



Article

Task Planning for Multiple-Satellite Space-Situational-Awareness Systems

Yutao Chen ¹, Guoqing Tian ¹, Junyou Guo ¹ and Jie Huang ^{1,2,*}

¹ College of Electrical Engineering and Automation, Fuzhou University, Fuzhou 350108, China; ytchen_fzu@163.com (Y.C.); N190127122@fzu.edu.cn (G.T.); N190127103@fzu.edu.cn (J.G.)

² Beijing Institute of Control Engineering, Beijing 100081, China

* Correspondence: jie.huang@fzu.edu.cn

Abstract: Space situational awareness (SSA) plays an important role in maintaining space advantages. Task planning is one of the key technologies in SSA to allocate multiple tasks to multiple satellites, so that a satellite may be allocated to supervise multiple space objects, and a space object may be supervised by multiple satellites. This paper proposes a hierarchical and distributed task-planning framework for SSA systems with focus on fast and effective task planning customized for SSA. In the framework, a global task-planner layer performs satellite and object clustering, so that satellites are clustered into multiple unique clusters on the basis of their positions, while objects are clustered into multiple possibly intersecting clusters, hence allowing for a single object to be supervised by multiple satellites. In each satellite cluster, a local task planner performs distributed task planning using the contract-net protocol (CNP) on the basis of the position and velocity of satellites and objects. In addition, a customized discrete particle swarm optimization (DPSO) algorithm was developed to search for the optimal task-planning result in the CNP. Simulation results showed that the proposed framework can effectively achieve task planning among multiple satellites and space objects. The efficiency and scalability of the proposed framework are demonstrated through static and dynamic orbital simulations.

Keywords: space situational awareness; task planning; contract-net protocol; multiagent system



Citation: Chen, Y.; Tian, G.; Guo, J.; Huang, J. Task Planning for Multiple-Satellite Space-Situational-Awareness Systems. *Aerospace* **2021**, *8*, 73. <https://doi.org/10.3390/aerospace8030073>

Academic Editors: Alexei Sharpanskykh and Pierre Rochus

Received: 5 February 2021

Accepted: 10 March 2021

Published: 12 March 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Space situational awareness (SSA) describes the knowledge and understanding of the situation in near-Earth space, including energy, particles, and natural and artificial objects [1,2]. SSA is required to detect, track, or supervise space objects and environments [3]. SSA can be achieved either from the ground or space, and the latter is usually realized by space platforms, e.g., multiple small satellites [4]. Compared with a single large satellite, multiple-satellite systems (MSS) have increasing popularity due to their better robustness and flexibility. In particular, heterogeneous satellite systems can undertake different tasks within a large area of space and over a long period of time.

As the main purpose of SSA is to detect and supervise multiple spacecraft, stations, and millions of space debris objects, one of the key technologies for MSS is task planning, i.e., how to allocate multiple tasks to multiple satellites to achieve the best overall execution performance [5]. As MSS with onboard planning capabilities can be classified into intelligent multiagent systems (MAS), results from task planning for MAS, including unmanned aerial vehicles, multirobot systems, and flexible manufacturing, can be employed.

Typically, MSS can be classified into two types by way of connection among satellites, namely, centralized and distributed. Centralized MSS are controlled by a central station or a single satellite that makes decisions and plans for the entire system. Such an architecture is easy to realize, but suffers from poor real-time computation capabilities and adaptivity to urgent situations [6,7]. The centralized architecture formulates the task-planning problem into an optimization problem where the cost function is overall system performance.

Swarm optimization algorithms [8,9], mixed-integer optimization [10], and heuristic algorithms [11] can be applied to solve such an optimization problem. In contrast, distributed MSS achieves certain configuration and task planning through intersatellite communication and coordination. Such a structure can allocate computations, communications, and control actions through distributed coordination algorithms [4]. The distributed architecture has better scalability, and can achieve parallel planning and fast replanning. In addition, the distributed architecture is more robust to agent failure, hence being more flexible in dynamic environments. Several distributed task-planning methods for MAS were proposed, including behavior- and affect-based [12,13], market-based [14–16], vacancy-chain [17], and swarm-intelligence [4,7,18] methods.

The contract-net protocol (CNP) is one of the most popular market-based task-planning methods, and it is widely applied to solve communication and control problems for MAS [16,19–21]. The CNP is known for its planning efficiency with explicit communications among agents, hence being suitable for MSS. In the CNP, a contract is signed through a process of mutual selection based on two-way information transfer. The contract net consists of a number of nodes, each of which represents an agent, and can be classified into two roles, manager and contractor. The former formulates the bidding task, publishes and collects tenders, evaluates which contractor wins the bidding, and lastly signs the contract. The latter focuses on bidding on the basis of its own working status and ability. Traditionally, the CNP requires heavy communication among agents to achieve dynamic task planning. In recent years, researchers developed a variety of remedies and enhancements to reduce the tender space in order to reduce communication and computation burden [16,22–24]. For example, a clustering method was proposed to reduce the number of bids, thereby reducing the computational burden [23]. In [24], machine-learning methods were employed to quickly initialize a feasible solution of a task-assignment problem of image-sensing satellites. Despite these improvements, how to develop effective and customized methods for SSA task-planning problems is still an open question.

Task-planning problems for SSA and for widely studied Earth observation applications, e.g., remote imaging and sensing, are different. In Earth observation applications, tasks are usually defined as observing a fixed ground station or a certain ground area within a time window [11,16,24], while SSA applications deal with dynamic objects in near-Earth space. These objects move with high velocity along orbits, such as space debris, or along a complex trajectory, such as a parabola for rockets. Task planning for Earth observation applications usually follows a single-to-multiple allocation pattern, i.e., a single satellite is allocated multiple tasks, while for SSA applications, a multiple-to-multiple allocation pattern is required. In this context, a satellite may need to supervise multiple objects, and an object may be supervised by multiple satellites.

To address these issues, in this paper, we propose a hierarchical and distributed task-planning framework with two layers. On the top layer, a global planner decomposes satellites and objects into multiple clusters using hard and soft clustering, respectively. This way of clustering leads to unique satellite clusters, but intersected object clusters, i.e., an object may belong to multiple object clusters. Such clustering is compatible with the multiple-to-multiple allocation pattern. Preprocessing planning is conducted using the CNP to allocate object clusters to satellite ones. On the bottom layer, local planners in each satellite cluster employ the CNP to allocate tasks. In the preprocessing and local levels, the criterion for evaluating tenders was designed as a combination of distance and relative velocity among satellites, thus taking into account fast-moving objects in SSA applications. In addition, a customized discrete particle swarm optimization (DPSO) algorithm was developed to search for the optimal task-planning result in the CNP. Tasks are allocated on the basis of their priorities, hence allowing for the execution of high-priority tasks even if the algorithm prematurely stops. By constraining the number of particles for searching the optimal solution, constraints on the maximal number of tasks that each satellite can execute are implicitly considered without increasing the computational burden.

To achieve fast task replanning in the case of urgent tasks, an urgent task planner is introduced to deal with urgent tasks in an event-triggered manner. Once urgent tasks come, the urgent planner quickly allocates them to clusters that are already determined by the global planner on the basis of the proximity between objects and satellites. Then, two methods are proposed to replan local tasks within each cluster. The first is to replan tasks using the CNP from scratch to warrant planning optimality at the cost of higher computational cost. The second method exploits task priorities to replan part of tasks to increase planning efficiency. To verify the proposed framework, a simulation using the MATLAB CubeSat Simulation Library was conducted. Results showed that the proposed planning framework is capable of generating multiple-to-multiple task-planning patterns, and is significantly more computationally efficient when compared to centralized CNP methods. Moreover, the consideration of both relative distance and velocity during planning reduces the overall distance among satellites and objects in consecutive sampling instants.

2. Problem Description

Assuming that there are in total n satellites and m objects to be planned with $m \gg n$ in typical SSA applications, for a satellite s , two parameters are defined as

$$s(\mathbf{p}, \mathbf{v}) = g(a, e, \theta, \Omega, \omega, v), \quad (1)$$

where \mathbf{p} and velocity \mathbf{v} are 3D position and velocity vectors in the Earth-centered-Earth-fixed (ECEF) reference frame that can be obtained by a function g using 6 Keplerian orbital elements given a specific time epoch [25]. The Keplerian orbital elements are: semimajor axis a , eccentricity e , inclination θ , right ascension of ascending node Ω , argument of periapsis ω , and true anomaly v . Similarly, 3 parameters are defined for object t as

$$t(\mathbf{p}, \mathbf{v}, P) = g(a, e, \theta, \Omega, \omega, v), \quad (2)$$

where the first two are the same as those of the satellite, and the last parameter P is the priority of the task for observing the corresponding object. Throughout the paper, we assumed the following for the working satellite.

Assumption 1. *Every working satellite is able to execute at least one task at any time.*

Assumption 2. *If a satellite malfunctions or loses connection, it is eliminated from the MSS and no longer participates in the task-planning and -execution process.*

Task planning is a problem to decide a match among satellites and objects. Cost function

$$\mathcal{L} = \sum_{i=1}^n \sum_{j=1}^m \alpha_{ij} h(s_i, t_j) \quad (3)$$

is defined as the sum of a planning criterion h for all satellites and objects. Defining $\alpha_{ij} = 1$ the match between satellite i and object j , and $\alpha_{ij} = 0$ for cases without match, the task-planning problem is to minimize \mathcal{L} over α_{ij} .

In this paper, resource, communication, and distance constraints were not considered when minimizing (3). The inclusion of these constraints certainly increases the complexity of the problem, which is usually solved by computationally demanding algorithms such as mixed-integer [10] and swarm intelligence optimization [18]. While we focused on providing the multiple-to-multiple pattern among satellites and objects, the design of criterion h can implicitly fulfil part of these constraints. For example, the proximity of satellites can be part of h , and the task-planning results would exclude matches that are too far away.

3. Task-Planning Framework

In this section, the proposed hierarchical and distributed task-planning framework of MSS for SSA applications is presented. Figure 1 shows that the framework consists of two layers, where a global planner on the top layer is responsible for periodically decomposing satellites and objects into clusters. Then, a preprocessing-planning step is conducted to perform cluster-to-cluster task planning. Lastly, in each satellite cluster, a local planner periodically allocates objects to satellites using the CNP. In addition, an urgent planner who has complete knowledge of all satellite clusters is introduced. When an urgent task appears, the urgent planner decides to which satellite cluster the urgent task belongs and then activates task replanning within the corresponding local cluster. Below, the global, local, and urgent task planner is described in detail.

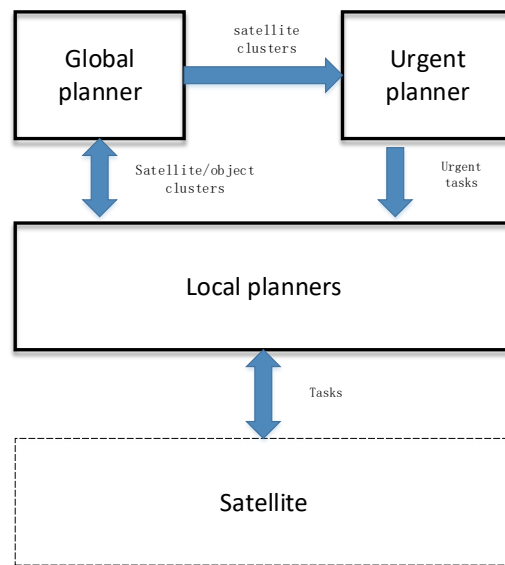


Figure 1. Hierarchical mission-planning framework of distributed satellite systems for space situational awareness (SSA).

3.1. Global Task Planner

The global planner, either a satellite in the MSS or a ground station, is aware of all other satellites and objects. To avoid the heavy computation and communication required by CNP in large-scale systems, the global planner first decomposes satellites into n_s unique clusters using canonical k -means++ hard-clustering algorithm [26]. The algorithm computes a centroid for each cluster and minimizes the sum of point-to-centroid distances summed over all n_s clusters. Parameter n_s is tuned on the basis of the total number of satellites and their distribution in near-Earth space.

To achieve multiple-to-multiple planning pattern, the objects are decomposed into n_t clusters using the Gaussian mixture model (GMM) soft-clustering method [27,28]. Unlike k -means++ clustering, the GMM method estimates cluster membership posterior probabilities, and then assigns each object to the cluster corresponding to the maximum posterior probability. Since the output of GMM is probability, an object may belong to multiple clusters and leads to multiple-to-multiple planning results. In this work, object proximity was used to build the GMM for such soft clustering. Number of clusters n_t is a tuning parameter that can be examined a posteriori for optimality.

After satellite and object clustering, the global planner works as the manager to allocate n_t object clusters to n_s satellite ones with a period of T_{global} . In detail, the global planner works through the following steps:

1. Tender publication. The global planner broadcasts centroids of all object clusters as

$$\langle AID_g, t_j \rangle, \quad (4)$$

where AID_g is the global manager.

2. Bidding. According to Assumption 1, one satellite from each satellite cluster bids for at least one tender on the basis of its own status and constraints once the task cluster is received. In general, the manager satellite in a cluster is selected depending on satellite ability, such as energy and computation power. Hence, the role of the manager and contractor is not fixed, and can be dynamically switched back and forth. In this paper, we assume satellites are homogeneous and select the one closest to cluster centroid as the cluster manager. The bidding can have three results: reject, not responding, and bidding. If a satellite decides to bid for a certain task cluster, the bidding can be expressed as

$$\langle bidder_j, B_{i,j} \rangle, \quad (5)$$

where $bidder_j$ is the bidding satellite cluster, and $B_{i,j}$ is the cost for the i -th satellite cluster bidding for the j -th object cluster, defined as

$$B_{i,j} = \begin{cases} \infty, & \text{reject,} \\ \infty, & \text{not responding,} \\ 0, & \text{bidding.} \end{cases} \quad (6)$$

3. Task cluster allocation. When the manager receives bidding or the bidding deadline is over, n_t task clusters are allocated to n_s satellite clusters by minimizing the cost function (3) as

$$\alpha_{cluster}^* = \arg \min \sum_{i=1}^{n_s} \sum_{j=1}^{n_t} \alpha_{ij} (h_c(\bar{s}_i, \bar{t}_j) + B_{ij}), \quad (7)$$

where h_c is the cluster tender-evaluation criterion, \bar{s}_i, \bar{t}_j are average parameter values, such as centroids and mean velocities for the i -th satellite cluster and the j -th object cluster. In this paper, optimal match $\alpha_{cluster}^*$ was obtained by developing a discrete particle swarm optimization (DPSO) algorithm [29] with task priority P embedded. Algorithmic details are given in Section 4. Lastly, we have

$$\begin{aligned} \{\bar{t}_1\}, \{\bar{t}_2\} \dots, \{\bar{t}_{l_1}\} &\rightarrow \bar{s}_1, \\ \{\bar{t}_{l_1+1}\}, \{\bar{t}_{l_1+2}\} \dots, \{\bar{t}_{l_1+l_2}\} &\rightarrow \bar{s}_2, \\ \dots, & \end{aligned} \quad (8)$$

where one satellite cluster is allocated with multiple object clusters, and the number of satellites and objects in each cluster is different based on clustering results.

3.2. Local Task Planner

The local task planner is a satellite belonging to a satellite cluster. The local planner allocates tasks from multiple object clusters in (8) to satellites in the satellite cluster in a period of $T_{local} \ll T_{global}$ to quickly adapt to external disturbances and urgent tasks. The local planner employs CNP and DPSO algorithm as follows:

1. Tender publication. Similar to the global planner, the local planner publishes tenders to local satellites in the same cluster as

$$\langle AID_l, t_j \rangle \quad (9)$$

where AID_l is the local manager.

2. Bidding. This step is exactly same as Step 2 in the global planner, except that only local satellites bid for the tenders.

- Task allocation. When the local manager receives bidding or the bidding deadline is over, $n_{t,l}$ local tasks are allocated to $n_{s,l}$ satellites as

$$\alpha_{local}^* = \arg \min \sum_{i=1}^{n_{s,l}} \sum_{j=1}^{n_{t,l}} \alpha_{ij} (h_l(s_i, t_j) + B_{ij}), \quad (10)$$

where h_l is the local tender-evaluation criterion, designed as

$$h_l = a \|s_i(\mathbf{p}) - t_j(\mathbf{p})\|_2^2 + b \|s_i(\mathbf{v}) - t_j(\mathbf{v})\|_2^2. \quad (11)$$

Unlike cluster planning criterion (12), the local tender-evaluation criterion is a weighted combination of relative position and velocity among satellites and objects. The rationale is that a cluster centroid is a representation of multiple satellites or objects that covers a large space area; hence, the moving of satellites and objects has fewer effects on how a cluster is planned. However, on the local level, the moving of satellites and objects must be taken into account due to their high velocity in a limited space area.

- Contract signed. Similarly to the global planner, the local planner employs the iterative DPSO algorithm to obtain (10). The contract is signed, and the task-planning problem is completed.

3.3. Urgent Task Planner

The urgent task planner operates in an event-triggered manner to achieve fast urgent task planning. As shown in Figure 1, the urgent planner receives satellite and object cluster information (8) from the global planner, and assigns urgent tasks to object clusters with minimal distance between urgent objects and cluster centroids. After receiving the urgent tasks, the local planner in this cluster activates a task-replanning procedure to allocate these urgent tasks to local satellites.

In this work, we propose two local task-replanning methods as follows. The first method is to replan local tasks by performing the CNP from scratch following Steps 1–4 of the local planner. This method guarantees that the local replanning procedure is optimal in terms of the cost function (10) when urgent tasks are detected.

The second method is to replan local tasks on the basis of their priorities P . The idea is to further decompose the local satellite cluster into two parts: one s_{high} consisting of satellites of which the lowest priority task has a higher or equal priority than that of the urgent task, and the other s_{low} consisting of the remaining satellites in the cluster. The main steps of this method are as follows.

- Extract priorities P of the urgent tasks.
- Decompose satellite cluster into s_{high} and s_{low} parts.
- Plan urgent tasks to satellites in the s_{low} part following Steps 1–4 of the local planner.

Remark 1. *The two task-replanning methods can be applied in different scenarios. If one concerns the optimality of local planning, and computational power is sufficient for complete replanning, the first method is desired. On the other hand, if extremely fast task replanning is required for urgent tasks, the second method is more suitable. In addition, the second method guarantees that tasks of highest priorities are executed without replanning for safety or timeliness reasons.*

Remark 2. *The proposed hierarchical framework periodically updates task-planning results in accordance with global update period T_{global} and local update period T_{local} . As a result, a satellite supervises an object only in a period of time during which the two are close enough for feasible supervision. When satellite and object move apart until the distance between them is longer than the sensor range is, the satellite starts supervising newly planned objects within the sensor range, and the object is supervised by other satellites. The selection of T_{global} and T_{local} depends on various factors such as sensor range, and satellite and object velocity.*

Example 1. To better present how the proposed task-planning method works, an example is shown in Table 1 with 5 satellites and 12 objects, with positions randomly generated from Keplerian orbital elements. Clustering results using the *k*-means++ and GMM soft-clustering algorithm for 3 satellite clusters and 8 object clusters based on their proximity are shown in Table 2. Satellite Cluster 2 contains 3 satellites, and the 2 other satellite clusters have only one satellite. Object Clusters 2 and 6 have 2 objects, and Object 9 belongs to both Object Cluster 5 and 7, complying with the multiple-to-multiple planning pattern. Given the clustering results, Step 3 of the global planner was performed, and results are shown in Table 3. Tender-evaluation cost function h_c in (7) for the global planner was designed as the distance between a satellite cluster centroid and an object cluster centroid as

$$h_c = \|\bar{s}_i(\mathbf{p}) - \bar{t}_j(\mathbf{p})\|_2^2. \quad (12)$$

On the local level, we chose $a = 1, b = 500$ for local tender-evaluation criterion (11) to balance the magnitude of satellite position and velocity: the position vector is usually in the range of $10^6 - -10^7$ m, while velocity is around 10^3 m/s. The local planning results of all clusters are shown in Table 4 and animated in Figure 2. In this example, since the number of satellites and objects was small, the clustering result is unique even though the GMM soft-clustering method was employed. The planning results using a baseline CNP without the proposed hierarchical structure, and satellite and object clustering are shown in Figure 3. The two planning methods generated slightly different results, but in principle, objects were allocated to nearby satellites. As is shown in Section 5, the two methods have advantages and disadvantages in terms of computational efficiency and planning cost.

Suppose that Urgent Task 13 requires supervising an object at position $(-10, 206, 197, 2716)$ km. By computing the distances between the urgent object and the centroids of the three clusters given in Table 4, the urgent object can be assigned to Cluster 3. A task-replanning procedure following the steps of the local planner leads to an update of the task-planning result of Cluster 3 shown in Table 5 and Figure 4. The urgent object is allocated to the closest, Satellite 4. Complete replanning for all satellites and objects is avoided.

Suppose that an urgent task Object 13 has a priority $P_{13} = \text{medium}$, and satellites in Cluster 3 have tasks with priorities $P_1 = \text{high}$, $P_4 = \text{high}$, and $P_5 = \text{low}$. Using the second task-replanning method, Cluster 3 is decomposed into two parts, $s_{\text{high}} = \{1, 4\}$ and $s_{\text{low}} = \{5\}$. Urgent Task 13 is, therefore, allocated to Satellite 5 immediately without local replanning from scratch for 3 satellites and 5 tasks.

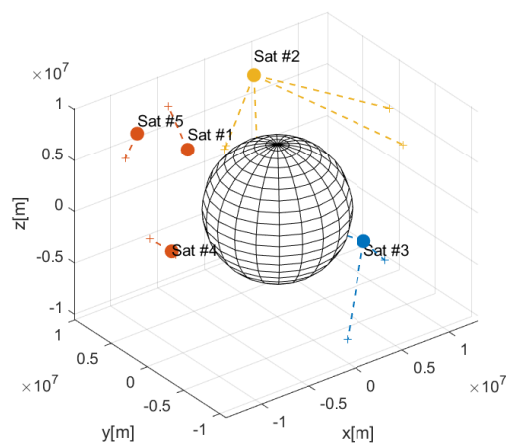


Figure 2. Task-planning animation for results presented in Table 4. Filled circles, satellites; '+', objects. Blue, red, and yellow represent Clusters 1–3, respectively.

Table 1. Position coordinates of randomly generated 5 satellites and 12 objects in Earth-centered-Earth-fixed (ECEF) reference frame.

No.	x [km]	y [km]	z [km]
Satellites:			
1	−7883	2176	7598
2	2269	6213	9469
3	7091	−2794	−4344
4	−12,203	−1237	738
5	−8294	8632	6378
Objects:			
1	−11,304	−640	−220
2	−7350	5563	10,105
3	−13,852	−373	2120
4	4049	7363	−10,494
5	60	−9711	−8224
6	10,512	−1976	7099
7	−9299	8989	4098
8	5793	4092	−5105
9	2254	10,094	791
10	616	8124	1947
11	13,052	−573	2002
12	3138	−10,907	−1095

Table 2. Satellite- and object-clustering results using *k*-means++ and Gaussian mixture model (GMM) soft-clustering method, respectively.

Satellite cluster no.	1	2	3					
Satellite no.	2	3	[1,4,5]					
Object cluster no.	1	2	3	4	5	6	7	8
Object no.	12	[1,3]	4	[2,7]	5	[9,10]	8	[6,11]

Table 3. Satellite- and object-cluster planning results.

Satellite cluster no.	1	2	3
Object cluster no.	[3,6,8]	[1,5,7]	[2,4]

Table 4. Final task-planning results of the example.

	Cluster 1	Cluster 2	Cluster 3		
Satellite no.	2	3	1	4	5
Object no.	[4,6,9,10,11]	[5,8,12]	3	1	[2,7]

Table 5. Updated task-planning results of example in case of urgent task using first task-replanning method.

	Cluster 1	Cluster 2	Cluster 3		
Satellite no.	2	3	1	4	5
Object no.	[4,6,9,10,11]	[5,8,12]	3	[1,13]	[2,7]

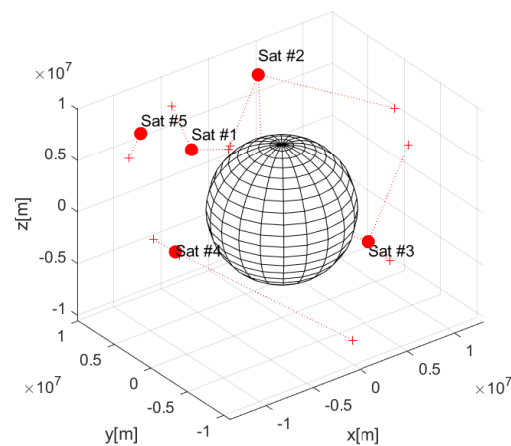


Figure 3. Task-planning animation for example using baseline flat contract-net protocol (CNP) without hierarchical architecture. Filled circles, satellites; '+', objects.

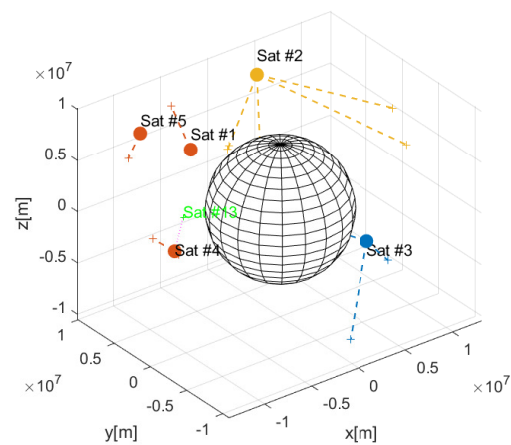


Figure 4. Updated task-planning animation for results presented in Table 5. Filled circles, satellites; '+', objects. Blue, red, and yellow represent Clusters 1–3, respectively. Urgent task represented as magenta '+' that is allocated to Satellite 4.

4. Tender-Evaluation DPSO Algorithm

In this section, a tender-evaluation method based on a customized DPSO is described that was developed for task-planning problems in SSA applications. Comparing with the canonical DPSO algorithm, as in [30], the customized DPSO algorithm was designed by considering two additional requirements:

1. tasks of high priorities are allocated first; and
2. there is an upper bound on the maximal number of tasks with which each satellite can be allocated.

The first requirement ensures that tasks of high priorities can be allocated first if the algorithm prematurely stops due to limited computational resources or external disturbances. The second requirement is a constraint on satellite abilities to execute tasks that are not explicitly included in optimization Problems (7) and (10) to avoid computationally expensive constrained mixed-integer optimization.

In the developed algorithm, the position of each particle represents a solution to the task-planning problem. The mechanism of DPSO is that particles form a particle group and search for their optimal positions. In implementation, objects are first clustered into multiple batches, each of which contains objects with the same priority P . The DPSO algorithm allocates objects in a batch to satellites and repeats this procedure to all batches. Suppose that the satellites and a batch of objects are encoded by natural numbers from 1 to

n and 1 to m_b , respectively, where m_b is the number of objects in a batch. Particle position is a vector of length l_p in which the j -th element has a value of i , meaning that the j -th object is assigned to the i -th satellite. In addition, constraint vector $\alpha = [\alpha_1, \alpha_2, \alpha_3, \dots, \alpha_n]$ is defined to indicate the maximal number of tasks that each satellite can execute, i.e., $\alpha_i = k$ means that at most k objects can be allocated to the i -th satellite. When $l_p > m$, elements that are encoded by numbers greater than m in the particle position vector do not play a role in computing the tender-evaluation cost.

An example of particle vector encoding is given in Figure 5. Assuming that the number of tasks in the batch is 5 and the number of satellites is 3, we also assume that the maximal number of tasks for the 3 satellites are

$$\alpha = [3, 2, 1]. \quad (13)$$

In the beginning, particle position is randomly initialized with length $l_p = 6$ as $X_i = [2, 3, 1, 1, 2, 1]$, which means that Object 1 is assigned to Satellite 2, Object 2 assigned to Satellite 3, etc. In this work, the particle position vector is always initialized in a way in which Constraint (13) is fulfilled. Operations in later iterations only affect the index of the elements in the position vector, but not their values, thereby obeying Constraint (13) at all iterations.

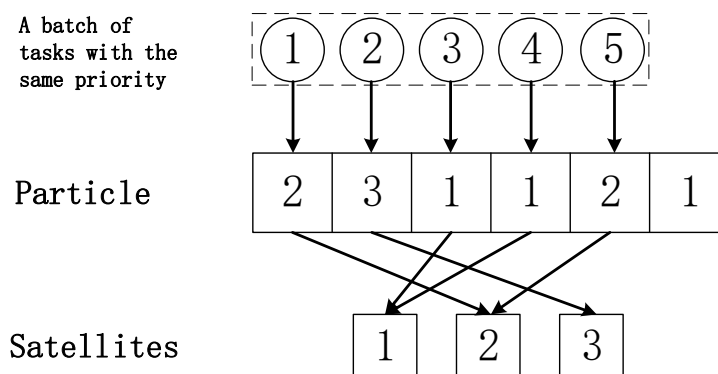


Figure 5. Particle encoding in discrete particle swarm optimization (DPSO) algorithm for multiple-satellite and -object task-planning problems.

After initialization, the position of each particle in the particle group is iteratively updated by an updating law to reduce the cost of the corresponding optimization problem. In [30], a DPSO updating law suitable for multisatellite task-planning problems was proposed that considers three factors: (i) the current position of the particle, (ii) the optimal historical record of the particle position, and (iii) the historical optimal record of the position of the particle group. The updating law can be written as

$$X_i(k+1) = X_i(k) \otimes (c_1 * X_i(k)) \oplus (c_2 * P_i(k)) \oplus (c_3 * P_g(k)), \quad (14)$$

where $X_i(k)$ is the position of the i -th particle at the k -th iteration, $P_i(k)$ is the optimal historical record of the i -th particle position, and $P_g(k)$ is the historical optimal record of the position of the particle group. c_1 , c_2 , and c_3 are the inferior operation thresholds [30] corresponding to the three factors, respectively.

The updating law can be interpreted as follows. First, an “inertial” operation of the particle, i.e., the position of the particle, is updated by adjusting the internal structure of the particle. This operation is expressed as “ \otimes ”, generating two random integer numbers c and d within $[1, l_p]$ and then exchanging the c -th element and d -th element in the particle vector. The second and the third operations are the “self-recognition” and “social cognition” of particles, respectively. A particle adjusts its position according to its optimal record P_i and the particle group optimal record P_g . The operation is represented as “ \oplus ”. In this operation, a random integer e within $[1, l_p]$ is generated so that the e -th element of the

vector is exchanged with the element that has the same value as that of the e -th element of its historically optimal position vector P_i or P_g . There may be multiple elements with the same value in the particle position vector. However, the maximal number of these elements is constrained by α from the initialization, and the element for exchange can be randomly selected. Readers can refer to [30] for more algorithmic details. Rigorous mathematical demonstrations of the complexity and scalability of the algorithm are still an open question and usually tested using specific benchmark problems [31,32].

5. Simulation

In this section, we first present a static task-planning result, i.e., task planning at a certain time instant. Then, we show dynamic planning where the global and local planners update their planning results in the period of T_{global} and T_{local} , as described in Section 3. The simulation platform was MATLAB 2019b, running on a PC with i5-9600KF CPU and 32 GB memory. The MATLAB CubeSat Simulation Library was used to generate the random positions and velocities of satellites and objects around the near-space of Earth.

5.1. Static Planning

We first show the performance of the proposed task-planning framework for tens of satellites and hundreds of objects in a static scenario, where task planning is performed at a single time instant. In this simulation, 20 satellites and 200 objects were randomly generated in the near-space of Earth with different Keplerian orbital elements. As shown in Figure 6, the proposed distributed task-planning framework decomposed the satellites into 5 clusters and the objects into 12 clusters. The number of satellites and objects in each cluster is shown in Table 6. Results clearly show that satellite clusters were determined by their proximity. In this simulation, 34 objects were allocated to multiple satellites and are labeled by black circles.

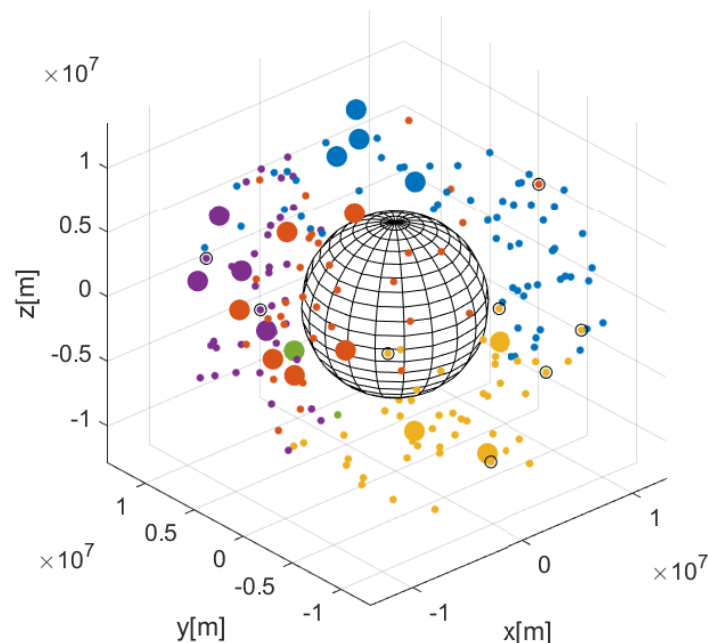


Figure 6. Task-planning result for 20 satellites and 200 objects with 5 and 12 clusters, respectively. Filled circles, satellites; crosses, objects; black circles, objects allocated to multiple satellites.

Table 6. Number of objects allocated to each satellite cluster. There were 34 objects allocated to multiple satellites. The total number of objects is 200.

Cluster No.	1	2	3	4	5	Total
No. of Satellites	6	4	4	3	3	20
No. of Objects	29	87	54	44	20	234

The computational cost, dominated by GMM clustering and solving optimization Problems (7) and (10) using the DPSO algorithm, was compared against a baseline flat CNP method that allocates all tasks to all satellites at once, and does not consist of a hierarchical structure of global and local planners. Figure 7 shows a comparison for 20 satellites with 3 clusters. Objects were decomposed into 8 clusters. The result demonstrated that the proposed framework had better scalability than that of the baseline flat CNP, especially when considering that space objects such as debris number in the tens of thousands, or even more. The proposed framework has higher computational time when the number of objects is small because the clustering process has little or even negative effect on computational time when the scale of a problem is rather small.

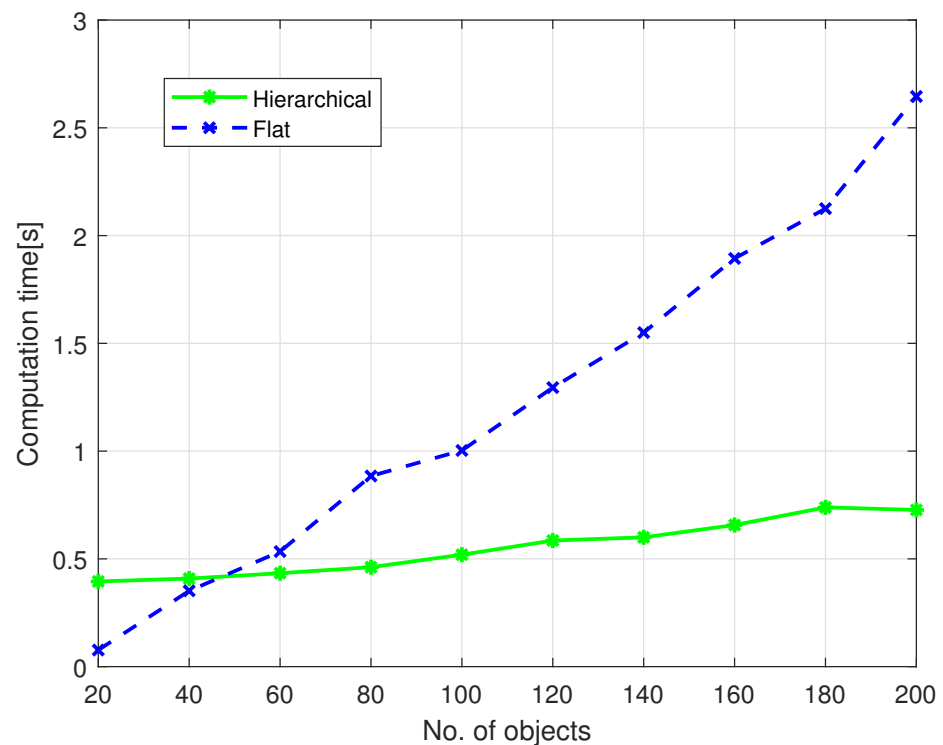


Figure 7. Computational time comparison between baseline flat CNP and proposed hierarchical task-planning framework for a single task planning. There were 20 satellites with 5 clusters. Objects were decomposed into 12 clusters.

In the static case, the task planning cost of the proposed hierarchical framework and the baseline flat CNP, represented by the cost function (3), is compared in Figure 8. The proposed framework has slightly higher planning cost than that of the baseline CNP. The task planning cost grows with the number of tasks and the gap between the proposed framework and the baseline one also increases. However, such increased planning cost can be accepted considering the significant computational time gain shown in Figure 7.

To further understand how the number of clusters affect computational time and task-planning cost, we put these two factors of the proposed framework in Figure 9. A clear trade-off can be observed between computational time and task-planning cost.

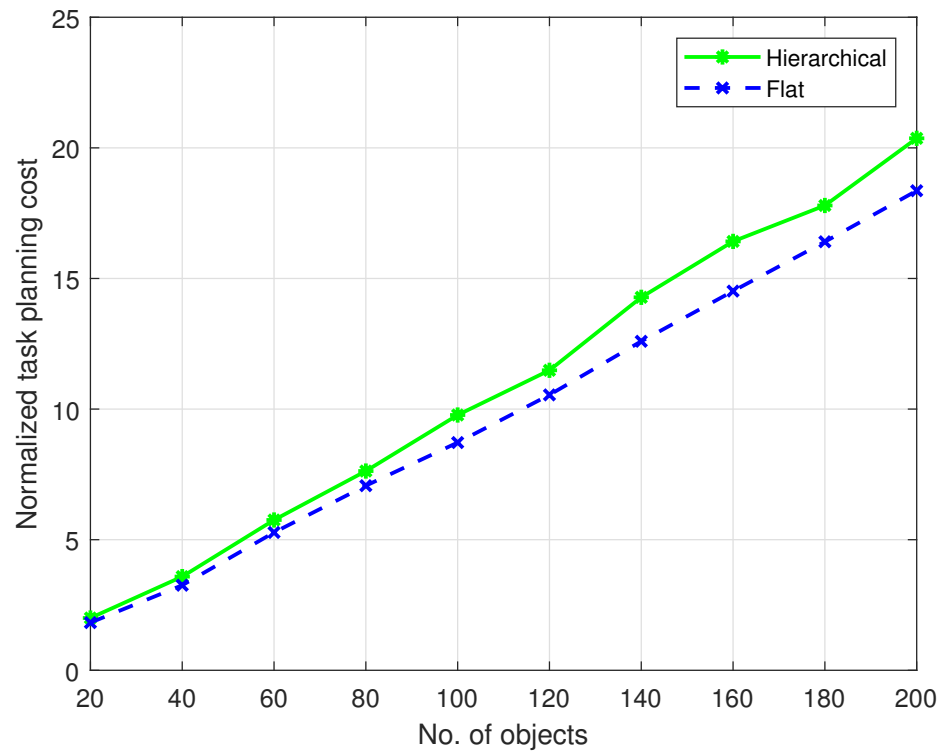


Figure 8. Normalized task-planning cost of baseline flat CNP and proposed task-planning framework in the static case. There were 20 satellites with 5 clusters. Objects were decomposed into 12 clusters.

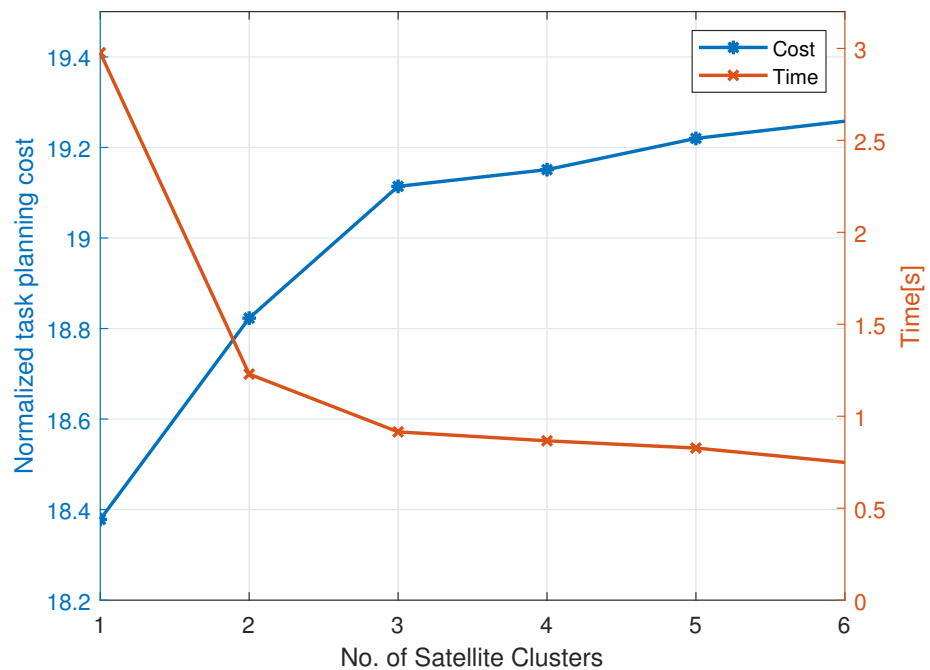


Figure 9. Computational time and normalized task-planning cost with regard to number of satellite clusters in the static case. Total numbers of satellites and objects were 20 and 200, respectively.

5.2. Dynamic Planning

In the dynamic-planning scenario, the global and local planners update their planning results in periods of T_{global} and T_{local} , respectively. The baseline flat CNP also updated its planning in a period of T_{flat} . In this scenario, we performed a 2 h orbital simulation, collected the position and velocity information every 144 s, and computed the correspond-

ing planning cost for all tasks at each time instant. The planning update period was set to be $T_{global} = 2T_{local} = 288$ s. Figure 10 shows the relative planning cost C of the two methods as

$$C = cost_{hierarchical} / cost_{flat}$$

with regard to different period ratios

$$r = T_{flat} / T_{local}.$$

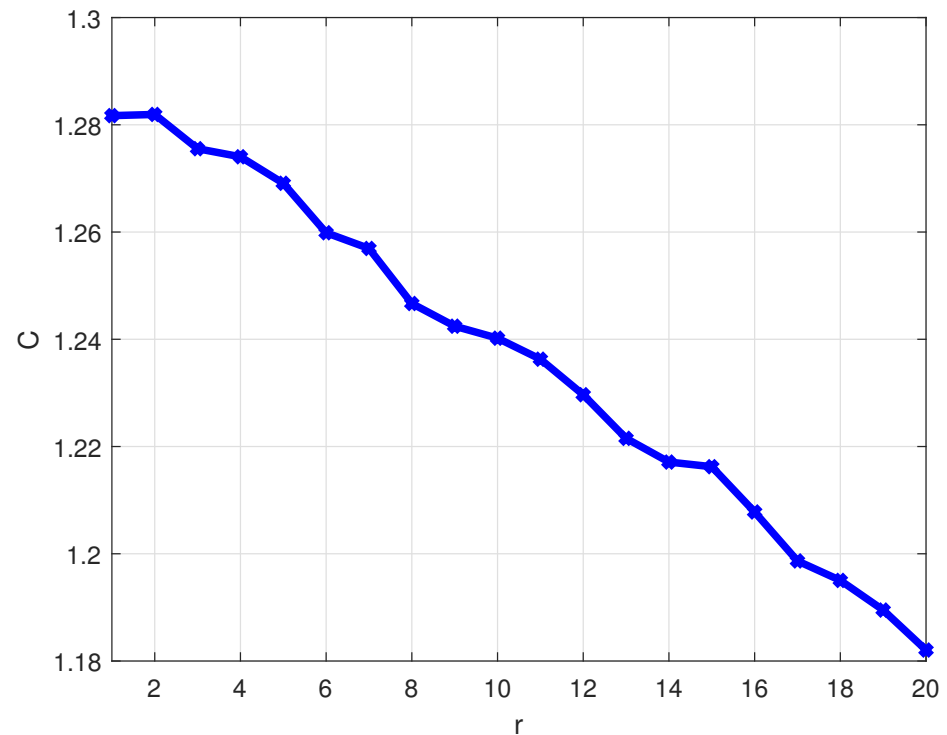


Figure 10. Task-planning cost ratio C with regard to different period ratios r in dynamic-planning scenario. There were 20 satellites with 5 clusters. The 200 objects were decomposed into 12 clusters.

Relative planning cost with regard to the number of objects is shown in Figure 11. The planning-cost gap between the proposed method and the flat CNP decreased as the proposed method updated its planning result faster than the baseline CNP did. The planning-cost gap also decreased as the number of objects grew, demonstrating the scalability of the proposed algorithm. Therefore, the proposed method approached the baseline CNP when it updated faster for large-scale problems, showing potential for practical deployment.

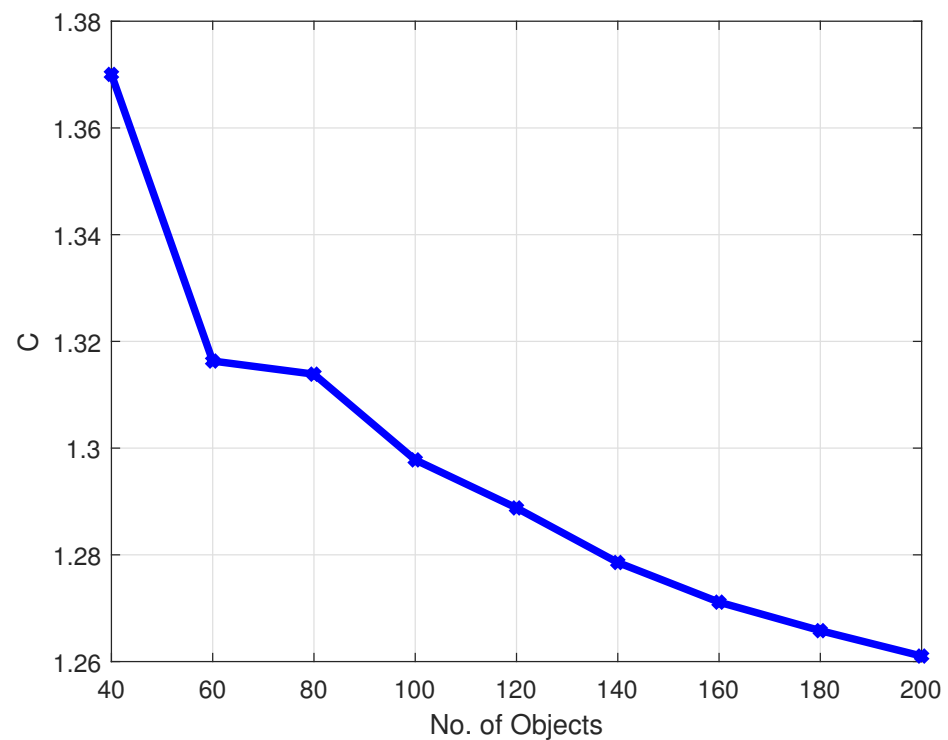


Figure 11. Task-planning cost ratio C regarding number of objects in dynamic-planning scenario with $r = 5$. There were 20 satellites with 5 clusters. Objects were decomposed into 12 clusters.

6. Conclusions

This paper proposed a hierarchical and distributed task-planning framework customized for multiple-satellite SSA systems. A global planner decomposes multiple satellites and tasks into clusters where local planners allocate tasks to local satellites using the CNP. The framework introduced an urgent task planner for fast task replanning in the case of urgent tasks. A customized DPSO algorithm was developed to effectively and efficiently find optimal planning results. Simulation results in static and dynamic scenarios demonstrated that the proposed framework is effective and efficient in large-scale task-planning and -replanning problems.

Author Contributions: Methodology, Y.C. and G.T.; software, Y.C. and G.T.; investigation, Y.C., G.T., and J.G.; writing—original-draft preparation, Y.C. and G.T.; writing—review and editing, Y.C. and J.H.; supervision, J.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded in part by the Science and Technology on Space Intelligent Control Laboratory, no. 6142208190203.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kennewell, J.A.; Vo, B.N. An overview of space situational awareness. In Proceedings of the 16th International Conference on Information Fusion, Istanbul, Turkey, 9–12 July 2013; pp. 1029–1036.
2. Jian, P.; Xiong, W.; Li, Z. Research on mission planning of space situational awareness. In Proceedings of the 4th China Conference on Command and Control, Beijing, China, 4 July 2016; pp. 520–523.
3. Du, Y. Research on Key Technologies of Space Situational Awareness Based on Binocular Vision Satellite Formation. Master's Thesis, Zhengjiang University, Hangzhou, China, 2020.
4. Zheng, Z.; Guo, J.; Gill, E. Distributed onboard mission planning for multi-satellite systems. *Aerosp. Sci. Technol.* **2019**, *89*, 111–122. [[CrossRef](#)]
5. Khamis, A.; Hussein, A.; Elmogy, A. Multi-robot task allocation: A review of the state-of-the-art. In *Cooperative Robots and Sensor Networks 2015*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 31–51.

6. Lv, C. A System for Space Situational Awareness (Ssa) Based on Centralized Control. Ph.D. Thesis, Beijing University of Posts and Telecommunications, Beijing, China, 2013.
7. Zheng, Z.; Guo, J.; Gill, E. Onboard autonomous mission re-planning for multi-satellite system. *Acta Astronaut.* **2018**, *145*, 28–43. [[CrossRef](#)]
8. Sui, Z.; Pu, Z.; Yi, J. Optimal uavs formation transformation strategy based on task assignment and particle swarm optimization. In Proceedings of the 2017 IEEE International Conference on Mechatronics and Automation (ICMA), Takamatsu, Japan, 6–9 August 2017; pp. 1804–1809.
9. Miloradović, B.; Çürüklü, B.; Ekström, M.; Papadopoulos, A.V. A genetic algorithm approach to multi-agent mission planning problems. In *International Conference on Operations Research and Enterprise Systems*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 109–134.
10. Afonso, R.J.; Maximo, M.R.; Galvão, R.K. Task allocation and trajectory planning for multiple agents in the presence of obstacle and connectivity constraints with mixed-integer linear programming. *Int. J. Robust Nonlinear Control* **2020**, *30*, 5464–5491. [[CrossRef](#)]
11. Chang, Z.; Chen, Y.; Yang, W.; Zhou, Z. Mission planning problem for optical video satellite imaging with variable image duration: A greedy algorithm based on heuristic knowledge. *Adv. Space Res.* **2020**, *66*, 2597–2609. [[CrossRef](#)]
12. Parker, L.E. Task-oriented multi-robot learning in behavior-based systems. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS'96, Osaka, Japan, 4–8 November 1996; Volume 3, pp. 1478–1487.
13. Gage, A. Multi-Robot Task Allocation Using Affect. Ph.D. Thesis, University of South Florida, Tampa, FL, USA, 2004.
14. Dias, M.B.; Zlot, R.; Kalra, N.; Stentz, A. Market-based multirobot coordination: A survey and analysis. *Proc. IEEE* **2006**, *94*, 1257–1270. [[CrossRef](#)]
15. Bogdanowicz, Z.; Coleman, N. Sensor-target and weapon-target pairings based on auction algorithm. In Proceedings of the 11th WSEAS International Conference on APPLIED MATHEMATICS, Dallas, TX, USA, 22–24 March 2007; pp. 92–96.
16. Du, B.; Li, S. A new multi-satellite autonomous mission allocation and planning method. *Acta Astronaut.* **2019**, *163*, 287–298. [[CrossRef](#)]
17. Dahl, T.S.; Matarić, M.; Sukhatme, G.S. Multi-robot task allocation through vacancy chain scheduling. *Robot. Auton. Syst.* **2009**, *57*, 674–687. [[CrossRef](#)]
18. Dos Santos, D.S.; Bazzan, A.L. Distributed clustering for group formation and task allocation in multiagent systems: A swarm intelligence approach. *Appl. Soft Comput.* **2012**, *12*, 2123–2131. [[CrossRef](#)]
19. Smith, R.G. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Trans. Comput.* **1980**, *12*, 1104–1113. [[CrossRef](#)]
20. Zhang, J.; Wang, G.; Song, Y. Task Assignment of the Improved Contract Net Protocol under a Multi-Agent System. *Algorithms* **2019**, *12*, 70. [[CrossRef](#)]
21. Yu, L.; Wu, X.; Mao, Y.; Gao, H.; Hao, Y. Task allocation for distributed remote sensing satellites based on contract network algorithm. *J. Harbin Eng. Univ.* **2020**, *41*, 1059–1065.
22. Zhou, Z.; Cheng, S.; Liu, Q. A novel contract net negotiation model based on trust mechanism. In Proceedings of the 2008 IEEE Conference on Cybernetics and Intelligent Systems, Chengdu, China, 21–24 September 2008; pp. 884–887.
23. Feng, P.; Chen, H.; Peng, S.; Chen, L.; Li, L. A method of distributed multi-satellite mission scheduling based on improved contract net protocol. In Proceedings of the 2015 11th International Conference on Natural Computation (ICNC), Zhangjiajie, China, 15–17 August 2015; pp. 1062–1068.
24. Song, Y.J.; Zhou, Z.Y.; Zhang, Z.S.; Yao, F.; Chen, Y.W. A framework involving MEC: imaging satellites mission planning. *Neural Comput. Appl.* **2020**, *32*, 15329–15340. [[CrossRef](#)]
25. Wikipedia Contributors. Orbital Elements—Wikipedia, The Free Encyclopedia. Available online: https://en.wikipedia.org/w/index.php?title=Orbital_elements&oldid=997827172 (accessed on 23 February 2021).
26. Arthur, D.; Vassilvitskii, S. k-means++: The advantages of careful seeding. In Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), New Orleans, LA, USA, 7–9 January 2007; pp. 1027–1035.
27. Yang, M.S.; Lai, C.Y.; Lin, C.Y. A robust EM clustering algorithm for Gaussian mixture models. *Pattern Recognit.* **2012**, *45*, 3950–3961. [[CrossRef](#)]
28. Zhao, Y.; Shrivastava, A.K.; Tsui, K.L. Regularized Gaussian mixture model for high-dimensional clustering. *IEEE Trans. Cybern.* **2018**, *49*, 3677–3688. [[CrossRef](#)] [[PubMed](#)]
29. Clerc, M. Discrete particle swarm optimization, illustrated by the traveling salesman problem. In *New Optimization Techniques in Engineering*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 219–239.
30. Wei, H.; Xue-qing, Z. A Multisatellite Task Planning Algorithm Based on Discrete Particle Swarm. *Radio Eng.* **2015**, *45*, 47.
31. Al-Dujaili, A.; Tanweer, M.R.; Suresh, S. On the Performance of Particle Swarm Optimization Algorithms in Solving Cheap Problems. In Proceedings of the 2015 IEEE Symposium Series on Computational Intelligence, Cape Town, South Africa, 7–10 December 2015; pp. 1318–1325. doi:10.1109/SSCI.2015.188. [[CrossRef](#)]
32. Al-Kazemi, B.S.; Habib, S.J. Complexity Analysis of Problem-Dimension Using PSO. In Proceedings of the 7th WSEAS International Conference on Evolutionary Computing, EC'06, Cavtat, Croatia, 12–14 June 2006; pp. 45–52.