Adaptive Planning Method for Operations of a Multi-satellite Swarm for Earth Remote Sensing in Real Time

Petr Skobelev[1][a], Elena Simonova[2][b], Vladimir Galuzin[3][c], Anastasiya Galitskaya[4][d] and Vitaly Travin[4][e]

[1]Samara Federal Research Scientific Center RAS, Institute for the Control of Complex Systems RAS, Sadovaya Str., 61, 443020, Samara, Russia
[2]Samara National Research University, Moskovskoye Shosse, 34, 443086, Samara, Russia
[3]Samara State Technical University, Molodogvardeyskaya Str., 244, 443100, Samara, Russia
[4] SEC «Smart Solutions», Moskovskoye Shosse, Office 1201, 17, 443013, Samara, Russia

Keywords: Small Spacecraft, Ground Stations, Space System, Observation Objects, Multi-agent Technologies, Adaptive Planning.

Abstract: The paper describes a method for adaptive planning of imaging operations for a multi-satellite swarm in real time, based on a multi-agent approach. The key object in this approach is the intelligent agent of an application for imaging of the observation object. Its goal is the most advantageous placement in the schedule. The solution to the optimization problem is obtained as a result of reaching an equilibrium point in multiple negotiations between agents through mutual compromises and concessions. The paper provides a brief problem statement of planning the operation of a multi-satellite swarm for Earth remote sensing (ERS). Furthermore, it describes the developed method, which makes it possible to process applications for imaging observation objects in real time. The paper also presents results of experimental studies that demonstrate efficiency of the developed multi-agent method in solving this problem versus traditional approaches. Finally, prospects for further development and practical application of the presented method are discussed.

1 INTRODUCTION

Development of a method for adaptive planning for multi-satellite swarms of small spacecrafts for remote sensing of the Earth (ERS) in real time is primarily relevant now due to the developing trend in the space industry aimed at creating, deploying and operating space systems (SS), including a multi-satellite (more than 100 spacecrafts) swarm of low-orbit satellites and a distributed network of ground stations for receiving information (GS). The purpose of creating such space systems is to meet the existing needs for remote sensing data, which are used in various fields: agriculture, geological and hydrological research, the military sphere, elimination of consequences of natural disasters, creating plans for certain territories, etc. (Shimoda, 2016).

Examples of such systems are the Planet Labs project with its satellite swarm of more than 200 operating Dove (Flock) satellites and 13 SkySat sub-meter satellites, and the BlackSky Global project, the orbital swarm of which consists of 60 Earth remote sensing satellites (Kopacz, 2020).

The consequence of such an increase in dimension and performance of the orbital swarm is the growth of requirements for algorithms and planning systems. Thus, for a SS consisting of dozens of satellites and GS, it may be required to draw up a plan of thousands of points for observations objects (OO) on a significant horizon, and the time for

[a] https://orcid.org/0000-0003-2199-9557
[b] https://orcid.org/0000-0003-2638-2572
[c] https://orcid.org/0000-0002-1460-613X
[d] https://orcid.org/0000-0002-7752-4262
[e] https://orcid.org/0000-0003-4084-418X

placing new applications for imaging should be measured in minutes from the moment of their arrival. The use of traditional systems for managing SS, which are based only on the ground control loop and traditional planning methods, together with multi-satellite orbital swarms will lead to conflict situations when several small satellites simultaneously claim to perform imaging of the same OO, or to transmit data to the same GS (Karsaev, 2016). Increased requirements for operational efficiency of processing the applications for imaging propels implementation of a dynamic adaptive adjustment of the SS work schedule as new applications enter the system or in case of unpredictable events associated with equipment failure or rapidly changing meteorological conditions.

One of the possible ways to overcome the above limitations inherent in traditional scheduling algorithms is the extended use of multi-agent technology (Rzevski, 2014), on the basis of which the method described in this paper has been developed. Their application at the moment has already proven itself in various industrial solutions (Gorodetsky, 2020). Multi-agent technology makes it possible to flexibly and adaptively synthesize a schedule in real time, taking into account individual characteristics of orders and resources. Besides, the underlying actor computation model makes it possible to create high-performance, distributed, fault-tolerant solutions, which all together helps more efficiently manage SS resources in comparison with traditional methods.

The paper is structured as follows. The second chapter provides a brief problem statement for planning operations of a multi-satellite ERS group. The third chapter describes the current state of research and development on this problem. The fourth chapter describes the data model used with description of the main classes, the fifth chapter considers the developed method of adaptive planning. The sixth chapter describes the carried out experimental studies. Finally, the seventh chapter summarizes the main results and discusses the prospects for development and application of the solution.

2 PROBLEM STATEMENT

The task of planning operations for a multi-satellite ERS swarm can be presented as follows. Let there be a simplified model of a spacecraft, which is a combination of two segments: a space complex and a ground-based special complex. The space complex performs the functions of receiving and transmitting information, the ground-based special complex - the functions of receiving and processing the transmitted information.

The space complex consists of a set of satellites $S = \{s_i\}, i=\overline{1,L}$. Each spacecraft s_i is characterized by a set of orbital elements and parameters of onboard equipment. In its turn, the ground complex is represented by a set of GS $G_R = \{g_r\}, r = \overline{1,R}$. Each g_r station is characterized by its geographic location and parameters of installed antenna. For GS and satellites, restrictions may be indicated in the form of a work schedule and unavailability intervals.

The space system must ensure fulfillment of a set of applications for imaging point and area observation objects $O = \{o_p\}, p = \overline{1,P}$. For the imaging application o_p, its priority pr_p can be specified (an application with a low priority should not interfere with the optimal location of a higher-priority application) and a set of restrictions: the point in time until which it is necessary to obtain images t_p^{end}, the balance coefficient between efficiency and quality of the information received c_p (set in the range from 0 to 1), the minimum and desired linear resolution of the resulting image $minR_p$ and $maxR_p$. Each application for imaging o_p, depending on its type, corresponds to one or more areas of surveying $SA = \{sa_j\}, j = \overline{1,M}$ (Figure 1).

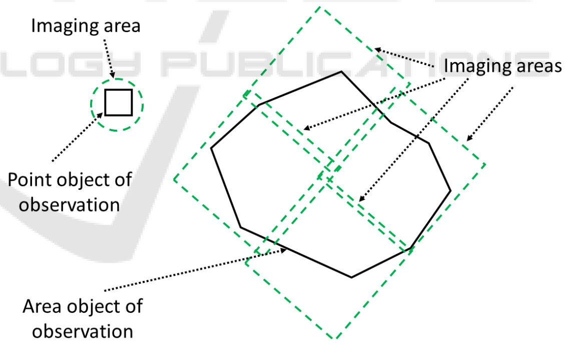

Figure 1: Imaging areas for point and area OO.

In the considered SS model, the satellite performs two operations:

- imaging a certain area sa_j $imaging_j$, characterized by the execution interval $t_j^{imag} = [t_j^{imagStart}; t_j^{imagEnd}]$ and the roll angle of the satellite $sAngle_j$;
- conducting a communication session of the satellite with the GS for transmitting the received data to the Earth $drop_j$, characterized by the execution interval $t_j^{drop} = [t_j^{dropStart}; t_j^{dropEnd}]$.

GS, in turn, performs one operation - receiving data from the satellite *receiv$_j$*, characterized by the execution interval $t_j^{receiv} = [t_j^{receivStart};\ t_j^{receivEnd}]$.

To implement ERS satellite imagery based on applications from customers, it is required to form a comprehensive plan for performing operations for a given planning horizon, drawn up in accordance with the criterion of minimizing the time for delivering images, as well as maximizing their quality. Thus, the objective function (OF) of the system is:

$$OF = \frac{1}{M}\sum_{k=1}^{N} OF_k \rightarrow max, \qquad (1)$$

$$OF_k = c_k F_1^k + (1 - c_k) F_2^k \rightarrow max, \qquad (2)$$

$$F_1^k = \frac{t_k^{end} - t_k^{dropEnd}}{t_k^{end} - t_k^{start}},$$

$$F_2^k = \begin{cases} \frac{minR_k - r_k}{minR_k - maxR_k}, & if\ r_k \geq maxR_k \\ \frac{r_k}{maxR_k}, & else \end{cases}$$

where OF – is the objective function of the system,
OF_k is the objective function of the k-th task,
N is the number of planned imaging sessions,
F_1^k – evaluation of the efficiency criterion of receiving data for the k-th task,
F_2^k – evaluation of the quality criterion of the resulting image for the k-th task,
t_k^{start}, t_k^{end} – the planning horizon for the k-th task,
res_k – the actual linear resolution of the resulting image for the k-th task.

In this case, a number of restrictions are imposed on the resulting solution:

1) fulfillment of the observability condition between the small satellite and the OO during imaging;
2) radio visibility between the small satellite and the GSwhen transmitting the imaging results;
3) availability of free space in the on-board memory of the satellite;
4) fulfillment of condition for prioritizing applications;
5) consistency of the sequence of times of operations;
6) satellite and GScan simultaneously perform no more than one operation.

In addition, it is necessary to carry out adaptive rebuilding of the locally optimal operating plan of the satellite with dynamic appearance of events that change the initial data for planning, such as changes in the composition and characteristics of the satellite elements (events of adding/removing the satellite and GS, changes in the available volume of satellite memory, adding/deleting work schedules and intervals of inaccessibility of satellite and GS, etc.), changes in the composition and parameters of applications for imaging of point and area objects.

3 OVERVIEW OF REFERENCES

As a solution to the problem of planning operations for orbital swarms, various heuristic algorithms are proposed, which have been previously tested on classical tasks of resource planning and allocation. Thus, the paper (Wang, 2016) describes application of the linear integer programming method for planning imaging of OO by a satellite group, taking into account possible cloudiness, modeled as stochastic events. The authors transform the random constraint programming model into a linear integer programming model using the sample approximation method. Then a search for solution is carried out using the developed branching and cutting algorithm based on generation of "lazy" calculations.

Another approach to solving this problem is given in (Iacopino, 2014). The authors suggest using an ant algorithm based on the model of behavior of ants looking for the shortest path from the colony to the food source.

Application of a multi-agent approach to planning the work of a swarm of ERS satellites is considered in (Bonnet, 2015). Advantages of self-adaptation and self-organization are given as prerequisites for application of the multi-agent approach for solving this problem, in relation to multi-criteria problems of large dimensions, requiring dynamic adaptation of the plan in case of abnormal events.

The work (Xiaolu, 2017) is devoted to solution of the subproblem of planning the operation of a satellite with several degrees of freedom using the method of local search in an expandable neighborhood. Planning of satellite communication sessions based on the methods of simulating annealing and search for options with restrictions is described in (Karapetyan, 2015). The use of a genetic algorithm for planning imaging of area objects by an ERS satellite swarm is considered in (Niu, 2018).

Besides, autonomous planning on board the satellite, which is the subject of papers (Gorodetsky, 2017) and (Lenzen, 2014), can also be singled out as a promising area of research. However, these studies are mostly theoretical in their nature. For their practical implementation, it is necessary to solve a number of fundamental problems.

The review has shown that the currently available methods of scheduling the work of orbital swarms are mainly of a centralized, hierarchical and monolithic nature, which greatly complicates flexible adaptation of the schedule in the rapidly changing conditions of the target environment. Moreover, these methods are based not on traditional mathematical optimization methods, the use of which leads to an avalanche increase in the volume of calculations, but on various kinds of heuristics for reducing exhaustive search. Meanwhile, other methods and algorithms are beginning to appear, which take into account domain semantics, analyzing conflicts, non-deterministic behavior, self-organization, adaptation, and working in real time. However, there are yet no descriptions of integral solutions to the problem of increasing efficiency of managing large-scale orbital swarms of small spacecrafts, suitable for practical digital implementation.

4 DATA MODEL

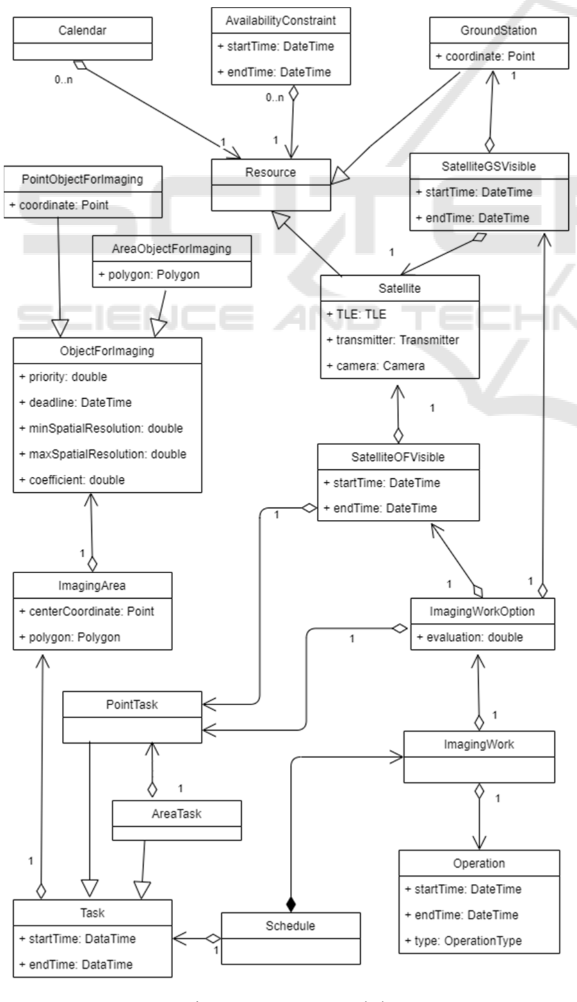

Figure 2: Data model.

The developed method uses a data model, the structure of which is shown in Figure 2 in the form of a class diagram in UML notation.

The *Resource* class describes some abstract resource, and its time of use must be scheduled. The spacecraft (*Satellite*) and GS (*GroundStation*) act as resources. For a resource, limitations can be set in the form of schedules (*Calendar*) and intervals of unavailability (*AvailabilityConstraint*).

The *ObjectForImaging* class is an abstract observation object that needs to be captured. The point OO corresponds to the *PointObjectForImaging* class, and the area OO corresponds to the *AreaObjectForImaging*. An imaging area (*ImagingArea*) is created for each OO.

The *Task* class is a task that needs to be scheduled. A task for imaging an area object (*AreaTask*) is a set of tasks for imaging point objects (*PointTask*).

For each task for imaging a point object, placement options (*ImagingWorkOption*) are formed, which are a combination of the satellite-observation object visibility (*SatelliteOOVisible*) and the satellite-ground station visibility (*SatelliteGSVisible*), as well as the OF value (2). During the planning process, one of the placement options is selected for the task, on the basis of which the *ImagingWork* is created. The imaging job consists of three operations (*Operation*): imaging, transmitting the data (drop) and receiving the data. A *Schedule* is used to store all planned imaging jobs.

5 ADAPTIVE PLANNING METHOD

Figure 3 shows the state diagram of the adaptive planning method, which includes the following main stages:

1) generation of tasks for imaging of OO;
2) calculation of options for possible placement;
3) conflict-free planning;
4) proactive planning.

5.1 Generation of Tasks for OO Imaging

At the first stage, tasks are generated for OO imaging based on the received applications. At the same time, depending on the type of OO, one or more imaging areas are created. Thus, an application for imaging a point OO is associated with one imaging area containing this object, and for an application for imaging an area object, it is divided into a set of adjacent areas, each corresponding to a point OO.

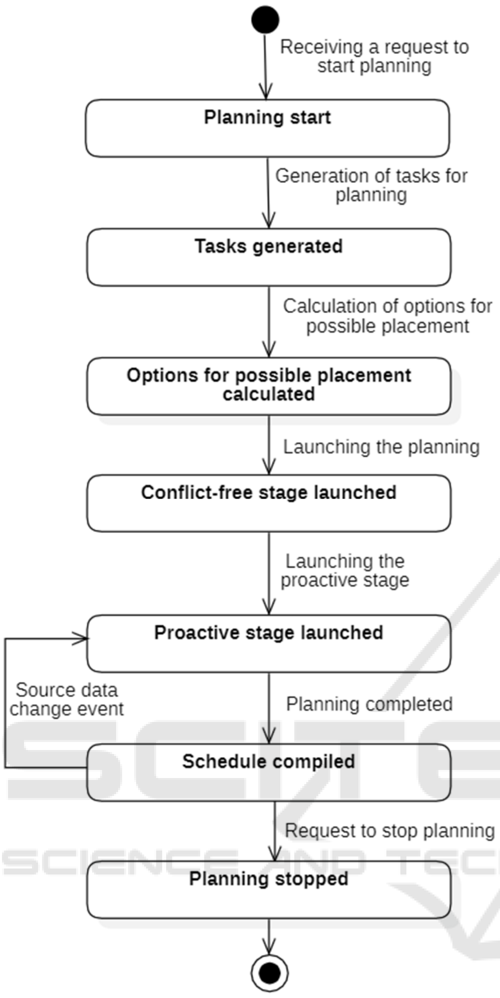
Receiving a request to start planning
Planning start
Generation of tasks for planning
Tasks generated
Calculation of options for possible placement
Options for possible placement calculated
Launching the planning
Conflict-free stage launched
Launching the proactive stage
Proactive stage launched
Source data change event
Planning completed
Schedule compiled
Request to stop planning
Planning stopped

Figure 3: State diagram of the adaptive planning method.

For each task, the deadlines are set in which it must be completed. The start time is the beginning of the planning horizon, and the end time is selected as the smallest of the two values: the end of the planning horizon or the point in time until which it is necessary to receive images, if it is specified in the application.

5.2 Calculation of Possible Placement Options

At the next stage, for each task, calculation of possible placement options is carried out, implemented on the basis of the method of successive concessions between criteria for efficiency and quality of received data, set by the coefficient c_p. The efficiency criterion has been chosen as the main one. Based on formula (2), the endpoint (boundary time) for transmitting the imaging results x is calculated by formula (3).

$$x = \frac{1}{c_p}\left(t_p^{end} - t_p^{start}\right) \times \times \left(1 - \frac{minR_p - r_p}{minR_p - maxR_p}\right) \times \times (1 - c_p) + t_p^{dropEnd}, \tag{3}$$

where x is the endpoint for transmitting the imaging results,

r_p is the current linear image resolution,

$t_p^{dropEnd}$ is the current drop time.

The pseudocode for the algorithm is shown in Algorithm 1.

Algorithm 1: Calculation of options for possible placement of the task.

```
Input: task_j, sa_j, SGSV – set of visibilities between the
satellite and GS, SOSV_j – set of visibilities between the
satellite and sa_j
Output: IWO_j – set of placement options for task_j
1:   sort(SGSV, ‘startTime’, ‘asc’)
2:   sort(SOSV_j, ‘startTime’, ‘asc’)
3:   x = task_j.endTime
4:   IWO_j = []
5:   bestEvaluation = 0
6:   do
7:   sgsv_k = SGSV.next()
8:     if sgsv_k.startTime <= x
9:       do
10:          sosv_p = SOSV_j.next()
11:          if sosv_p.endTime <= sgsv_k.startTime
12:             iwo = createIWO(task_j, sgsv_k, sosvp)
13:             IWO_j add iwo
14:       while SOSV_j.hasNext() and sosv_p.endTime
          <= sgsv_k.startTime
15:       sort(IWO_j, ‘evaluation’, ‘desc’)
16:       firstIWO = IWO_j[0]
17:       if firstIWO.evaluation > bestEvaluation
18:          bestEvaluation = firstIWO.evaluation
19:          x = calcBoundary(task_j, firstIWO)
20:   while SGSV.hasNext() and
      sgsv_k.startTime <= x
21:   return IWO_j
```

Let us take a closer look at the operation of this algorithm. The input of the algorithm is the following: $task_j$, for which it is necessary to calculate the options for possible placement, the imaging area sa_j corresponding to this task, as well as the pre-calculated satellite-OO visibility $SOSV_j$ and satellite-

GS visibility *SGSV*. At the beginning of the algorithm, the visibilities $SOSV_j$ and *SGSV* are sorted in ascending order of the start time of the corresponding interval (lines 1-2). The time limit for resetting the imaging results x is taken equal to the end time of scheduling the task $task_j.endTime$ (line 3). Next, sequential enumeration of the satellite-GS visibility *SGSV* is performed until the start time of the next visibility period $sgsv_k.startTime$ exceeds the current boundary time x (lines 6-20). At each iteration, the satellite-GS visibility *SGSV* and the previous satellite-OO visibility $SOSV_j$ are combined, and a placement option *iwo* is formed based on the pair $sgsv_k$ and $sosv_p$ (lines 9-14). At the end of each iteration, the system checks that the evaluation of the newly found variants does not exceed the current *bestEvaluation* (line 17). If so, the boundary time is recalculated to reset the imaging results x, and *bestEvaluation* takes on the new best value (lines 18-19).

In the course of the algorithm, a sequence of possible placement options for the problem $IWO_j = \{iwo_l\}, l = \overline{1, S}$ is built, at the beginning of which there is a placement option located at the global optimum OF_k of the problem's OF (2).

5.3 Conflict-free Planning

At the stage of conflict-free planning, an initial feasible schedule is constructed using a greedy optimization algorithm. The solution obtained at this stage will show the main bottlenecks of the considered schedule and will become a reference point for further improvements.

The pseudocode for the algorithm is shown in Algorithm 2. At the beginning, the list of tasks is ordered and grouped by the value of the pr_p priority (lines 1-2), thereby enforcing the constraint that a lower priority task cannot interfere with placement of a higher priority one. Then, sequentially for each group of tasks, an attempt for placement is made (lines 4-11), during which the tasks are placed on the first available option from the set IWO_j, where there are no conflicts with other tasks. Meanwhile, a set of planned jobs $IW = \{iw_k\}, k = \overline{1, K}$ is formed.

5.4 Proactive Planning

At the stage of proactive planning, using a multi-agent algorithm, the schedule obtained in the previous step is optimized by resolving conflicts between tasks that arise during placement.

Algorithm 2: Conflict-free planning algorithm.

```
Input: tasks, IWO is set of possible placement options
for the problem
Output: IW is set of planned jobs
1:   groupedTasks = group(tasks, 'priority')
2:   sort(groupedTasks, 'priority', 'desc')
3:   IW = []
4:   for taskGroup in groupedTasks
5:      parallel for $task_j$ in taskGroup
6:         $IWO_j$ = IWO[$task_j$]
7:         for $iwo_k$ in $IWO_j$
8:            conflicts = findConflicts($iwo_k$)
9:            if conflicts.empty
10:              iw = createImagingWork($iwo_k$, $task_j$)
11:              IW.add(iw)
12:  return IW
```

In the developed method, there are two types of agents: a task agent, the purpose of which is to occupy the most advantageous option in the schedule, and a scene agent, designed to control the activity of task agents and interact with external systems. The task agent is responsible for performing permutations in the schedule and has the satisfaction function *SF* (4) (Rzevski, 2020), determining the evaluation of the current satisfaction of its requirements:

$$SF_k(iw_k) = 1 - (OF_k(\dot{\iota wo}_k) - - OF_k(iw_k)), \quad (4)$$

where SF_k is the agent's satisfaction function,
$\dot{\iota wo}_k$ is the placement option located at the global optimum point of the task's OF,
iw_k is the current job on OO imaging.

Before starting the planning process, a smart agent is created for each task. Planning is controlled by the scene agent, which acts in accordance with Algorithm 3.

This algorithm works as follows. The launch of agents for proactivity is carried out iteratively, and before the start of each iteration, the system checks for new events changing the initial data (line 2). If there are such events, the planning context is updated by applying the events to the current initial data (line 3). Then the list of all *taskAgents* received as input is placed in a queue which is sorted in ascending order by the value of the agent satisfaction function (lines 4-5). Thus, at the very beginning of the queue there are those agents that are either the most unsatisfied with their position in the schedule, or not scheduled at all. Task agents are sequentially retrieved from the queue (line 9), and if the agent is not completely satisfied with its current position in the schedule (the value of its satisfaction function *SF* (4) is less than 1) (line 10), then a signal about the beginning of proactivity is sent to it (line 11). After receiving a message with results of proactivity, if the proactivity ended successfully, evaluations of all tasks affected by this proactivity are recalculated and the order of tasks in the queue is updated (lines 12-14). The condition for completing proactive planning is the absence of permutations of task agents at the next planning iteration, which means reaching an equilibrium point during negotiations and the possibility of issuing a ready-made solution (line 15).

Algorithm 3: Proactive planning algorithm.

```
Input: taskAgents, N – maximum number of
simultaneously active task agents
Output: Optimized schedule
1:   do
2:     if events of changes in the source data
3:       update planning context
4:     tasksQueue = new Queue(taskAgents)
5:     sort(tasksQueue, ‘elevation, ‘asc’)
6:     while !tasksQueue.isEmpty
7:       if number of active agents >= N
8:         waiting for the end of the proactivity of one
           of the agents
9:       taskAgent = tasksQueue.poll()
10:      if taskAgent.satisfaction < 1
11:        sceneAgent.sendMessage(taskAgent,
           “Start proactivity”) .then(proactiveResult
           =>
12:          if proactiveResult.isSuccessful
13:            updateEvaluation(
               proactiveResult.changedTasks)
14:            sort(tasksQueue, ‘elevation’,
               ‘asc’))
15:  while there were relocations in the schedule
```

Upon receiving a signal about the beginning of proactivity, the task agent attempts to find a more advantageous placement option for it according to Algorithm 4. To do this, it sequentially searches through possible placement options which are better than the current one (lines 2-16). At each search iteration, the agent first calculates the maximum possible compensation for displacement compensation, which it can provide to task agents conflicting for placement (line 4). This compensation is calculated according to the formula (5). It then searches for placement conflicts (line 6) and, if any, attempts to resolve them using the computed compensation (lines 7-12). In this case, each agent of the conflicting task is sequentially sent a message with a request to find other allocation intervals (line 10). The latter, in turn, upon receipt of this message, makes an attempt to find a new placement option using the compensation provided by Algorithm 4. If the attempt to resolve the conflict is successful and the agent of the conflicting task is ready to move, the compensation required by this agent is deducted from the total *compensation*, the conflict is marked as resolved and is removed from the general list of conflicts (lines 11-13). Otherwise, it proceeds to the next possible placement. After all conflicts are resolved, based on this placement option, an imaging job is created and added to the schedule instead of the previous one (lines 14-16).

$$\Delta SF = SF_j(\widetilde{iw}_j) - SF_j(iw_j) \qquad (5)$$

where ΔSF is the increment of the agent's satisfaction function,
iw_j is the current job on OO imaging,
$\widetilde{iw}_j$ is the new job on OO imaging.

Algorithm 4: Proactivity of the task agent.

```
Input: taskAgent$_j$, IWO$_j$, iw$_j$
Output: $\widetilde{iw}_j$ the new task for imaging task$_j$
1:   compensation = 1
2:   while IWO$_j$.hasNext() and
     compensation > 0
3:     iwo$_k$ = IWO$_j$.next()
4:     compensation = iwo$_k$.evaluation -
       iw$_j$.evaluation
5:     if compensation > 0
6:       conflicts = findConflicts(iwo$_k$)
7:       while conflicts.hasNext() and
         compensation > 0
8:         conflict = conflicts.next()
9:         conflictingTaskAgent = conflict
           .conflictingTaskAgent
10:        response = taskAgent$_j$ .sendMessage(
           conflictingTaskAgent, “Find other
           allocation intervals”, compensation)
11:        if response.message == “Found”
12:          compensation -=
             response.compensation
13:          conflicts.remove(conf$_p$)
14:      if conflicts.empty
15:        $\widetilde{sw}_j$ = createImagingWork(iwo$_k$,
           task$_j$)
16:        addToSchedule($\widetilde{iw}_j$ )
17:    return $\widetilde{iw}_j$
```

When new events changing the initial planning data are received, the proactive phase is launched again and dynamic adaptation of the schedule is performed in accordance with the changes that have occurred.

6 EXPERIMENTAL STUDIES

To conduct experimental studies in order to assess suitability of the proposed method for solving problems of managing swarms of satellites in real time, the SS model is used, which includes a group of 30 identical satellites, and a network of 10 GS.

Experiments were carried out on a PC with an Intel Core i7-3770 CPU (4 cores / 8 threads, 3.4GHz) and 8GB RAM, running under Windows 10.

6.1 Analysis of the Planning Process and Its Results

In this study, statistical information was obtained in order to assess the quality of the schedule and analyze the process of its construction.

The graphs in Figure 4 show the history of changes of values of the current and limiting objective function of the system since the start of planning, on the basis of which it is possible to estimate the difference between the current OF value (1) and its maximum possible value (6).

$$limOF = \frac{1}{M}\sum_{k=1}^{N} OF_k(\iota\dot{w}o_k), \qquad (6)$$

where *limOF* is the maximum possible OF of the system,
M is the total number of tasks,
N is the number of considered tasks,
$\iota\dot{w}o_k$ is the placement option located at the global optimum point of the OF of the *k*-th problem.

Figure 5 shows a diagram of distribution of the number of task placement options, which are better than their current placement. The diagram analysis shows that about 1200 tasks are scheduled at the most optimal option for them, and more than half of the remaining tasks are located in the 30% of the best options, which indicates a good quality of the resulting schedule.

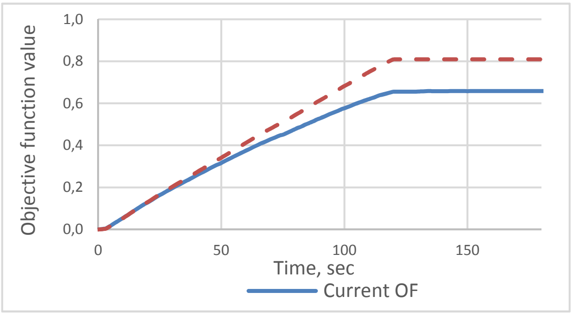

Figure 4: Graphs of changes in the value of the current and limiting OF during planning.

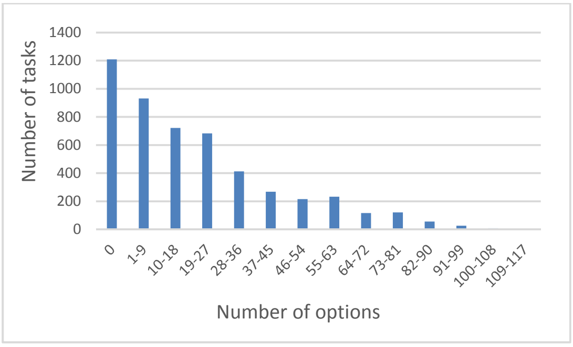

Figure 5: Distribution of the number of task placement options, which are better than their current placement.

The diagram of distribution of the number of task permutations at each iteration of proactive planning (Figure 6) demonstrates its fast convergence – the number of permutations already at the second iteration of planning is 7 times less than the number of permutations at the first iteration.

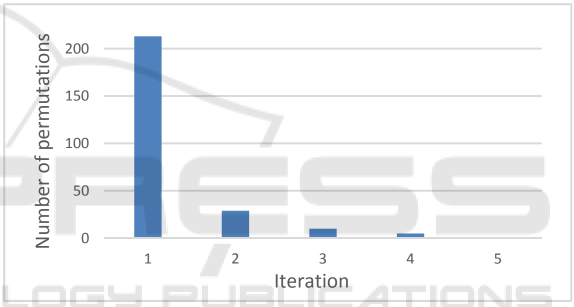

Figure 6: Number of task permutations at each iteration of proactive planning.

6.2 Studying the Method's Capability of Adapting the Schedule

In this study, capability of the method to adapt the schedule damaged by failure of one of the satellites has been evaluated. The time spent on rescheduling and the quality of the resulting schedule are the studied parameters. A series of 10 experiments was carried out, during which it was initially planned to execute 3000 applications for OO imaging, generated randomly according to a uniform distribution law. After all applications were successfully placed in the schedule, one of the satellites was excluded from the system, and the time spent on rebuilding the schedule, changes in the value of the system OF (1) and changes in the number of planned applications were measured. Results of the experiment are presented in Table 1.

Table 1: Results of experiments to study the system's ability to adapt the schedule.

№	Rescheduling time, s	After failure of a satellite		After reconstruction of schedule	
		Number of planned applications	ΔOF	Number of rescheduled applications	ΔOF
1	9	422	-0,11	418	0,07
2	8	379	-0,10	371	0,04
3	10	468	-0,11	465	0,08
4	11	411	-0,10	406	0,07
5	9	407	-0,11	396	0,06
6	10	425	-0,11	422	0,09
7	7	397	-0,11	395	0,06
8	8	388	-0,10	376	0,05
9	8	377	-0,07	372	0,05
10	9	419	-0,10	417	0,06

Thus, failure of one of the satellites led to a sharp drop in the system's OF by an average of 0.1 and the need to search for new placement options for 409 applications. During rescheduling to other satellites, 403 applications were rescheduled, which is 98% of the number of applications planned for the removed satellite. As a result of schedule reconstruction, OF increased to 0.69, which is less than the initial value by only 0.04. The average rescheduling time was about 9 seconds. Thus, the use of a multi-agent approach in planning makes it possible to quickly parry external events leading to a change in conditions of the problem being solved.

6.3 Efficiency Analysis versus Planning Algorithms based on Traditional Optimization Methods

In this study, efficiency of the developed method has been analyzed in comparison with planning algorithms based on traditional optimization methods, such as the simulated annealing algorithm, the Late Acceptance Hill Climbing algorithm and the Tabu Search algorithm. These were compared in terms of the quality of the resulting schedule and the time required for its compilation.

Within this series of experiments, the number of applications for OO imaging varied from 100 to 20,000. The time spent on compiling the plan and the system OF (1) were measured.

Based on results of these experiments, graphs of dependence of OF (Figure 7) and planning time (Figure 8) on the number of applications for various planning algorithms were built. For the simulated annealing algorithm and the Tabu Search algorithm, results were received only up to 5000 applications for imaging, because after that, an exponential increase in the time of work and consumed resources was observed.

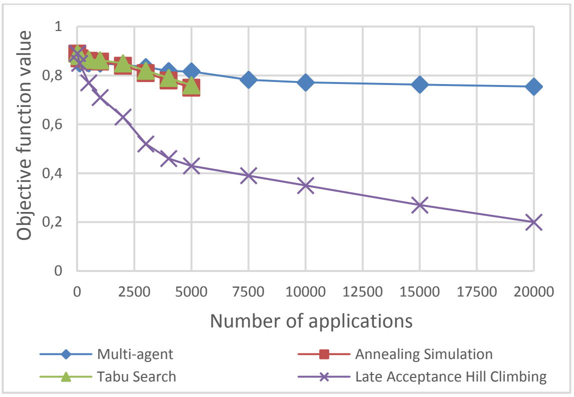

Figure 7: Graph of dependence of OF on the number of applications.

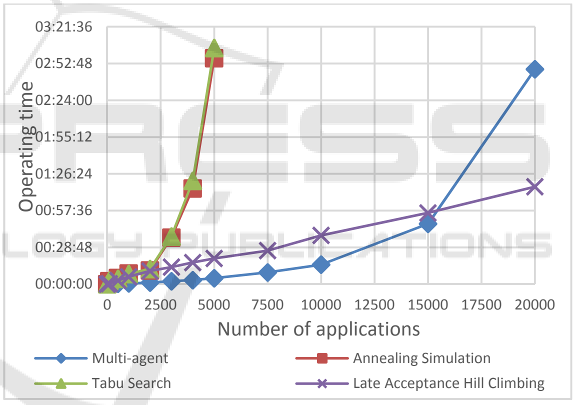

Figure 8: Graph of dependence of planning time on the number of applications.

Results of these experiments show that the proposed multi-agent method is not inferior to traditional heuristic algorithms for low-dimensionality problems, and with an increase in the number of planned applications, it demonstrates a higher speed of scheduling without losing the quality.

7 CONCLUSIONS

The authors of the paper propose a method for solving the problem of adaptive planning of operations for a large-scale orbital swarm of remote sensing of the Earth small satellites on the basis of a multi-agent approach.

Experimental studies have demonstrated high suitability of the method for increasing efficiency of using resources of the new generation satellites.

Further research will focus on improving the planning algorithms by introducing a virtual marketplace and adding deeper analysis of the current planning context to reduce enumeration of options. In addition, it is planned to introduce the space system ontology in order to provide a more flexible and adaptive ability to customize the applied rules. All these actions will ultimately make it possible to create a real management system with the ability to service a large number of small satellites and applications.

ACKNOWLEDGEMENTS

The paper has been prepared based on materials of scientific research within the subsidized state theme of the Samara Federal Research Scientific Center RAS, Institute for Control of Complex Systems RAS for research and development on the topic: № AAAA-A19-119030190053-2 “Research and development of methods and means of analytical design, computer-based knowledge representation, computational algorithms and multi-agent technology in problems of optimizing management processes in complex systems”.

REFERENCES

Shimoda, H., 2016. Remote Sensing Data Applications. In *Handbook of Satellite Applications*, P. 1-70.

Kopacz, J., Herschitz, R., Roney, J., 2020. Small Satellites an Overview and Assessment. *Acta Astronautica.* Vol. 170, P. 93-105

Karsaev, O., 2016. A Review of Conventional and Innovative Satellite Mission Planning Systems. *Tr. SPIIRAN*. Issue 48, P. 151-181. (In Russian)

Rzevski, G., Skobelev, P., 2014. *Managing complexity*. WIT Press. Boston.

Gorodetsky, V., Skobelev, P., 2020. System engineering view on multi-agent technology for industrial applications: barriers and prospects. *Cybernetics and Physics*. Vol. 9, No. 1, P. 13-30.

Wang, J., Demeulemeester, E., Qiu, D., 2016. A pure proactive scheduling algorithm for multiple earth observation satellites under uncertainties of clouds. *Computers & Operations Research*. Vol. 74. P. 1-13

Iacopino, C., Palmer, P., Policella, N., Donati, A., Brewer, A., 2014. How ants can manage your satellites. *Acta Futura*. No. 9, P. 59-70.

Bonnet J., Gleizes M., Kaddoum E., Rainjonneau S., Flandin G., 2015. Multi-satellite Mission Planning Using a Self-Adaptive Multi-agent System. In *Proceedings of the 2015 IEEE 9th International Conference on Self-Adaptive and Self-Organizing Systems (SASO ’15)*. P. 11-20

Xiaolu L., Laporte G., Chen Y., He R., 2017. An adaptive large neighborhood search metaheuristic for agile satellite scheduling with time-dependent transition time. *Computers & Operations Research.* P. 41-53

Karapetyan D., Minic S., Malladi K.T., Punnen A., 2015. Satellite downlink scheduling problem: A case study. *Omega.* P. 115-123

Niu X., Tang H., Wu L., 2018. Satellite Scheduling of Large Areal Tasks for Rapid Response to Natural Disaster Using a Multi-Objective Genetic Algorithm. *International Journal of Disaster Risk Reduction*. Vol. 28, P. 813-825

Gorodetsky V., Karsaev O., 2017. Self-organization of group behavior of a cluster of small satellites of a distributed observation system. *Izvestiya Yuzhnogo federalnogo universiteta. Tekhnicheskie nauki*. No. 2, P. 234-247 (In Russian)

Lenzen, C., 2014. Onboard Planning and Scheduling Autonomy within the Scope of the Fire Bird Mission. *Proceedings of the 14-th International Conference on Space Operations*. P. 517-527

Rzevski, G., Skobelev, P., Zhilyaev, A., Lakhin, O., Mayorov, I., Simonova, E., 2018. Ontology-Driven Multi-Agent Engine for Real Time Adaptive Scheduling. *International Conference on Control, Artificial Intelligence, Robotics & Optimization (ICCAIRO)*, P. 14-22