targets. The targets can be observed within an instant time slot by a sensor of an airship.
2) Every task is an atomic task, i.e., the tasks considered in this study cannot be partitioned while scheduling, and they cannot be preempted when they are running. In addition, one airship has the ability to execute one atomic task.
3) An airship is hovering in a spot when it is observing targets [36].

We consider a set $T = \{t_1, t_2, \ldots, t_n\}$ of tasks that are independent, nonpreemptive, aperiodic, and more importantly, with deadlines. For a given task $t_i \in T$, it can be modeled by $t_i = (a_i, d_i, \mathrm{nd}_i, p_i)$, where a_i, d_i, nd_i, and p_i represent t_i's arrival time, deadline, needed duration, and priority, respectively. Note that, before a task is submitted, its priority was set by a user on a task template or was atomically produced by a software in the system. The resources in this paper are the sensors of airships to take photographs of targets. Let $R = \{r_1, r_2, \ldots, r_m\}$ be a resource set. When a task arrives, the following matrices can be determined by our scheduling algorithm. In addition, these matrices are essential for recording the scheduling information to optimize our objectives. $S = (s_{ij})_{n\times m}$ is a start time matrix, where element s_{ij} denotes the start time of task t_i executed by resource r_j. Similarly, $F = (f_{ij})_{n\times m}$ is a finish time matrix, where element f_{ij} represents the finish time of task t_i by resource r_j. If the start time and the finish time of a task can be obtained, the algorithm can use them to determine which resource is better to allocate this task to. An allocation matrix $X = (x_{ij})_{n\times m}$ is used to reflect a mapping of n tasks to m resources. Element x_{ij} in X is "1" if task t_i is assigned to resource r_j; otherwise, x_{ij} is "0". In addition, the matrix X changes in time (i.e., X is changed when a new task is assigned). Let $\Psi = (\psi_{ij})_{n\times m}$ be a basic ability satisfaction matrix, where element $\psi_{ij} = 1$ represents that resource r_j has the ability to execute task t_i; else, $\psi_{ij} = 0$. When a task is executed, the airship may need to change its position or its sensor's orientation, in which some time is required, i.e., provision time. Thus, we let $\mathrm{PT} = (\mathrm{pt}_{ij})_{n\times m}$ denote the provision time matrix, where element pt_{ij} represents the provision time after finishing t_i's preceding task to execute task t_i by airship resource r_j.

B. Constraints

Since each task is neither splittable nor preemptive, a task can only be allocated to one airship resource, and the airship resource cannot be shared by another task if the two tasks have an overlap in execution. Therefore, we have the constraint C_1 as follows

$$C_1: \begin{cases} \sum_{j=1}^{m} x_{ij} = 0 \text{ or } 1 & i \in [1, n], \\ x_{kj} = 0 & \text{if } (x_{ij} = 1) \\ & \quad \wedge ([s_{kj}, f_{kj}] \cap [s_{ij}, f_{ij}] \neq \varnothing) \end{cases} \tag{1}$$

where s_{ij} and f_{ij} are the start time and the finish time of task t_i executed by r_j, respectively.

The airships may be equipped with different optical sensors (e.g., CCD, synthetic aperture radar, and infrared) that have different resolutions. In practice, the users can select specific sensors and probably require given resolutions. Thereby, the basic ability refers that an airship resource being selected to execute a task satisfies the needed sensor type constraint, resolution constraint, and availability constraint. The basic ability constraint C_2 is as follows:

$$C_2: \begin{cases} \psi_{ij} = 1 & \text{if } \psi_{ij}^1 = \psi_{ij}^2 = \psi_{ij}^3 = 1 \\ \psi_{ij} = 0 & \text{else} \end{cases} \tag{2}$$

where $\psi_{ij}^1 = 1$ represents that the sensor type satisfies the user's requirement, $\psi_{ij}^2 = 1$ means that the resolution satisfies the user's requirement, and $\psi_{ij}^3 = 1$ reflects that the airship sensor is currently available.

The start time s_{ij} of a task t_i is related to its arrival time a_i and the ready time of r_j, i.e.,

$$s_{ij} = \max\{a_i, f_{kj} + \mathrm{pt}_{ij}\} \tag{3}$$

where $f_{kj} + \mathrm{pt}_{ij}$ is the ready time of airship resource r_j to execute task t_i, and f_{kj} is the finish time of task t_i's preceding task t_k on r_j.

The finish time f_{ij} of a task t_i executed by r_j can be easily determined as follows:

$$f_{ij} = s_{ij} + \mathrm{nd}_i \tag{4}$$

where nd_i is the needed duration of task t_i.

The finish time is, in turn, used to determine whether the task's deadline can be satisfied. Therefore, we have the following deadline constraint C_3 on a resource allocation:

$$C_3: \begin{cases} x_{ij}=0 & \text{if } \forall\, j \in [1,m], s_{ij}+e_{ij} > d_i \\ x_{ij} \le 1 & \text{if } \exists j \in [1,m], s_{ij}+e_{ij} \le d_i. \end{cases} \tag{5}$$

If task t_i is executed by r_j, the needed duration nd_i must be guaranteed, and r_j is unavailable to other tasks in this time slot. This comes to the following constraint C_4:

$$C_4: \begin{cases} f_{ij}-s_{ij} \ge \text{nd}_i & \text{if } \forall\, j \in [1,m], x_{ij}=1 \\ (s_{ij} \ge f_{kj}) \vee (f_{ij} \le s_{kj}) & \text{if } \forall\, i \ne k, j \in [1,m] \\ & x_{ij}=x_{kj}=1 \end{cases} \tag{6}$$

where $f_{ij}-s_{ij}$ is the execution time of task t_i by resource r_j, and it must be larger than or equal to the needed duration nd_i. If task t_i and task t_k are both allocated to resource r_j, they cannot be executed at the same time, namely, $s_{ij} \ge f_{kj}$ and $f_{ij} \le s_{kj}$.

Additionally, the task scheduling is conducted in the airship control center on ground; thus, the communication cost can be ignored.

C. Scheduling Objectives

The task guarantee ratio and the priority guarantee ratio are the main objectives in emergency, particularly when a mass of emergent tasks simultaneously arrives. Thus, the scheduling objectives in our scheduling algorithm are to maximize the task guarantee ratio and the priority guarantee ratio as given below:

1) task guarantee ratio objective

$$\max \left\{ \sum_{i=1}^{n} \sum_{j=1}^{m} x_{ij} \Big/ n \right\} \tag{7}$$

2) priority guarantee ratio objective

$$\max \left\{ \sum_{i=1}^{n} \sum_{j=1}^{m} x_{ij} \cdot p_i \Big/ \sum_{i=1}^{n} p_i \right\}. \tag{8}$$

IV. Agent-Based Scheduling Model

The agent-based scheduling, in this study, is focused on the combination of a kind of market-like mechanism, i.e., CNP, which can allow agents to coordinate and produce desirable system-wide behavior. On the basis of the CNP, a novel bidirectional announcement mechanism is designed by employing some heuristics.

A. Agent Design

In our agent-based scheduling model, three kinds of agents are designed, i.e., administrator agent, resource agent, and task agent. Each of them has individual function. Importantly, they cooperate with each other to accomplish the auction process of CNP. A triple tuple $\langle \text{AD}^A, S^A, T^A \rangle$ is adopted to describe the three kinds of agents.

1) AD^A is the administrator agent.
2) $S^A = \{s_j^A, j = 1, 2, \ldots, m\}$ is the resource agent set, where s_j^A represents the jth resource agent in S^A.
3) $T^A = \{t_i^A, i = 1, 2, \ldots, n\}$ is the task agent set, where t_i^A denotes the ith task agent in T^A.

One task agent and one resource agent are corresponding to one task and one resource, respectively. Task agents yield with the arriving of tasks and die out with the finish of tasks. However, the administrator agent and the resource agents are always existent. The resource agents constantly update their own information based on the airships' states and release their information to the administrator agent.

B. Bidirectional Announcement Mechanism Design

Before the introduction of bidirectional announcement mechanism, we give the definitions of forward announcement and backward announcement as follows.

Definition 1. Forward Announcement: The forward announcement is the announcement from the task's perspective, i.e., the task information is treated as the announcement information to announce toward airship resources.

Definition 2. Backward Announcement: The backward announcement is the announcement from the resource's perspective, i.e., the resource information is treated as the announcement information to tasks.

Both the forward and backward announcements perform the collaborative process based on some rules and jointly accomplish the scheduling of tasks. Fig. 1 illustrates the process of the bidirectional announcement mechanism investigated in our study.

1) Forward Announcement: The forward announcement is triggered when a new task arrives to guarantee the quick response in emergency. In addition, one task agent corresponds to one task, and it starts to announce independently. The detailed process of forward announcement is described as follows.

a) The task agent produces announcement information, including the task ID, sensor, and resolution requirements and sends them to the administrator agent as well.
b) The administrator agent first receives plentiful task information announcements. Then, it matches each task agent with resource agents to select those resources that can satisfy the basic ability constraints (i.e., C_2). After that, the administrator agent sends the resource information to the corresponding task agent.
c) According to the feedback of the administrator agent, each task agent sends its detailed announcement information, including task ID, duration, deadline, and priority, to those related resource agents.
d) The resource agents receive the task's announcement information and calculate the corresponding bidding values based on some heuristic rules. If the value achieves the bidding standard, then the resource agent bids to the task agent, or it quits to bid.
e) Each task agent receives information from those resource agents that bid to the task. Then, it awards a contract to one resource agent based on some rules and then bids to it reversely.

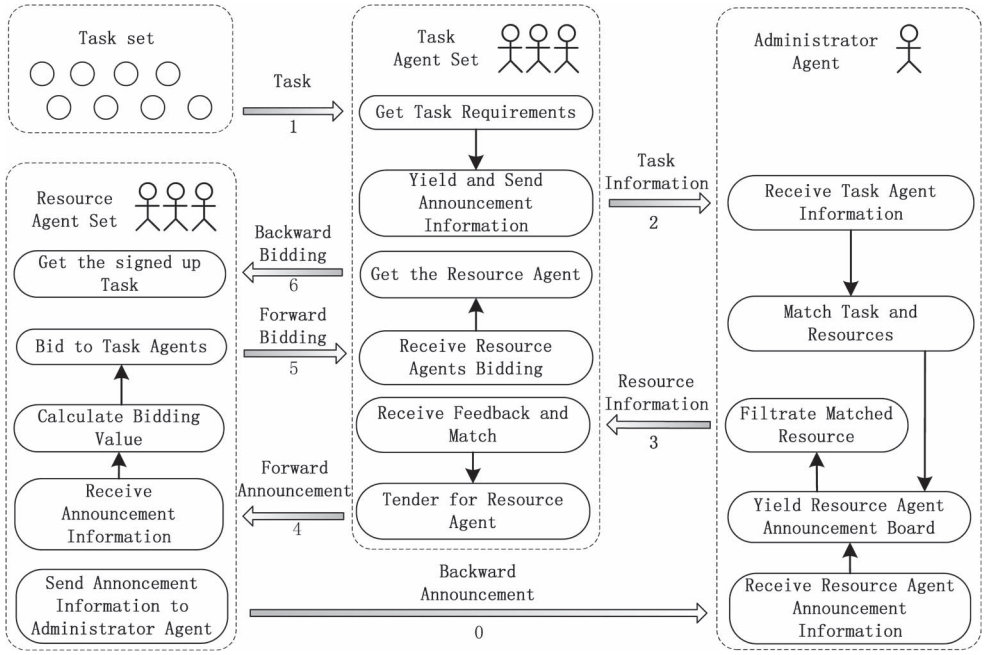

Fig. 1. Bidirectional announcement mechanism.

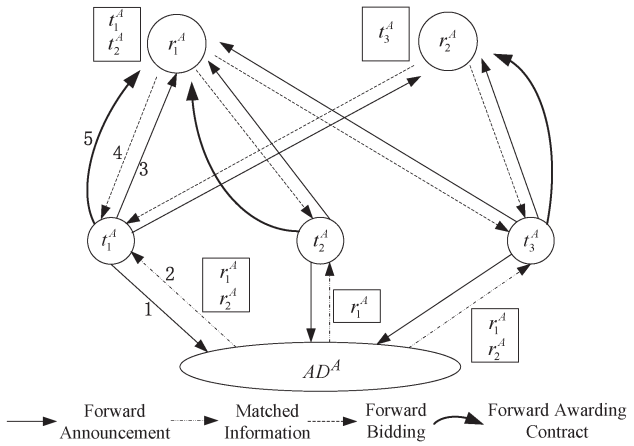

Fig. 2. Example of forward announcement.

Fig. 2 depicts an example of the forward announcement in which three task agents and two resource agents are considered.

Taking task agent t_1^A as an example, the sequence of forward announcement is given. From Fig. 2, we can observe that task agents t_1^A and t_3^A announce toward resource agents r_1^A and r_2^A, whereas task agent t_2^A can only announce to resource r_1^A. This is because resource agent r_2^A cannot satisfy the basic ability requirement of task agent t_1^A according to the feedback of administrator agent AD^A. Thus, the resource agent r_1^A bids to the three task agents t_1^A, t_2^A, and t_3^A; but the resource agent r_2^A only bids to the task agents t_1^A and t_3^A. Finally, based on some rules that will be introduced in Section IV-D, resource agent r_1^A is forwardly awarded a contractor by task agents t_1^A and t_2^A; resource agent r_2^A is forwardly awarded a contractor by task agent t_3^A.

2) Backward Announcement: In the backward announcement, resource agents find leaseholders, and the whole process includes two phases. In the first phase, resource agents announce their real-time information to the administrator agent. In the second phase, each resource agent selects task agents that bid to it reversely. It should be noted that there is only one queue for task arrival. In this paper, the allocation of tasks to resources is viewed as a matching process. The novelty of our work is to allow both the tasks and the resources to make announcements on the information to facilitate the matching. In the forward announcement, task agents supply task specifications and requirements. In the backward announcement, resource agents supply information about resources (e.g., the state of the resources). The whole process is described as follows.

a) The resource agents update their announcement information according to the airship resources' real-time states and announce the information to the administrator agent.
b) The administrator agent receives the resource announcement information and updates the resource announcement information billboard.
c) The task agents bid to resource agents reversely, i.e., the task agents send the contract confirmation requests to the candidate contractors—candidate resource agents. If a resource agent has only one task agent that bids to it, then it confirms the contract. Otherwise, the resource agent selects one task agent based on some rules in Section IV-D to finish the contract.

Similarly, Fig. 3 gives an example of the backward announcement on the basis of the example in Fig. 2.

The first phase in the backward announcement starts before the forward announcement. Resource agents r_1^A and r_2^A send their backward announcement information to the administrator agent. In the second phase, the resource agent r_1^A compares the backward bidding task agents and then selects the task agent t_1^A to award a contract according to some rules. For resource agent r_2^A, the only bidding task agent t_3^A is chosen to award a contract.

Consequently, based on the bidirectional announcement mechanism, task t_1 is allocated to resource r_1, and task t_3 is

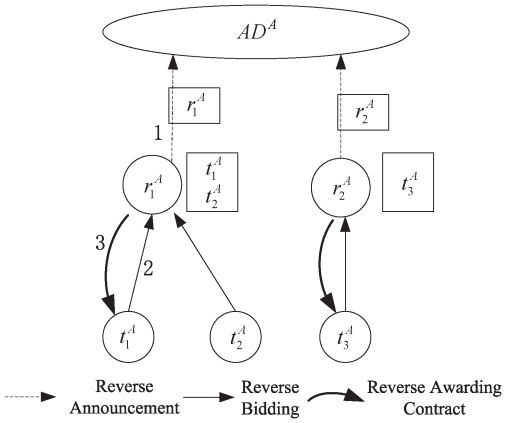

Fig. 3. Example of backward announcement.

allocated to resource r_2. For task t_2, it will be scheduled in the next-round announcement.

Noticeably, although the aforementioned bidirectional announcement mechanism adds the step that the administrator agent deals with the resource announcement information, it can prominently reduce the burden of the system. The administrator agent filters all the resources satisfying the basic ability constraint, which shrinks the scope of forward announcement. Thus, the calculations of forward announcement values are only for the matched resource agents. Those nonmatched resource agents cannot receive the forward announcement information, and thus, the calculation time is sufficiently decreased.

C. Calculation of Bidding Values

In the bidirectional announcement mechanism, the calculation of bidding values is a key part. From the point of resources, the resources having more capability to finish tasks are inclined to be selected. From the perspective of tasks, the tasks with tighter deadlines should be assigned as soon as possible. Before introducing the calculation of bidding values, we give the following two definitions.

Definition 3. Resource Adjustable Degree α_{ij}: The adjustable degree is the start time's changeable scope of task t_i if t_i is allocated to resource r_j, i.e.,

$$\alpha_{ij} = d_i - f_{kj} - \mathrm{nd}_i - \mathrm{pt}_{ij} \tag{9}$$

where f_{kj} is the finish time of task t_i's preceding task t_k on r_j.

Definition 4. Task Tight Degree β_{ij}: The tight degree is used to indicate a task's tight degree before its deadline and is relevant to the task priority, needed duration, deadline, and start time as follows:

$$\beta_{ij} = p_i^{\theta} \cdot \mathrm{nd}_i/(d_i - s_{ij}). \tag{10}$$

The parameter θ denotes the weight of priority. Note that $d_i - s_{ij}$ in (10) is larger than zero; else, the task will not be allocated to any resources.

1) Calculation of Bidding Value in Forward Announcement: In the forward announcement, task agents are responsible for making announcement, and those resource agents satisfying the basic ability constraint are for bidding. The bidding values in forward announcement reflect the capabilities of resource agents. The calculations of bidding values in forward

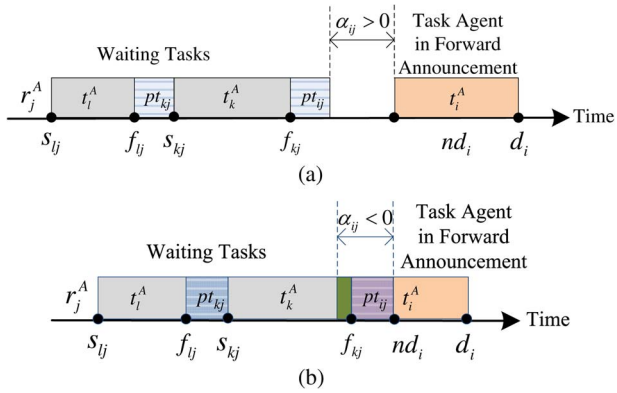

Fig. 4. Example of calculating α_{ij}.

announcement are on resource agents. We use BVFA_{ij} to denote the bidding value, i.e.,

$$\mathrm{BVFA}_{ij} = \alpha_{ij}. \tag{11}$$

If $\alpha_{ij} \geq 0$, it means that resource r_j has the ability to finish the task t_i without timing conflict. Meanwhile, the allocation of task t_i to resource r_j will not affect the executions of allocated tasks.

If $\alpha_{ij} < 0$, it indicates that resource r_j is heavily loaded. If task t_i is allocated to resource r_j, t_i cannot be finished before its deadline; else, it will affect the executions of other allocated tasks.

Fig. 4 shows an example of calculating α_{ij}.

2) Calculation of Bidding Value in Backward Announcement: In the backward announcement, the bidding value BVBA_{ij} of task t_i on resource r_j is calculated by the adjustable degree and the tight degree as follows:

$$\mathrm{BVBA}_{ij} = \beta_{ij} \Big/ \sum_{j=j_1}^{J} \alpha_{ij} \tag{12}$$

where $r_{j_1}, r_{j_2}, \ldots, r_J$ are resources whose adjustable degrees are larger than zero for task t_i.

The meaning in (12) is threefold. First, for a task, the tighter its finish time approaches its deadline, the higher the likelihood that this task cannot be successfully allocated; thus, it should be preferentially allocated. Second, the higher the priority, the higher the requirement to allocate the task. Third, the smaller the number of resources to execute a task, the smaller the feasibility to finish this task. Hence, the task should be allocated with higher preference.

D. Heuristic Strategies

In the bidirectional announcement mechanism, both the announcers need to select the contractors according to the bidding values. We denote $\mathrm{AR} = \{\mathrm{ar}_i, i = 1, 2, \ldots, n\}$ as an announcer set, $\mathrm{BR}_i = \{\mathrm{br}_{ij}, j = 1, 2, \ldots, m_i\}$ as a bidder set for announcer ar_i, and $\mathrm{BV} = \{\mathrm{bv}_{ij}, i = 1, 2, \ldots, n, j = 1, 2, \ldots, m_i\}$ as a bidding value set, where bv_{ij} represents the bidding value of bidder br_j for announcer ar_i. In our study, the selection of contractors abides by the following two strategies.

1) MAX Strategy: When more than one bidder simultaneously bid to one announcer, the announcer will select the bidder with the maximal bidding value.

$\forall \mathrm{ar}_i \in \mathrm{AR}$, if br_{ij} is selected as a contractor, it must have

$$\mathrm{bv}_{ij} = \max\{\mathrm{bv}_{iJ} | J = 1, 2, \ldots, m_i\}. \tag{13}$$

2) P Strategy: When more than one bidder bid to one announcer at the same time, the announcer will select a bidder with probability policy.

$\forall \mathrm{ar}_i \in \mathrm{AR}$, the winning probability pr_j of bidder br_{ij} can be calculated as

$$\mathrm{pr}_j = \mathrm{bv}_{ij} \Big/ \sum_{k=1}^{m_i} \mathrm{bv}_{ik}. \tag{14}$$

Without loss of generality, we let $\mathrm{pr}_0 = 0$. In addition, we let pr be a random number, and $\mathrm{pr} \in (0, 1)$. If the random number generated satisfies the formula in (15), then the bidder br_{ij} is selected as a contractor, i.e.,

$$\sum_{m=1}^{j-1} \mathrm{pr}_m < \mathrm{pr} \leq \sum_{n=1}^{j} \mathrm{pr}_n. \tag{15}$$

As a result, based on the preceding two strategies, we can produce four kinds of strategy combinations in the bidirectional announcement mechanism, i.e., *MAX-MAX*, *MAX-P*, *P-MAX*, and *P-P*.

V. ABDS Algorithm

Here, we present an efficient ABDS algorithm based on our agent-based scheduling model for independent aperiodic tasks with users' requirements in terms of timing, sensor, and resolution on multiple airships in emergency. Specifically, the ABDS integrates the aforementioned bidirectional announcement mechanism and the MAX and P strategies. In addition, ABDS efficiently considers the users' needs, schedulability, priority, and load balancing. To facilitate the presentation of the ABDS algorithm, it is necessary to introduce some scheduling principles about ABDS.

A. Fair Competition Principle of Roulette Wheel

The roulette wheel investigated in this paper is defined in Definition 5.

Definition 5. Roulette Wheel: In the forward announcement, the airship resources that satisfy the basic ability constraint are connected composing a daisy chain by their ID numbers. We call the daisy chain a roulette wheel. Every sector on the roulette wheel represents one resource agent.

Once a task agent begins the forward announcement, the roulette wheel rotates with step of one sector and with invariable direction. A fixed point is set on the center position of the sector where the corresponding resource agent is the first bidding resource. With the rotating of the roulette wheel, when the fixed point is in the center position of a sector, the corresponding resource agent starts to calculate its bidding value and bid. Since the buffer pool determines the count of bidding resources, the bidding probably ends before the roulette wheel finishes one round, i.e., there may be some resource agents that have not bidden yet while the buffer pool is full, and then the roulette wheel stops rotating. Consequently, the roulette wheel rotates until the buffer pool is full or it finishes one round. If the roulette wheel stops when the jth sector covers the fixed point, it begins rotating from the next sector in the next bidding.

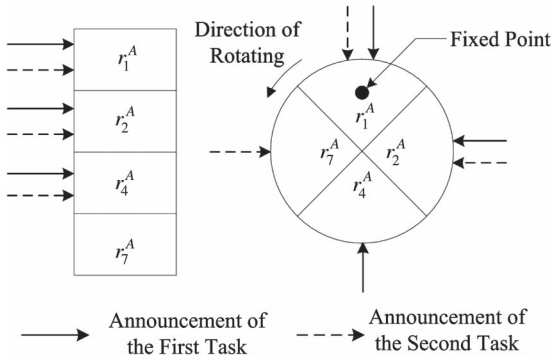

Fig. 5. Example of comparisons between resource listing and resource roulette wheel.

Now, we give the definition of buffer pool as follows.

Definition 6. Buffer Pool: The buffer pool in our study denoted by BP is an airship resource set in which each resource is able to bid in the forward announcement.

The fair competition principle is capable of efficiently avoiding some resource agents to get contracts in a short time interval. Due to the setting of the buffer pool, the traditional resource listing policy makes the resource agents start to bid from the first one for each task agent, which inevitably results in the local searching of resource agents. However, the roulette wheel can efficiently solve this problem. Fig. 5 depicts an example of comparisons between resource listing and resource roulette wheel.

We assume that the size of buffer pool s_{BP} is 3 and that there are four resources in the roulette wheel, i.e., r_1^A, r_2^A, r_4^A, and r_7^A. Two tasks announce to resources, respectively. The listing policy on the left in Fig. 5 indicates that every time the first resource agent begins to bid, it results in no chance for resource agent r_7^A to bid, whereas in the roulette wheel policy on the right in Fig. 5, the bidding from the resource agent r_1^A is for the first task, and bidding from r_7^A is for the second task. As a result, the loads of the resources are effectively balanced.

B. Dynamic Adjustment Principle of Buffer Pool

Our scheduling algorithm ABDS dynamically adjusts the BP size, i.e., the bidding resource count according to the change in the arrived tasks' count. We use s_{BP}, $s_{\mathrm{BP}}^{\mathrm{MAX}}$, and $s_{\mathrm{BP}}^{\mathrm{MIN}}$ to denote the size of BP, the maximal size of BP, and the minimal size of BP, respectively. Thus, we have the following simple expression:

$$s_{\mathrm{BP}} \in \left[s_{\mathrm{BP}}^{\mathrm{MIN}}, s_{\mathrm{BP}}^{\mathrm{MAX}}\right]. \tag{16}$$

Setting the value of s_{BP} is the key to decide the size of buffer pool. In this paper, we employ a simple but efficient approach. First, we define a function $f(x)$ as follows:

$$f(x) = \begin{cases} 0 & x \leq L_1 \\ 1 & L_1 < x \leq L_2 \\ \cdots & \cdots \\ K-1 & L_{K-1} < x \leq L_K \\ K & x > L_K \end{cases} \tag{17}$$

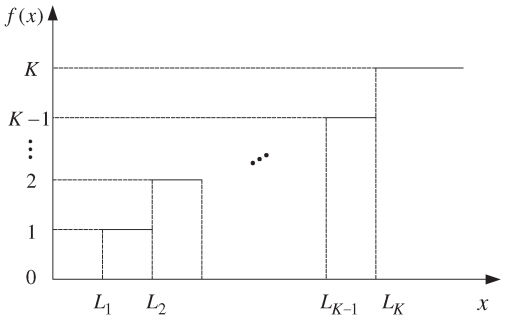

Fig. 6. Diagram of function $f(x)$.

where variable x is an integer, and function value $f(x)$ is a set of integers from 0 to K. $L_1, L_2, \ldots, L_K$ are a series of critical points used to describe the scope of x. Fig. 6 depicts the function $f(x)$.

We define a set $\{s_{\mathrm{BP}}^{\mathrm{MIN}} + \Delta \cdot f(x)\}$, in which, if the value of $\Delta \cdot f(x)$ changes, the size of buffer pool, i.e., s_{BP} varies. For each s_{BP}, it satisfies formula (16). In this paper, we set $L = \lfloor (n/s) \rfloor$, where n is the task count, and s is the count of resource agents in the roulette wheel; thus, the size of buffer pool can be determined by

$$s_{\mathrm{BP}} = s_{\mathrm{BP}}^{\mathrm{MIN}} + \left\lfloor \frac{s - s_{\mathrm{BP}}^{\mathrm{MIN}}}{K} \cdot f(L) \right\rfloor. \tag{18}$$

C. ABDS Description

Here, we present the novel ABDS algorithm for task scheduling on multiple airship resources. The ABDS is based on the cooperation of multiple agents. The detailed ABDS is depicted by the following steps.

Step 1. When dynamic tasks arrive, the newly arrived tasks and those tasks in waiting queue compose an allocating task set T, and set AT $= \emptyset$, TT $= \emptyset$.

Step 2. Match tasks in T with resources in resource set R based on constraint C_2 to find the resources satisfying the basic ability constraint and then put them into set AT.

Step 3. Compute the start times of tasks in AT by (3) and put those tasks satisfying constraints C_3 and C_4 into set TT.

Step 4. If set TT is $\emptyset$, then go to Step 17; otherwise, go to Step 5.

Step 5. Map tasks in TT to task agents, map resources in R to resource agents, and produce an administrator agent AD^A.

Step 6. Send all the resource agents satisfying C_2 of task agent t_i^A to AD^A and then put them in the roulette wheel.

Step 7. Set the buffer pool size by (18).

Step 8. Send the announcement information of t_i^A to all the resource agents in the roulette wheel by AD^A.

Step 9. Set $p = 0$, $q = 0$.

Step 10. If the buffer pool is full, or the roulette wheel finishes one round, go to Step 15; otherwise, go to Step 11.

Step 11. Compute the bidding value in forward announcement BVFA$_{ij}$ by (11).

Step 12. If BVFA$_{ij} \geq 0$, go to Step 13; else, go to Step 14.

Step 13. Resource agent r_j^A bids, set $p = p + 1$, $q = q + 1$, then put r_j^A to buffer pool, and the roulette wheel rotates to the next sector, go to Step 10.

Step 14. Resource agent r_j^A does not bid, set $p = p + 1$, and the roulette wheel rotates to the next sector, go to Step 10.

Step 15. Employ MAX strategy or P strategy to select the bidding resource agents in the buffer pool, i.e., confirm the contractor agent r_j^A for t_i^A.

Step 16. Remove the task t_i from TT, go to Step 4.

Step 17. If resource agent set R^A is $\emptyset$, then go to Step 22; else, go to Step 18.

Step 18. Set $D = \emptyset$.

Step 19. Each resource agent in R^A starts the backward announcement. Calculate the bidding value of backward announcement by (12) and put the bidding tasks in D.

Step 20. Employ MAX strategy or P strategy to select the task in D, i.e., confirm the contractor task agent t_k^A for resource agent s_j^A.

Step 21. Remove the resource agent s_j^A from S_A and remove t_k^A from AT.

Step 22. If AT is $\emptyset$, then go to Step 23; otherwise, go to Step 3.

Step 23. End.

In ABDS, the bidding values are calculated considering the states of all candidate bidders, which makes the sequence of tasks or airships show little impact on the scheduling results. As a result, the adaptivity and stability of ABDS can be guaranteed.

VI. Performance Evaluation

We evaluate here the performance of the proposed ABDS algorithm. By employing different strategies, namely, MAX-MAX, MAX-P, P-MAX, and P-P, in bidirectional announcement mechanism, four corresponding subalgorithms of ABDS can be produced: ABDS-1, ABDS-2, ABDS-3, and ABDS-4, respectively. To demonstrate the performance improvements gained by ABDS, we quantitatively compare it with two baseline algorithms—unidirectional announcement (UA) scheduling algorithm and genetic algorithm (GA). The UA algorithm is briefly described as the following steps.

Step 1. When dynamic tasks arrive, the newly arrived tasks and those tasks in waiting queue compose an allocating task set T.

Step 2. Select a task t_i with the highest priority from T and calculate the adjustable degree α_{ij} $(j = 1, 2, \ldots, m)$ on all resources.

Step 3. If $\exists\ \alpha_{ij} \geq 0$, then go to Step 4; else, go to Step 5;

Step 4. Select the resource with maximal α_{ij}, and then allocate task t_i to resource r_j, remove t_i from T, go to Step 6.

Step 5. Reject task t_i and remove t_i from T.

Step 6. If T is not $\emptyset$, then go to Step 2; else, go to Step 7.

Step 7. End.

The task scheduling employing GA [37] is briefly described as follows.

Step 1. When new tasks arrive in a batch, initialize the chromosome according to task count and generate some individual solutions randomly.

Step 2. Calculate the fitness of each individual: fitness $=$ task guarantee ratio $+$ priority guarantee ratio (task guarantee ratio and priority guarantee ratio will be defined in performance metric part).

Step 3. Select individuals (individuals with higher fitness are more likely to be selected) to breed the next generation through crossover and mutation.

Step 4. Calculate individuals' fitness according to the formula in step 2 and check whether it has reached the termination condition. If so, go to Step 5; otherwise, go to step 3.

Step 5. Find the individual with the maximal value of fitness and allocate tasks to airships according to its genes.

Step 6. End.

Note that using baseline algorithms to demonstrate strengths of proposed algorithms is widely used in many similar studies.

The performance metrics by which we evaluate the system performance include the following.

1) Task guarantee ratio (TGR) is defined as follows: TGR = total count of tasks guaranteed to meet their deadlines/ total number of tasks $\times$ 100%.
2) Priority guarantee ratio (PGR) is calculated as follows: PGR = priority sum of tasks guaranteed to meet their deadlines/priority sum of tasks $\times$ 100%.

A. Simulation Method and Parameters

In our simulations, we assume that there are four airships that cooperate to conduct the Earth-observing on a 10 000 km^2 area in emergency. The sensors in these airships are homogeneous CCDs. The best ground resolution is 1 m, and the radius of cover area is 100 km.

1) The simulation interval is 0–300. Random time instants are selected, i.e., 0, 12, 119, 187, and 209. At these time instants, tasks dynamically arrive in a batch mode.
2) The task length is set as

$$\text{taskLength} = 1000 + \text{Math.round}\left(\text{uniform}(0, 10\,000)\right) \quad (19)$$

where uniform() represents a function of uniform distribution, and Math.round() is used to round the generated values. In our study, the task length is uniformly distributed between 10 000 and 20 000.

3) The task deadline is generated by

$$\text{deadline} = \text{arrivalTime} + \text{Math.round}(\text{uniform}(30, 120)). \quad (20)$$

The task deadline adds a random value between 30 and 120 based on its arrival time.

4) The priority is set as

$$\text{priority} = \text{Math.round}\left(\text{uniform}(1, 10)\right). \quad (21)$$

5) The needed sensor type is CCD, and the required resolution is 1 m.
6) The five-group data are tested with different task counts. The upper limits of task count for each group are 60, 70, 80, 90, and 100, respectively. The needed durations of tasks are randomly chosen in (0–10].
7) The value of θ is set to 1 when calculating the bidding value in the backward announcement.
8) The minimal size of BP $s_{\text{BP}}^{\text{MIN}}$ is set to 1; and $K = 3$, $L_1 = 25$, $L_2 = 50$, and $L_3 = 100$.

TABLE I
TASK INFORMATION

Group No.	Task count in a batch					Total task count	Total priorities
1	17	14	46	27	52	156	835
2	31	26	11	63	44	175	956
3	43	29	62	8	70	212	1153
4	66	32	84	45	58	285	1550
5	78	53	94	31	82	338	1764

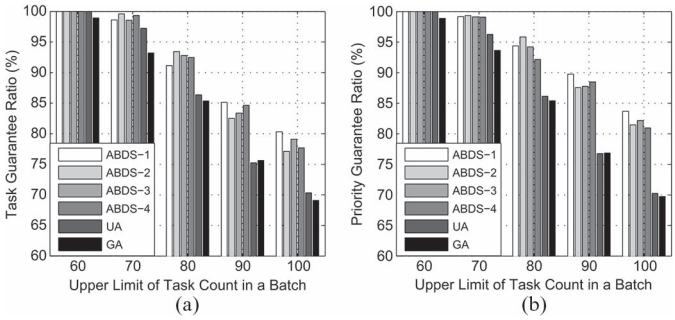

Fig. 7. Performance impact on random selection of tasks.

The generated task information for simulations is listed in Table I.

B. Comparisons Between ABDS and Baseline Algorithms

Two series of experiments were carried out to compare ABDS and baseline algorithms in two settings: 1) random selection of tasks and 2) priority-driven selection of tasks.

1) Random Selection of Tasks: In this experiment, both ABDS and baseline algorithms randomly choose tasks for allocation. Fig. 7 shows the performances of ABDS-1, ABDS-2, ABDS-3, ABDS-4, UA, and GA in terms of task guarantee ratio and priority guarantee ratio.

We observe from Fig. 7(a) that ABDS-1, ABDS-2, ABDS-3, and ABDS-4 always have higher task guarantee ratio than UA and GA. ABDS employs the bidirectional announcement mechanism and considers both the adjustable degree and the tight degree, thereby allowing more tasks to be accommodated. In contrast, UA does not employ the bidirectional announcement mechanism and only takes the adjustable degree into consideration. In the GA, the evolution of individual solutions mainly depends on crossover, while the crossover cannot ensure a better fit individual solution. The mix of genes from better fit "parent" individuals may result in resource conflict on the "child" individuals. This is because crossover is progressed from the perspective of genes rather than from the global view, leading to a lot of random evolution directions. Unlike the aforementioned baseline algorithms, ABDS adopts the buffer pool and roulette wheel mechanisms to improve global optimization and load balancing of airship resources. This allows more tasks to meet their deadlines. Another observation from Fig. 7(a) is that when the upper limit of task count in a batch reached 90, the task guarantee ratios of ABDS-1, ABDS-2, ABDS-3, and ABDS-4 significantly decrease. This is due to the fact that the resource count is invariable. With an increase in task count, the airships became heavily loaded. Consequently, more tasks cannot be accepted.

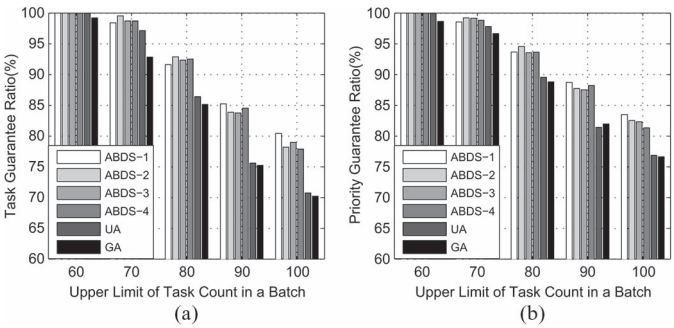

Fig. 8. Performance impact on priority-driven selection of tasks.

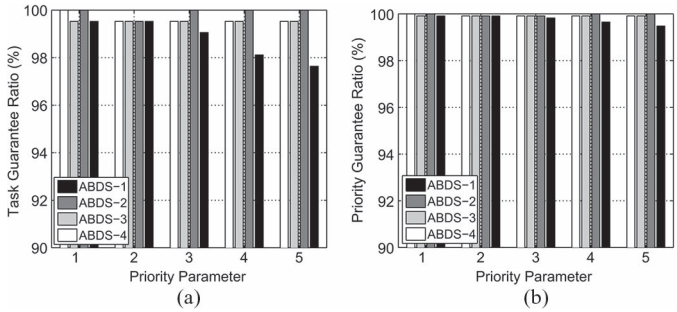

Fig. 9. Performance impact on light workload.

Fig. 7(b) shows that ABDS-1, ABDS-2, ABDS-3, and ABDS-4 achieved much better priority guarantee ratio than UA and GA.

It is shown in Fig. 7(a) and (b) that, when the increased task count results in heavy workload of resources (e.g., the upper limit of task count is 100), both the task guarantee ratio and the priority guarantee ratio of UA and GA are basically identical. However, ABDS-1, ABDS-2, ABDS-3, and ABDS-4 have achieved significantly higher priority guarantee ratios than their task guarantee ratios. This is because in the backward announcement, the selection policy of tasks by resources sufficiently considers the task priorities. The higher the priority of the task, the higher the probability that the task will be selected. Thus, ABDS preferentially allocates those tasks with higher priorities when the resources are heavily loaded, rejecting tasks with lower priorities.

2) Priority-Driven Selection of Tasks: In this group of experiments, when new tasks arrive, the ABDS and baseline algorithms sort the tasks by their priorities in descent order and then allocate these tasks. Fig. 8 shows the performances of ABDS-1, ABDS-2, ABDS-3, ABDS-4, UA, and GA.

It is shown in Fig. 8 that ABDS achieved significantly better performances than the baseline algorithms both in terms of task guarantee ratio and priority guarantee ratio.

By comparing the results in Figs. 7(a) and 8(a), we observe that the ABDS and baseline algorithms have similar performance in terms of task guarantee ratio. This suggests that the task scheduling sequence in the task set has little impact on the task guarantee ratio.

Unlike the experimental results for random task selection in the preceding subsection where UA and GA achieved similar task guarantee ratio and priority guarantee ratio, it can be observed in Figs. 7(b) and 8(b) that, for the experiments on priority-driven task selection, as the task count increased, the baseline algorithms achieved higher priority guarantee ratios than task guarantee ratios. For the UA, this is because the high-priority tasks are preferentially allocated. Regarding the GA, the reason is that those tasks with high priority would locate in the forepart of the chromosome under priority-driven task selection. When it finally chooses the individual with the best fit value to get a scheduling decision, the tasks in the forepart will be preferentially allocated, that is to say, tasks with higher priority would be scheduled first. As a consequence, GA can reach higher priority guarantee ratio under priority-driven task selection. However, both UA and GA are inferior to ABDS in terms of task guarantee ratio and priority guarantee ratio. This is because ABDS improves global optimization by employing the buffer pool and roulette wheel mechanisms. The priority guarantee ratios of ABDS-1, ABDS-2, ABDS-3, ABDS-4 in Fig. 8(b) are slightly higher than those in Fig. 7(b). The experimental results in Figs. 7(b) and 8(b) suggest that the task scheduling sequence has significant impact on UA and GA but little influence on ABDS.

Consequently, compared with baseline algorithms, ABDS has clearly achieved significantly better performances both in task guarantee ratio and priority guarantee ratio. Additionally, the scheduling results obtained by ABDS are not severely affected by the task scheduling sequence, but this is not the case for UA and GA. It can be concluded that ABDS has better adaptivity and stability and that its performance is not dependent on the distribution of task priorities.

C. Parameter Optimization of ABDS

Compared with the baseline algorithms, ABDS achieved higher priority guarantee ratio than task guarantee ratio because it is inherently designed to consider task priorities in its selection policy. The inclination to consider task priorities is derived from the bidding value in the backward announcement [see (10) and (12)]. Meanwhile, the value of θ plays an important role to determine the weight of priority. The goal of this experiment is to investigate the impacts of θ on the performance of ABDS.

From the experimental results comparing ABDS and baseline algorithms, the following analysis can be drawn. When the upper limit of task count in a batch is less than 70 and $\theta = 1$, ABDS basically has 100% task guarantee ratio. This indicates that the resources are under light workload. However, when the upper limit of the task count in a batch achieved 100, the task guarantee ratio of ABDS is only about 80%. This suggests that the current resources are heavily loaded. Therefore, in Table I, we choose the third and fifth groups of data, which represent the light workload and the heavy workload, respectively. In addition, the tasks in the task set are randomly selected for allocation.

1) Light Workload: In this set of experiment, the third group data in Table I are selected to test the impact of θ on ABDS when the resources are under light workloads. Fig. 9 shows the performance of ABDS when varying the parameter from 1 to 5 with an increment of 1.

It is observed from Fig. 9(a) that, with an increase in θ, the task guarantee ratio and the priority guarantee ratio of ABDS-1

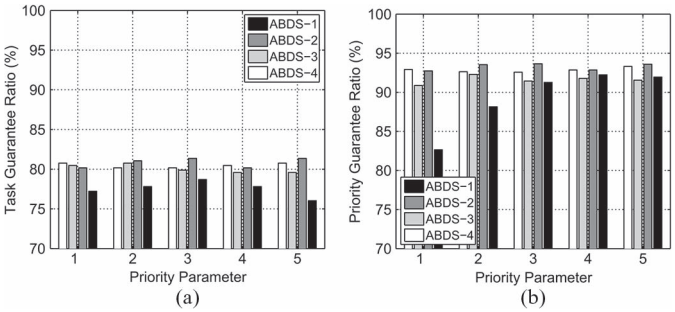

Fig. 10. Performance impact on heavy workload.

gradually decreased, but only a small variation is recorded in the other subalgorithms of ABDS. The task guarantee ratio and the priority guarantee ratio basically approach 100%. The experimental results suggest that, when the resources are lightly loaded, high-weight priority will result in lower task guarantee ratio and further negative impact of the priority guarantee ratio. Hence, it is appropriate to set $\theta = 1$.

2) Heavy Workload: In this set of experiments, the fifth group data in Table I are chosen to evaluate the impact of θ on ABDS when the resources are under heavy workload. We varied θ from 1 to 5. Fig. 10 shows the performances of ABDS.

When the upper limit of the task count in a batch achieved 100, the task guarantee ratio and the priority guarantee ratio of ABDS slightly changed. ABDS-1 achieved the highest task guarantee ratio when θ is set to 3, and its priority guarantee ratio increased with θ. However, ABDS-2, ABDS-3, and ABDS-4 generally have stable performances regardless of the change in θ. The results of the task guarantee ratio and the priority guarantee ratio of ABDS suggest that it is appropriate to set $\theta = 3$.

Consequently, when the resources are under light workload, θ should be set to a small value (e.g., setting $\theta = 1$). Otherwise, it would be more appropriate for θ to be set to a larger value, e.g., setting $\theta = 3$.

VII. Conclusion and Future Work

In this paper, we have presented a novel ABDS strategy for aperiodic independent real-time tasks on multiple airships in emergency. The ABDS employs a novel CNP-based bidirectional announcement mechanism, in which our contributions include designing the forward and backward announcements, as well as their process flows. Additionally, we defined the adjustable degree and the tight degree for calculating the bidding values. Furthermore, two heuristic strategies, i.e., MAX strategy and P strategy, are proposed to determine the contractors while bidding. To further enhance the scheduling quality, two principles, i.e., fair competition principle of roulette wheel and dynamic adjustment principle of buffer pool, are investigated, and they are seamlessly integrated into ABDS. Experimental results in Section VI show that ABDS efficiently improves the global optimization and load balancing of airship resources. The extensive simulation studies show that ABDS is a feasible scheduling strategy designed for multiple airships in emergency. The ABDS strategy can be generalized and applied to other domains involving distributed systems.

The ABDS strategy and our simulation studies are the first step toward the development of agent-based scheduling mechanisms for multiple airships. In our future studies, we plan to address the following four issues: First, we will implement a new scheduling mechanism in which communication and dispatching times are taken into account. Second, we will integrate the Lagrangian relaxation with our ABDS strategy to further optimize our scheduling objectives. Third, we will consider multiple different sensors in the same airship when scheduling. Fourth, the robustness will be further considered in the ABDS strategy.

References

[1] R. Y. Purandare, "A buoyancy-propelled airship," Ph.D. dissertation, New Mexico State Univ., Las Cruces, NM, USA, 2007.
[2] X. Li, X. Fang, Q. Dai, and Z. Zhou, "Modeling and analysis of floating performances of stratospheric semi-rigid airships," *Adv. Space Res.*, vol. 50, no. 7, pp. 881–890, Oct. 2012.
[3] N. Bianchessi, J. F. Cordeau, J. Desrosiers, G. Laporte, and V. Raymond, "A heuristic for the multi-satellite, multi-orbit and multi-user management of earth observation satellites," *Eur. J. Oper. Res.*, vol. 177, no. 2, pp. 750–762, Mar. 2007.
[4] K. Hwang and Z. Xu, *Scalable Parallel Computing: Technology, Architecture, Programming*. New York, NY, USA: McGraw-Hill, 1998.
[5] *McGraw-Hill Dictionary of Scientific & Technical Terms*, 6th ed., McGraw-Hill, New York, NY, USA, 2003.
[6] W. C. Lin and S. C. Chang, "Hybrid algorithms for satellite imaging scheduling," in *Proc. IEEE Int. Conf. Syst., Man Cybern.*, 2005, vol. 3, pp. 2518–2523.
[7] N. Bianchessi and G. Righini, "Planning and scheduling algorithms for the COSMO-SkyMed constellation," *Aerosp. Sci. Technol.*, vol. 12, no. 7, pp. 535–544, Oct. 2008.
[8] P. Wang and G. Reinelt, "A heuristic for an earth observing satellite constellation scheduling problem with download considerations," *Electron. Notes Discr. Math.*, vol. 36, no. 1, pp. 711–718, Aug. 2010.
[9] D. Y. Liao and Y. T. Yang, "Imaging order scheduling of an earth observation satellite," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 37, no. 5, pp. 794–802, Sep. 2007.
[10] D. Y. Liao and Y. T. Yang, "Satellite imaging order scheduling with stochastic weather condition forecast," in *Proc. IEEE Int. Conf. Syst., Man Cybern.*, 2005, vol. 3, pp. 2524–2529.
[11] J. Wang, X. Zhu, J. Zhu, and M. Ma, "Emergency scheduling of multiple imaging satellites with dynamic merging," in *Proc. 12th Int. Conf. Space Oper.*, 2012, pp. 1–12.
[12] B. Bethke, "UAV task assignment," *IEEE Robot. Autom. Mag.*, vol. 15, no. 1, pp. 39–44, Mar. 2008.
[13] L. F. Bertuccelli, "Robust planning for coupled cooperative UAV missions," in *Proc. 43rd IEEE Conf. Decision Control*, Paradise Island, Bahamas, 2004, pp. 2917–2922.
[14] A. Richards, "Coordination and control of multiple UAVs," presented at the AIAA Guidance, Navigation, Control Conference Exhibit, Monterey, CA, USA, 2002, Paper AIAA2002-4588.
[15] Y. Yin and S. Huang, "Optimization deployment of multi-sensor platforms in near-space based on adaptive genetic algorithm," in *Proc. Int. Conf. Inf. Eng. Comput. Sci.*, 2009, pp. 1–5.
[16] X. Wang, X. Gao, R. Zong, and P. Cheng, "An optimal model and solution of deployment of airships for high altitude platforms," in *Proc. Int. Conf. Wireless Commun. Signal Process.*, 2010, pp. 1–6.
[17] W. Xiang and H. P. Lee, "Ant colony intelligence in multi-agent dynamic manufacturing scheduling," *Eng. Appl. Artif. Intell.*, vol. 21, no. 1, pp. 73–85, Feb. 2008.
[18] H. Goldingay and J. Mourik, "The effect of load on agent-based algorithms for distributed task allocation," *Inf. Sci.*, vol. 222, pp. 66–80, Feb. 2013.
[19] N. Liu, M. A. Abdelrahman, and S. Ramaswamy, "A complete multiagent frame for robust and adaptable dynamic job shop scheduling," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 37, no. 5, pp. 904–916, Sep. 2007.
[20] M. Campos, E. Bonabeau, G. Theraulaz, and J. L. Deneubourg, "Dynamic scheduling and division of labor in social insects," *Adapt. Behav.*, vol. 8, no. 2, pp. 83–94, Mar. 2000.

[21] P. Janacik, T. Heimfarth, and F. Ramming, "Emergent topology control based on division of labour in ants," in *Proc. 20th Int. Adv. Inf. Netw. Appl.*, 2006, pp. 733–740.
[22] R. Price, "Evaluation of adaptive nature inspired task allocation against alternate decentralised multiagent strategies," Ph.D. dissertation, Univ. Birmingham, Birmingham, U.K., 2004.
[23] G. Beccari, S. Caselli, and F. Zanichelli, "A technique for adaptive scheduling of soft real-time tasks," *J. Real-Time Syst.*, vol. 30, no. 3, pp. 187–215, Jul. 2005.
[24] M. B. Dias, R. Zlot, N. Kalra, and A. Stentz, "Market-based multirobot coordination: A survey and analysis," *Proc. IEEE*, vol. 94, no. 7, pp. 1257–1270, Jul. 2006.
[25] R. G. Smith, "The contract net protocol: High-level communication and control in distributed problem solver," *IEEE Trans. Comput.*, vol. C-29, no. 12, pp. 1104–1113, Dec. 1980.
[26] H. V. D. Parunak, "Manufacturing experience with the contract net," in *Distributed Artificial Intelligence*. San Mateo, CA, USA: Morgan Kaufmann, 1987, pp. 285–310.
[27] R. Macchiaroli and S. Riemma, "A negotiation scheme for autonomous agents in job shop scheduling," *Int. J. Comput. Integr. Manuf.*, vol. 15, no. 3, pp. 222–232, Jan. 2002.
[28] P. Sousa and C. Ramos, "A distributed architecture and negotiation protocol for scheduling in manufacturing systems," *Comput. Ind.*, vol. 38, no. 2, pp. 103–113, Mar. 1999.
[29] J. S. K. Lau, G. Q. Huang, K. L. Mak, and L. Liang, "Agent-based modeling of supply chains for distributed scheduling," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 36, no. 5, pp. 847–861, Sep. 2006.
[30] M. Owliya, M. Saadat, G. G. Jules, M. Goharian, and R. Anane, "Agent-based interaction protocols and topologies for manufacturing task allocation," *IEEE Trans. Syst., Man, Cybern.*, vol. 43, no. 1, pp. 38–52, Jan. 2013.
[31] B. P.-C. Yen and O. Q. Wu, "Internet scheduling environment with market-driven agents," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 34, no. 2, pp. 281–289, Mar. 2004.
[32] J. Vancza and A. Markus, "An agent model for incentive-based production scheduling," *Comput. Ind.*, vol. 43, no. 2, pp. 173–187, Oct. 2000.
[33] A. D. Baker, "Metaphor or reality: A case study where agents bid with actual costs to schedule a factory," in *Market-Based Control: A Paradigm for Distributed Resource Allocation*. Singapore: World Scientific, 1996, ch. 8, pp. 184–223.
[34] K. Fischer, B. Chaib-draa, J. P. Muller, M. Pischel, and C. Gerber, "A simulation approach based on negotiation and cooperation between agents—A case study," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 29, no. 4, pp. 531–545, Nov. 1999.
[35] E. Kutanoglu and S. D. Wu, "On combinatorial auction and Lagrangian relaxation for distributed resource scheduling," *IIE Trans.*, vol. 31, no. 9, pp. 813–826, Sep. 1999.
[36] Y. Wang, F. Zhou, J. Zhou, and J. Guo, "Stratospheric airship spot hovering control based on improved genetic algorithm," *Fire Control Command Control*, vol. 35, no. 2, pp. 22–27, 2010.
[37] D. Whitley, "A genetic algorithm tutorial," *Stat. Comput.*, vol. 4, no. 2, pp. 65–85, Jun. 1994.

Xiaomin Zhu (M'10) received the B.S. and M.S. degrees from Liaoning Technical University, Liaoning, China, in 2001 and 2004, respectively, and the Ph.D. degree from Fudan University, Shanghai, China, in 2009, all in computer science.

He is currently an Associate Professor with the College of Information Systems and Management, National University of Defense Technology, Changsha, China. He has authored or coauthored over 50 research articles in refereed journals and conference proceedings such as the IEEE TRANSACTIONS ON COMPUTERS (TC), the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED Systems, and Journal of Parallel and Distributed Computing (JPDC). His research interests include scheduling and resource management in green computing, cluster computing, cloud computing, multiple satellites, and airships.

Kwang Mong Sim (SM'07) received the B.Sc. degree (*summa cum laude*) from the University of Ottawa, Ottawa, ON, Canada, and the Ph.D. and M.Sc. degrees from the University of Calgary, Calgary, AB, Canada.

He is currently a Chair (Professor) in computer science and the Founder and Director of the Computational Economics and Multi-Agent Systems Laboratory with the University of Kent, Kent, U.K. He has delivered many keynote lectures on intelligent cloud computing, agent-based cloud computing, and automated negotiation in many international conferences. He has authored or coauthored over 150 technical papers, including papers that received the honor of being selected as ACM Computing Review Best of 2013 paper and IEEE Transactions spotlight paper, frequently cited IEEE Transactions survey papers, and best papers in international conferences.

Jianqing Jiang received the B.S. and M.S. degrees in information systems from National University of Defense Technology, Changsha, China, in 2010 and 2013, respectively.

He is currently an Assistant Engineer with the College of Information Systems and Management, National University of Defense Technology. His research interests include airship scheduling and unmanned aerial vehicle application.

Jianjiang Wang received the B.S. degree in information systems from National University of Defense Technology, Changsha, China, in 2009. He is currently working toward the Ph.D. degree in the College of Information Systems and Management, National University of Defense Technology.

He is also currently with the Faculty of Economics and Business, Katholieke Universiteit Leuven, Leuven, Belgium, for joint cultivation. His research interests include satellite scheduling, and uncertain programming.

Chao Chen received the B.S. degree in information systems from National University of Defense Technology, Changsha, China, in 2013. He is currently working toward the M.S. degree in the College of Information Systems and Management, National University of Defense Technology.

His research interests include agent-based scheduling, cloud computing, and mobile cloud computing.

Zhong Liu received the B.S. degree in physics from Central China Normal University, Wuhan, China, in 1990 and the M.Sc. degree in computer science and the Ph.D. degree in management science from National University of Defense Technology, Changsha, China, in 1997 and 2000, respectively.

He is currently a Professor with the College of Information Systems and Management, National University of Defense Technology. His research interests include information management and decision-making support technology.