Collaboration Among a Satellite Swarm

Grégory Bonnet*
ONERA - DCSD / CNES / Alcatel Space Alenia
2 avenue Edouard Belin BP 4025
31055 Toulouse, France
gregory.bonnet@onera.fr

Catherine Tessier
ONERA - DCSD
2 avenue Edouard Belin BP 4025
31055 Toulouse, France
catherine.tessier@onera.fr

ABSTRACT

The paper deals with on-board planning for a satellite swarm via communication and negotiation. We aim at defining individual behaviours that result in a global behaviour that meets the mission requirements. We will present the formalization of the problem, a communication protocol, a solving method based on reactive decision rules, and first results.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Plan execution, formation, and generation*; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Coherence and coordination*

General Terms

Algorithms

Keywords

Multiagent systems, Cooperative distributed problem solving, Task and resource allocation, Coordination, Cooperation and teamwork

1. INTRODUCTION

Much research has been undertaken to increase satellite autonomy such as enabling them to solve by themselves problems that may occur during a mission, adapting their behaviour to new events and transferring planning on-board ; even if the development cost of such a satellite is increased, there is an increase in performance and mission possibilities [34]. Moreover, the use of satellite swarms - sets of satellites flying in formation or in constellation around the Earth - makes it possible to consider joint activities, to distribute skills and to ensure robustness.

*PhD student.

AAMAS'07 May 14–18 2007, Honolulu, Hawai'i, USA.

Multi-agent architectures have been developed for satellite swarms [36, 38, 42] but strong assumptions on deliberation and communication capabilities are made in order to build a collective plan.

Mono-agent planning [4, 18, 28] and task allocation [20] are widely studied. In a multi-agent context, agents that build a collective plan must be able to change their goals, reallocate resources and react to environment changes and to the others' choices. A coordination step must be added to the planning step [40, 30, 11]. However, this step needs high communication and computation capabilities. For instance, coalition-based [37], contract-based [35] and all negotiation-based [25] mechanisms need these capabilities, especially in dynamic environments.

In order to relax communication constraints, coordination based on norms and conventions [16] or strategies [17] are considered. Norms constraint agents in their decisions in such a way that the possibilities of conflicts are reduced. Strategies are private decision rules that allow an agent to draw benefit from the knowledgeable world without communication. However, communication is still needed in order to share information and build collective conjectures and plans.

Communication can be achieved through a stigmergic approach (via the environment) or through message exchange and a protocol. A protocol defines interactions between agents and cannot be uncoupled from its goal, e.g. exchanging information, finding a trade-off, allocating tasks and so on. Protocols can be viewed as an abstraction of an interaction [9]. They may be represented in a variety of ways, e.g. AUML [32] or Petri-nets [23]. As protocols are originally designed for a single goal, some works aim at endowing them with flexibility [8, 26]. However, an agent cannot always communicate with another agent or the communication possibilites are restricted to short time intervals.

The objective of this work is to use intersatellite connections, called InterSatellite Links or ISL, in an Earth observation constellation inspired from the Fuego mission [13, 19], in order to increase the system reactivity and to improve the mission global return through a hybrid agent approach. At the individual level, agents are deliberative in order to create a local plan but at the collective level, they use normative decision rules in order to coordinate with one another. We will present the features of our problem, a communication protocol, a method for request allocation and finally, collaboration strategies.

978-81-904262-7-5 (RPS)

2. PROBLEM FEATURES

An observation satellite constellation is a set of satellites in various orbits whose mission is to take pictures of various areas on the Earth surface, for example hot points corresponding to volcanos or forest fires. The ground sends the constellation observation requests characterized by their geographical positions, priorities specifying if the requests are urgent or not, the desired dates of observation and the desired dates for data downloading.

The satellites are equipped with a single observation instrument whose mirror can roll to shift the line of sight. A minimum duration is necessary to move the mirror, so requests that are too close together cannot be realized by the same satellite. The satellites are also equipped with a detection instrument pointed forward that detects hot points and generates observation requests on-board.

The constellations that we consider are such as the orbits of the various satellites meet around the poles. A judicious positioning of the satellites in their orbits makes it possible to consider that two (or more) satellites meet in the polar areas, and thus can communicate without the ground intervention. Intuitively, intersatellite communication increases the reactivity of the constellation since each satellite is within direct view of a ground station (and thus can communicate with it) only 10 % of the time.

The features of the problem are the following:
— 3 to 20 satellites in the constellation;
— pair communication around the poles;
— no ground intervention during the planning process;
— asynchronous requests with various priorities.

3. A MULTI-AGENT APPROACH

As each satellite is a single entity that is a piece of the global swarm, a multi-agent system fits to model satellite constellations [39]. This approach has been developped through the ObjectAgent architecture [38], TeamAgent [31], DIPS [14] or Prospecting ANTS [12].

3.1 Satellite swarm

An observation satellite swarm[1] is a multi-agent system where the requests do not have to be carried out in a fixed order and the agents (the satellites) do not have any physical interaction. Carrying out a request cannot prevent another agent from carrying out another one, even the same one. At most, there will be a waste of resources. Formally, a swarm is defined as follows:

DEFINITION 1 (SWARM). *A* satellite swarm $\mathcal{E}$ *is a triplet* $< \mathcal{S}, \mathbb{T}, Vicinity >$*:*
— $\mathcal{S}$ *is a set of* n *agents* $\{s_1 \dots s_n\}$*;*
— $\mathbb{T} \subseteq \mathbb{R}^+$ *or* $\mathbb{N}^+$ *is a set of dates with a total order* $<$*;*
— $Vicinity : \mathcal{S} \times \mathbb{T} \mapsto 2^{\mathcal{S}}$.

In the sequel, we will assume that the agents share a common clock.

For a given agent and a given time, the vicinity relation returns the set of agents with whom it can communicate at that time. As we have seen previously, this relation exists when the agents meet.

[1]This term will designate a satellite constellation with InterSatellite Links.

3.2 Requests

Requests are the observation tasks that the satellite swarm must achieve. As we have seen previously, the requests are generated both on the ground and on board. Each agent is allocated a set of initial requests. During the mission, new requests are sent to the agents by the ground or agents can generate new requests by themselves. Formally, a request is defined as follows:

DEFINITION 2 (REQUEST). *A* request R *is defined as a tuple* $< id_R, pos(R), prio(R), tbeg(R), b_R >$*:*
— id_R *is an identifier;*
— $pos(R)$ *is the geographic position of* R*;*
— $prio(R) \in \mathbb{R}$ *is the request priority;*
— $tbeg(R) \in \mathbb{T}$ *is the desired date of observation;*
— $b_R \in \{true, false\}$ *specifies if* R *has been realized.*

The priority $prio(R)$ of a request represents how much it is important for the user, namely the request sender, that the request should be carried out. Thus a request with a high priority must be realized at all costs. In our application, priorities are comprised between 1 and 5 (the highest).

In the sequel, we will note $\mathcal{R}^t_{s_i}$ the set of the requests that are known by agent s_i at time $t \in \mathbb{T}$.

For each request R in $\mathcal{R}^t_{s_i}$, there is a cost value, noted $\text{cost}_{s_i}(R) \in \mathbb{R}$, representing how far from the desired date of observation $tbeg(R)$ an agent s_i can realize R. So, the more an agent can carry out a request in the vicinity of the desired date of observation, the lower the cost value.

3.3 Candidacy

An agent may have several intentions about a request, i.e. for a request R, an agent s_i may:
— propose to carry out R : s_i may realize R;
— commit to carry out R : s_i will realize R;
— not propose to carry out R : s_i may not realize R;
— refuse to carry out R : s_i will not realize R.

We can notice that these four propositions are modalities of proposition C: s_i **realizes** R:
— $\Diamond C$ means that s_i proposes to carry out R;
— $\Box C$ means that s_i commits to carry out R;
— $\neg\Diamond C$ means that s_i does not propose to carry out R;
— $\neg\Box C$ means that s_i refuses to carry out R.
More formally:

DEFINITION 3 (CANDIDACY). *A* candidacy C *is a tuple* $< id_C, mod_C, s_C, R_C, obs_C, dnl_C >$*:*
— id_C *is an identifier;*
— $mod_C \in \{\Diamond, \Box, \neg\Diamond, \neg\Box\}$ *is a modality;*
— $s_C \in \mathcal{S}$ *is the candidate agent;*
— $R_C \in \mathcal{R}^t_{s_C}$ *is the request on which* s_C *candidates;*
— $obs_C \in \mathbb{T}$ *is the realization date proposed by* s_C*;*
— $dnl_C \in \mathbb{T}$ *is the download date.*

3.4 Problem formalization

Then, our problem is the following: we would like each agent to build request allocations (i.e a plan) dynamically such as if these requests are carried out their number is the highest possible or the global cost is minimal. More formally,

DEFINITION 4 (PROBLEM). *Let* $\mathcal{E}$ *be a swarm. Agents* s_i *in* $\mathcal{E}$ *must build a set* $\{A^t_{s_1} \dots A^t_{s_n}\}$ *where* $A^t_{s_i} \subseteq \mathcal{R}^t_{s_i}$ *such*

as:

— $|\bigcup_{s_i \in \mathcal{S}} A^t_{s_i}|$ *is maximal;*
— $\sum_{s_i \in \mathcal{S}} \sum_{R \in A^t_{s_i}} prio(R)$ *is maximal.*
— $\sum_{s_i \in \mathcal{S}} \sum_{R \in A^t_{s_i}} cost_{s_i}(R)$ *is minimal.*

Let us notice that these criteria are not necessarily compatible.

As the choices of an agent will be influenced by the choices of the others, it is necessary that the agents should reason on a common knowledge about the requests. It is thus necessary to set up an effective communication protocol.

4. COMMUNICATION PROTOCOL

Communication is commonly associated with cooperation. Deliberative agents need communication to cooperate, whereas it is not necessarily the case for reactive agents [2, 41].

Gossip protocols [22, 24], or epidemic protocols, are used to share knowledge with multicast. Each agent selects a set of agents at a given time in order to share information. The speed of information transmission is contingent upon the length of the discussion round.

4.1 The corridor metaphor

The suggested protocol is inspired from what we name the *corridor metaphor*, which represents well the satellite swarm problem. Various agents go to and fro in a corridor where objects to collect appear from time to time. Two objects that are too close to each other cannot be collected by the same agent because the action takes some time and an agent cannot stop its movement. In order to optimize the collection, the agents can communicate when they meet.

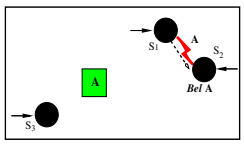

Figure 1: Time t

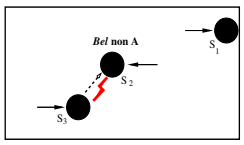

Figure 2: Time t'

Example 1. *Let us suppose three agents, s_1, s_2, s_3 and an object A to be collected. At time t, s_1 did not collect A and s_2 does not know that A exists. When s_1 meets s_2, it communicates the list of the objects it knows, that is to say A. s_2 now believes that A exists and prepares to collect it. It is not certain that A is still there because another agent may have passed before s_2, but it can take it into account in its plan.*
At time t', s_3 collects A. In the vicinity of s_2, s_3 communicates its list of objects and A is not in the list. As both agents meet in a place where it is possible for s_3 to have collected A, the object would have been in the list if it had not been collected. s_2 can thus believe that A does not exist anymore and can withdraw it from its plan.

4.2 Knowledge to communicate

In order to build up their plans, agents need to know the current requests and the others agents' intentions. For each agent two kinds of knowledge to maintain are defined:

— requests (Definition 2);
— candidacies (Definition 3).

Definition 5 (Knowledge). *Knowledge K is a tuple $< data(K), S_K, t_K >$:*

— *$data(K)$ is a request R or a candidacy C;*
— *$S_K \subseteq \mathcal{S}$ is the set of agents knowing K;*
— *$t_K \in \mathbb{T}$ is a temporal timestamp.*

In the sequel, we will note $\mathcal{K}^t_{s_i}$ the knowledge of agent s_i at time $t \in \mathbb{T}$.

4.3 An epidemic protocol

From the corridor metaphor, we can define a communication protocol that benefits from all the communication opportunities. An agent notifies any change within its knowledge and each agent must propagate these changes to its vicinity who update their knowledge bases and reiterate the process. This protocol is a variant of epidemic protocols [22] inspired from the work on overhearing [27].

Protocol 1 (Communication). *Let s_i be an agent in $\mathcal{S}$. $\forall t \in \mathbb{T}$:*

— *$\forall\ s_j \in Vicinity(s_i, t)$, s_i executes:*
  1. *$\forall\ K \in \mathcal{K}^t_{s_i}$ such as $s_j \notin S_K$:*
     a. *s_i communicates K to s_j*
     b. *if s_j acknowledges receipt of K, $S_K \leftarrow S_K \cup \{s_j\}$.*
— *$\forall\ K \in \mathcal{K}^t_{s_i}$ received by s_j at time t:*
  1. *s_j updates $\mathcal{K}^t_{s_j}$ with K*
  2. *s_j acknowledges receipt of K to s_i.*

Two kinds of updates exist for an agent:

— an *internal* update from a knowledge modification by the agent itself;
— an *external* update from received knowledge.

For an internal update, updating K depends on $data(K)$: a candidacy C is modified when its modality changes and a request R is modified when an agent realizes it. When K is updated, the timestamp is updated too.

Protocol 2 (Internal update). *Let $s_i \in \mathcal{S}$ be an agent. An* internal update *from s_i at time $t \in \mathbb{T}$ is performed:*

— *when knowledge K is created;*
— *when $data(K)$ is modified.*

In both cases:

1. *$t_K \leftarrow t$;*
2. *$S_K \leftarrow \{s_i\}$.*

For an external update, only the most recent knowledge K is taken into account because timestamps change only when $data(K)$ is modified. If K is already known, it is updated if the content or the set of agents knowing it have been modified. If K is unknown, it is simply added to the agent's knowledge.

Protocol 3 (External update). *Let s_i be an agent and $\mathcal{K}$ the knowledge transmitted by agent s_j. $\forall\ K \in \mathcal{K}$, the* external update *at time $t \in \mathbb{T}$ is defined as follows:*

1. *if $\exists\ K' \in \mathcal{K}^t_{s_i}$ such as $id_{data(K)} = id_{data(K')}$ then*
   a. *if $t_K \geq t_{K'}$ then*
      i. *if $t_K > t_{K'}$ then $S_K \leftarrow S_K \cup \{s_i\}$*
      ii. *if $t_K = t_{K'}$ then $S_K \leftarrow S_K \cup S_{K'}$*
      iii. *$\mathcal{K}^t_{s_i} \leftarrow (\mathcal{K}^t_{s_i} \backslash \{K'\}) \cup \{K\}$*
2. *else*
   a. *$\mathcal{K}^t_{s_i} \leftarrow \mathcal{K}^t_{s_i} \cup \{K\}$*
   b. *$S_K \leftarrow S_K \cup \{s_i\}$*

If the incoming information has a more recent timestamp, it means that the receiver agent has obsolete information. Consequently, it replaces the old information by the new one and adds itself to the set of agents knowing K (*1.a.i*).

If both timestamps are the same, both pieces of information are the same. Only the set of the agents knowing K may have changed because agents s_i and s_j may have already transmitted the information to other agents. Consequently, the sets of agents knowing K are unified (*1.a.ii*).

4.4 Properties

Communication between two agents when they meet is made of the conjunction of Protocol 1 and Protocol 3. In the sequel, we call this conjunction a *communication occurrence*.

4.4.1 Convergence

The structure of the transmitted information and the internal update mechanism (Protocol 2) allow the process to converge. Indeed, a request R can only be in two states (realized or not) given by the boolean b_R. Once an internal update is made - i.e. R is realized - R cannot go back to its former state. Consequently, an internal update can only be performed once.

As far as candidacies are concerned, updates only modify the modalities, which may change many times and go back to previous states. Then it seems that livelocks[2] would be likely to appear. However, a candidacy C is associated to a request and a realization date (the deadline given by obs_C). After the deadline, the candidacy becomes meaningless. Thus for each candidacy, there exists a date $t \in \mathbb{T}$ when changes will propagate no more.

4.4.2 Complexity

It has been shown that in a set of N agents where a single one has a new piece of information, an epidemic protocol takes $\mathcal{O}(logN)$ steps to broadcast the information [33]. During one step, each agent has a communication occurrence. As agents do not have much time to communicate, such a communication occurrence must not have a too big temporal complexity, which we can prove formally:

PROPOSITION 1. *The* temporal complexity *of a communication occurrence at time $t \in \mathbb{T}$ between two agents s_i and s_j is, for agent s_i,*

$$\mathcal{O}(|\mathcal{R}^t_{s_i}|.|\mathcal{R}^t_{s_j}|.|\mathcal{S}|^2)$$

PROOF 1. *For the worst case, each agent s_k sends $|\mathcal{R}^t_{s_k}|$ pieces of information on requests and $|\mathcal{R}^t_{s_k}|.|\mathcal{S}|$ pieces of informations on candidacies (one candidacy for each request and for each agent of the swarm). Let s_i and s_j two agents meeting at time $t \in \mathbb{T}$. For agent s_i, the complexity of Protocol 1 is*

$$\mathcal{O}(\underbrace{|\mathcal{R}^t_{s_i}| + |\mathcal{R}^t_{s_i}|.|\mathcal{S}|}_{emission} + \underbrace{|\mathcal{R}^t_{s_j}| + |\mathcal{R}^t_{s_j}|.|\mathcal{S}|}_{reception})$$

For each received piece of information, agent s_i uses Protocol 3 and searches through its knowledge bases: $|\mathcal{R}^t_{s_i}|$ pieces of information for each received request and $|\mathcal{R}^t_{s_i}|.|\mathcal{S}|$ pieces of information for each received candidacy. Consequently, the complexity of Protocol 3 is

$$\mathcal{O}(|\mathcal{R}^t_{s_j}|.|\mathcal{R}^t_{s_i}| + |\mathcal{R}^t_{s_j}|.|\mathcal{R}^t_{s_i}|.|\mathcal{S}|^2)$$

Thus, the temporal complexity of a communication occurrence is:

$$\mathcal{O}(|\mathcal{R}^t_{s_i}| + |\mathcal{R}^t_{s_i}|.|\mathcal{S}| + |\mathcal{R}^t_{s_j}|.|\mathcal{R}^t_{s_i}| + |\mathcal{R}^t_{s_j}|.|\mathcal{R}^t_{s_i}|.|\mathcal{S}|^2))$$

Then:

$$\mathcal{O}(|\mathcal{R}^t_{s_i}|.|\mathcal{R}^t_{s_j}|.|\mathcal{S}|^2)$$

[2]Communicating endlessly without converging.

5. ON-BOARD PLANNING

In space contexts, [5, 21, 6] present multi-agent architectures for on-board planning. However, they assume high communication and computation capabilities [10]. [13] relax these constraints by cleaving planning modules: on the first hand, satellites have a planner that builds plans on a large horizon and on the second hand, they have a decision module that enables them to choose to realize or not a planned observation.

In an uncertain environment such as the one of satellite swarms, it may be advantageous to delay the decision until the last moment (i.e. the realization date), especially if there are several possibilities for a given request. The main idea in contingency planning [15, 29] is to determine the nodes in the initial plan where the risks of failures are most important and to incrementally build contingency branches for these situations.

5.1 A deliberative approach

Inspired from both approaches, we propose to build allocations made up of a set of unquestionable requests and a set of uncertain disjunctive requests on which a decision will be made at the end of the decision horizon. This horizon corresponds to the request realization date. Proposing such partial allocations allows conflicts to be solved locally without propagating them through the whole plan.

In order to build the agents' initial plans, let us assume that each agent is equipped with an on-board planner. A plan is defined as follows:

DEFINITION 6 (PLAN). *Let s_i be an agent, $\mathcal{R}^t_{s_i}$ a set of requests and $\mathcal{C}^t_{s_i}$ a set of candidacies. Let us define three sets:*

— the set of potential requests*:*
$\mathcal{R}^p = \{R \in \mathcal{R}^t_{s_i} | b_R = false\}$

— the set of mandatory requests*:*
$\mathcal{R}^m = \{R \in \mathcal{R}^p | \exists C \in \mathcal{C}^t_{s_i} : mod_C = \Box, s_C = s_i, R_C = R\}$

— the set of given-up requests*:*
$\mathcal{R}^g = \{R \in \mathcal{R}^p | \exists C \in \mathcal{C}^t_{s_i} : mod_C = \neg\Box, s_C = s_i, R_C = R\}$

A plan $A^t_{s_i}$ generated at time $t \in \mathbb{T}$ is a set of requests such as $\mathcal{R}^m \subseteq A^t_{s_i} \subseteq \mathcal{R}^p$ and $\nexists R \in \mathcal{R}^g$ such as $R \in A^t_{s_i}$.

Building a plan generates candidacies.

DEFINITION 7 (GENERATING CANDIDACIES). *Let s_i be an agent and $A^{t_1}_{s_i}$ a (possibly empty) plan at time t_1. Let $A^{t_2}_{s_i}$ be the plan generated at time t_2 with $t_2 > t_1$.*

— $\forall R \in A^{t_1}_{s_i}$ such as $R \notin A^{t_2}_{s_i}$, a candidacy C such as $mod(C) = \neg\Diamond$, $s_C = s_i$ and $R_C = R$ is generated;

— $\forall R \in A^{t_2}_{s_i}$ such as $R \notin A^{t_1}_{s_i}$, a candidacy C such as $mod(C) = \Diamond$, $s_C = s_i$ and $R_C = R$ is generated;

— Protocol 2 is used to update $\mathcal{K}^{t_1}_{s_i}$ in $\mathcal{K}^{t_2}_{s_i}$.

5.2 Conflicts

When two agents compare their respective plans some conflicts may appear. It is a matter of redundancies between allocations on a given request, i.e.: several agents stand as candidates to carry out this request. Whereas such redundancies may sometimes be useful to ensure the realization of a request (the realization may fail, e.g. because of clouds), it may also lead to a loss of opportunity. Consequently, conflict has to be defined:

DEFINITION 8 (CONFLICT). *Let s_i and s_j be two agents with, at time t, candidacies C_{s_i} and C_{s_j} respectively ($s_{C_{s_i}} = s_i$ and $s_{C_{s_j}} = s_j$). s_i and s_j are in* conflict *if and only if:*
— $R_{C_{s_i}} = R_{C_{s_j}}$
— $mod_{C_{s_i}}$ *and* $mod_{C_{s_j}} \in \{\Box, \Diamond\}$

Let us notice that the agents have the means to know whether they are in conflict with another one during the communication process. Indeed, they exchange information not only concerning their own plan but also concerning what they know about the other agents' plans.

All the conflicts do not have the same strength, meaning that they can be solved with more or less difficulty according to the agents' communication capacities. A conflict is *soft* when the concerned agents can communicate before one or the other carries out the request in question. A conflict is *hard* when the agents cannot communicate before the realization of the request.

DEFINITION 9 (SOFT/HARD CONFLICT). *Let s_i and s_j ($i < j$) two agents in conflict with, at time t, candidacies C_{s_i} and C_{s_j} respectively ($s_{C_{s_i}} = s_i$ and $s_{C_{s_j}} = s_j$). If $\exists\ V \subseteq \mathcal{S}$ such as $V = \{s_i \dots s_j\}$ and if $\exists\ T \in \mathbb{T}$ such as $T = \{t_{i-1} \dots t_{j-1}\}$ ($t_{i-1} = t$) where: $\forall\ i \leq k < j$, $s_{k+1} \in Vicinity(s_k, t_k)$ with $t_k < obs_{C_{s_i}}$, $t_k < obs_{C_{s_j}}$ and $t_k \geq t_{k-1}$ then the conflict is* soft *else it is* hard.

A conflict is soft if it exists a chain of agents between the two agents in conflict such as information can propagate before both agents realize the request. If this chain does not exist, it means that the agents in conflict cannot communicate directly or not. Consequently, the conflict is hard.

In satellite swarms, the geographical positions of the requests are known as well as the satellite orbits. So each agent is able to determine if a conflict is soft or hard.
We can define the conflict cardinality:

DEFINITION 10 (CONFLICT CARDINALITY). *Let s_i be an agent and R a request in conflict. The* conflict cardinality *is $card_c(R) = |\{C \in \mathcal{C}^t_{s_i} | mod_C \in \{\Box, \Diamond\}, C_R = R\}|$.*

The conflict cardinality corresponds to the number of agents that are candidates or committed to the same request. Thus, a conflict has at least a cardinality of 2.

6. COLLABORATION STRATEGIES

In space contexts, communication time and agents' computing capacities are limited. When they are in conflict, the agents must find a local agreement (instead of an expensive global agreement) by using the conflict in order to increase the number of realized requests, to decrease the time of mission return, to increase the quality of the pictures taken or to make sure that a request is carried out.

EXAMPLE 2. *Let us suppose a conflict on request R between agents s_i and s_j. We would like that the most expert agent, i.e. the agent that can carry out the request under the best conditions, does it. Let us suppose s_i is the expert. s_i must allocate R to itself. It remains to determine what s_j must do: s_j can either select a substitute for R in order to increase the number of requests potentially realized, or do nothing in order to preserve resources, or allocate R to itself to ensure redundancy.*

Consequently, we can define collaboration strategies dedicated to conflict solving. A strategy is a private (namely intrinsic to an agent) decision process that allows an agent to make a decision on a given object. In our application, strategies specify what to do with redundancies.

6.1 Cost and expertise

In our application, cost is linked to the realization dates. Carrying out a request consumes the agents' resources (e.g.: on-board energy, memory). Consequently, an observation has a cost for each agent which depends on when it is realized: the closer the realization date to the desired date of observation, the lower the cost.

DEFINITION 11 (COST). *Let s_i be an agent. The cost $cost_{s_i}(R_C) \in \mathbb{R}$ to carry out a request R_C according to a candidacy C is defined as: $cost_{s_i}(R_C) = |obs_C - tbeg(R_C)|$.*

From this cost notion, we can formally define an expert notion between two agents. The expertise for an agent means it can realize the request at the lower cost.

DEFINITION 12 (EXPERTISE). *Let s_i and $s_j \in \mathcal{S}$ be two agents and R a request. Agent s_i is an* expert *for R if and only if $cost_{s_i}(R) \leq cost_{s_j}(R)$.*

6.2 Soft conflict solving strategies

Three strategies are proposed to solve a conflict. The *expert* strategy means that the expert agent maintains its candidacy whereas the other one gives up. The *altruist* strategy means that the agent that can download first[3], provided the cost increase is negligible, maintains its candidacy whereas the other one gives up. The *insurance* strategy means that both agents maintain their candidacies in order to ensure redundancy.

STRATEGY 1 (EXPERT). *Let s_i and s_j be two agents in conflict on their respective candidacies C_{s_i} and C_{s_j} such as s_i is the expert agent. The* expert strategy *is: $mod_{C_{s_i}} = \Box$ and $mod_{C_{s_j}} = \neg\Box$.*

STRATEGY 2 (ALTRUIST). *Let s_i and s_j be two agents in conflict on their respective candidacies C_{s_i} and C_{s_j} such as s_i is the expert agent. Let $\epsilon \in \mathbb{R}^+$ be a threshold on the cost increase. The* altruist strategy *is : if $dnl_{C_{s_i}} > dnl_{C_{s_j}}$ and $|cost_{s_i}(R) - cost_{s_j}(R)| < \epsilon$ then $mod_{C_{s_i}} = \neg\Box$ and $mod_{C_{s_j}} = \Box$.*

STRATEGY 3 (INSURANCE). *Let s_i and s_j be two agents in conflict on their respective candidacies C_{s_i} and C_{s_j} such as s_i is the expert agent. Let $\alpha \in \mathbb{R}$ be a priority threshold. The* insurance strategy *is : if $\frac{prio(R)}{card_c(R)-1} > \alpha$ then $mod_{C_{s_i}} = \Diamond$ and $mod_{C_{s_j}} = \Diamond$.*

[3]i.e. the agent using memory resources during a shorter time.

In the insurance strategy, redundancy triggering is adjusted by the conflict cardinality $card_c(R)$. The reason is the following: the more redundancies on a given request, the less a new redundancy on this request is needed.

The three strategies are implemented in a negotiation protocol dedicated to soft conflicts. The protocol is based on a subsumption architecture [7] on strategies: the insurance strategy (*1*) is the major strategy because it ensures redundancy for which the swarm is implemented. Then the altruist strategy comes (*2*) in order to allocate the resources so as to enhance the mission return. Finally, the expert strategy that does not have preconditions (*3*) enhances the cost of the plan.

PROTOCOL 4 (SOFT CONFLICT SOLVING). *Let R be a request in a soft conflict between two agents, s_i and s_j. These agents have C_{s_i} and C_{s_j} for respective candidacies. Let s_i be the* expert *agent. Agents apply strategies as follows:*

1. *insurance strategy (α)*
2. *altruist strategy (ϵ)*
3. *expert strategy*

The choice of parameters α and ϵ allows to adjust the protocol results. For example, if $\epsilon = 0$, the altruist strategy is never used.

6.3 Hard conflict solving strategies

In case of a hard conflict, the agent that is not aware will necessarily realize the request (with success or not). Consequently, a redundancy is useful only if the other agent is more expert or if the priority of the request is high enough to need redundancy. Therefore, we will use the insurance strategy (refer to Section 6.2) and define a *competitive strategy.*

The latter is defined for two agents, s_i and s_j, in a hard conflict on a request R. Let s_i be the agent that is aware of the conflict[4].

STRATEGY 4 (COMPETITIVE). *Let $\lambda \in \mathbb{R}^+$ be an cost threshold. The* competitive strategy *is: if $cost_{s_i}(R) < cost_{s_j}(R) - \lambda$ then $mod_{C_{s_i}} = \diamond$.*

PROTOCOL 5 (HARD CONFLICT SOLVING). *Let s_i be an agent in a hard conflict with an agent s_j on a request R. s_i applies strategies as follows:*

1. *insurance strategy (α)*
2. *competitive strategy (λ)*
3. *withdrawal : $mod_{C_{s_i}} = \neg\Box$*

6.4 Generalization

Although agents use pair communication, they may have information about several agents and conflict cardinality may be more than 2. Therefore, we define a k-conflict as a conflict with a cardinality of k on a set of agents proposing or committing to realize the same request. Formally,

DEFINITION 13 (k-CONFLICT). *Let $S = \{s_1 \dots s_k\}$ be a set of agents with respective candidacies $C_{s_1} \dots C_{s_k}$ at time t. The set S is in a k-*conflict *if and only if:*

— $\forall 1 \leq i \leq k$, $s_{C_{s_i}} = s_i$;

— $!\exists R$ *such as* $\forall 1 \leq i \leq k$, $R_{C_{s_i}} = R$;

— $\forall 1 \leq i \leq k$, $mod_{C_{s_i}} \in \{\Box, \diamond\}$.

— *S is maximal ($\subseteq$) among the sets that satisfy these properties.*

As previously, a k-conflict can be soft or hard. A k-conflict is soft if each pair conflict in the k-conflict is a soft conflict with respect to Definition 9.

As conflicts bear on sets of agents, expertise is a total order on agents. We define rank-i-expertise where the concerned agent is the ith expert.

In case of a soft k-conflict, the rank-i-expert agent makes its decision with respect to the rank-$(i+1)$-expert agent according to Protocol 4. The protocol is applied recursively and α and ϵ parameters are updated at each step in order to avoid cost explosion[5].

In case of a hard conflict, the set S of agents in conflict can be splitted in S^S (the subset of agents in a soft conflict) and S^H (the subset of unaware agents). Only agents in S^S can take a decision and must adapt themselves to agents in S^H. The rank-i-expert agent in S^S uses Protocol 5 on the whole set S^H and the rank-$(i-1)$-expert agent in S^S. If an agent in S^S applies the competitive strategy all the others withdraws.

7. EXPERIMENTS

Satellite swarm simulations have been implemented in JAVA with the JADE platform [3]. The on-board planner is implemented with linear programming using ILOG CPLEX [1]. The simulation scenario implements 3 satellites on 6-hour orbits. Two scenarios have been considered: the first one with a set of 40 requests with low mutual exclusion and conflict rate and the second one with a set of 74 requests with high mutual exclusion and conflict rate.

For each scenario, six simulations have been performed: one with centralized planning (all requests are planned by the ground station before the simulation), one where agents are isolated (they cannot communicate nor coordinate with one another), one informed simulation (agents only communicate requests) and three other simulations implementing the instanciated collaboration strategies (politics):

— *neutral politics*: α, ϵ and λ are set to average values;

— *drastic politics*: α and λ are set to higher values, i.e. agents will ensure redundancy only if the priorities are high and, in case of a hard conflict, if the cost payoff is much higher;

— *lax politics*: α is set to a lower value, i.e. redundancies are more frequent.

In the case of low mutual exclusion and conflict rate (Table 1), centralized and isolated simulations lead to the same number of observations, with the same average priorities. Isolation leading to a lower cost is due to the high number of redundancies: many agents carry out the same request at different costs. The informed simulation reduces the number of redundancies but sligthly increases the average cost for the same reason. We can notice that the use of

[4]i.e. the agent that must make a decision on R.

[5]For instance, the rank-1-expert agent withdraws due to the altruist strategy and the cost increases by ϵ in the worst case, then rank-2-expert agent withdraws due to the altruist strategy and the cost increases by ϵ in the worst case. So the cost has increased by 2ϵ in the worst case.

Simulation	Observations	Redundancies	Messages	Average priority	Average cost
Centralized	34	0	0	2.76	176.06
Isolated	34	21	0	2.76	160.88
Informed	34	6	457	2.65	165.21
Neutral politics	31	4	1056	2.71	191.16
Drastic politics	24	1	1025	2.71	177.42
Lax politics	33	5	1092	2.7	172.88

Table 1: Scenario 1 - the 40-request simulation results

Simulation	Observations	Redundancies	Messages	Average priority	Average cost
Centralized	59	0	0	2.95	162.88
Isolated	37	37	0	3.05	141.62
Informed	55	27	836	2.93	160.56
Neutral politics	48	25	1926	3.13	149.75
Drastic politics	43	21	1908	3.19	139.7
Lax politics	53	28	1960	3	154.02

Table 2: Scenario 2 - the 74-request simulation results

collaboration strategies allows the number of redundancies to be much more reduced but the number of observations decreases owing to the constraint created by commitments. Furthermore, the average cost is increased too. Nevertheless each avoided redundancy corresponds to saved resources to realize on-board generated requests during the simulation.

In the case of high mutual exclusion and conflict rate (Table 2), noteworthy differences exist between the centralized and isolated simulations. We can notice that all informed simulations (with or without strategies) allow to perform more observations than isolated agents do with less redundancies. Likewise, we can notice that all politics reduce the average cost contrary to the first scenario. The drastic politics is interesting because not only does it allow to perform more observations than isolated agents do but it allows to highly reduce the average cost with the lowest number of redundancies.

As far as the number of exchanged messages is concerned, there are 12 meetings between 2 agents during the simulations. In the worst case, at each meeting each agent sends N pieces of information on the requests plus $3N$ pieces of information on the agents' intentions plus 1 message for the end of communication, where N is the total number of requests. Consequently, 3864 messages are exchanged in the worst case for the 40-request simulations and 7128 messages for the 74-request simulations. These numbers are much higher than the number of messages that are actually exchanged. We can notice that the informed simulations, that communicate only requests, allow a higher reduction.

In the general case, using communication and strategies allows to reduce redundancies and saves resources but increases the average cost: if a request is realized, agents that know it do not plan it even if its cost can be reduce afterwards. It is not the case with isolated agents. Using strategies on little constrained problems such as scenario 1 constrains the agents too much and causes an additional cost increase. Strategies are more useful on highly constrained problems such as scenario 2. Although agents constrain themselves on the number of observations, the average cost is widely reduce.

8. CONCLUSION AND FUTURE WORK

An observation satellite swarm is a cooperative multi-agent system with strong constraints in terms of communication and computation capabilities. In order to increase the global mission outcome, we propose an hybrid approach: deliberative for individual planning and reactive for collaboration.

Agents reason both on requests to carry out and on the other agents' intentions (candidacies). An epidemic communication protocol uses all communication opportunities to update this information. Reactive decision rules (strategies) are proposed to solve conflicts that may arise between agents. Through the tuning of the strategies (α, ϵ and λ) and their plastic interlacing within the protocol, it is possible to coordinate agents without additional communication: the number of exchanged messages remains nearly the same between informed simulations and simulations implementing strategies.

Some simulations have been made to experimentally validate these protocols and the first results are promising but raise many questions. What is the trade-off between the constraint rate of the problem and the need of strategies? To what extent are the number of redundancies and the average cost affected by the tuning of the strategies?

Future works will focus on new strategies to solve new conflicts, specially those arising when relaxing the independence assumption between the requests. A second point is to take into account the complexity of the initial planning problem. Indeed, the chosen planning approach results in a combinatory explosion with big sets of requests: an anytime or a fully reactive approach has to be considered for more complex problems.

Acknowledgements

We would like to thank Marie-Claire Charmeau (CNES[6]), Serge Rainjonneau and Pierre Dago (Alcatel Space Alenia) for their relevant comments on this work.

[6]The French Space Agency

9. REFERENCES

[1] ILOG inc. CPLEX. http://www.ilog.com/products/cplex.

[2] T. Balch and R. Arkin. Communication in reactive multiagent robotic systems. *Autonomous Robots*, pages 27–52, 1994.

[3] F. Bellifemine, A. Poggi, and G. Rimassa. JADE - a FIPA-compliant agent framework. In *Proceedings of PAAM'99*, pages 97–108, 1999.

[4] A. Blum and M. Furst. Fast planning through planning graph analysis. *Artificial Intelligence*, Vol. 90:281–300, 1997.

[5] E. Bornschlegl, C. Guettier, G. L. Lann, and J.-C. Poncet. Constraint-based layered planning and distributed control for an autonomous spacecraft formation flying. In *Proceedings of the 1st ESA Workshop on Space Autonomy*, 2001.

[6] E. Bornschlegl, C. Guettier, and J.-C. Poncet. Automatic planning for autonomous spacecraft constellation. In *Proceedings of the 2nd NASA Intl. Workshop on Planning and Scheduling for Space*, 2000.

[7] R. Brooks. A robust layered control system for a mobile robot. *MIT AI Lab Memo*, Vol. 864, 1985.

[8] A. Chopra and M. Singh. Nonmonotonic commitment machines. *Lecture Notes in Computer Science: Advances in Agent Communication*, Vol. 2922:183–200, 2004.

[9] A. Chopra and M. Singh. Contextualizing commitment protocols. In *Proceedings of the 5th AAMAS*, 2006.

[10] B. Clement and A. Barrett. Continual coordination through shared activites. In *Proceedings of the 2nd AAMAS*, pages 57–64, 2003.

[11] J. Cox and E. Durfee. Efficient mechanisms for multiagent plan merging. In *Proceedings of the 3rd AAMAS*, 2004.

[12] S. Curtis, M. Rilee, P. Clark, and G. Marr. Use of swarm intelligence in spacecraft constellations for the resource exploration of the asteroid belt. In *Proceedings of the Third International Workshop on Satellite Constellations and Formation Flying*, pages 24–26, 2003.

[13] S. Damiani, G. Verfaillie, and M.-C. Charmeau. An Earth watching satellite constellation : How to manage a team of watching agents with limited communications. In *Proceedings of the 4th AAMAS*, pages 455–462, 2005.

[14] S. Das, P. Gonzales, R. Krikorian, and W. Truszkowski. Multi-agent planning and scheduling environment for enhanced spacecraft autonomy. In *Proceedings of the 5th ISAIRAS*, 1999.

[15] R. Dearden, N. Meuleau, S. Ramakrishnan, D. Smith, and R. Wahington. Incremental contingency planning. In *Proceedings of ICAPS'03 Workshop on Planning under Uncertainty and Incomplete Information*, pages 1–10, 2003.

[16] F. Dignum. Autonomous agents with norms. *Artificial Intelligence and Law*, Vol. 7:69–79, 1999.

[17] E. Durfee. Scaling up agent coordination strategies. *IEEE Computer*, Vol. 34(7):39–46, 2001.

[18] K. Erol, J. Hendler, and D. Nau. HTN planning : Complexity and expressivity. In *Proceedings of the 12th AAAI*, pages 1123–1128, 1994.

[19] D. Escorial, I. F. Tourne, and F. J. Reina. Fuego : a dedicated constellation of small satellites to detect and monitor forest fires. *Acta Astronautica*, Vol.52(9-12):765–775, 2003.

[20] B. Gerkey and M. Matarić. A formal analysis and taxonomy of task allocation in multi-robot systems. *Journal of Robotics Research*, Vol. 23(9):939–954, 2004.

[21] C. Guettier and J.-C. Poncet. Multi-level planning for spacecraft autonomy. In *Proceedings of the 6th ISAIRAS*, pages 18–21, 2001.

[22] I. Gupta, A.-M. Kermarrec, and A. Ganesh. Efficient epidemic-style protocols for reliable and scalable multicast. In *Proceedings of the 21st IEEE Symposium on Reliable Distributed Systems*, pages 180–189, 2002.

[23] G. Gutnik and G. Kaminka. Representing conversations for scalable overhearing. *Journal of Artificial Intelligence Research*, Vol. 25:349–387, 2006.

[24] K. Jenkins, K. Hopkinson, and K. Birman. A gossip protocol for subgroup multicast. In *Proceedings of the 21st International Conference on Distributed Computing Systems Workshops*, pages 25–30, 2001.

[25] N. Jennings, S. Parsons, P. Norriega, and C. Sierra. On augumentation-based negotiation. In *Proceedings of the International Workshop on Multi-Agent Systems*, pages 1–7, 1998.

[26] J.-L. Koning and M.-P. Huget. A semi-formal specification language dedicated to interaction protocols. *Information Modeling and Knowledge Bases XII: Frontiers in Artificial Intelligence and Applications*, pages 375–392, 2001.

[27] F. Legras and C. Tessier. LOTTO: group formation by overhearing in large teams. In *Proceedings of 2nd AAMAS*, 2003.

[28] D. McAllester, D. Rosenblitt, P. Norriega, and C. Sierra. Systematic nonlinear planning. In *Proceedings of the 9th AAAI*, pages 634–639, 1991.

[29] N. Meuleau and D. Smith. Optimal limited contingency planning. In *Proceedings of the 19th AAAI*, pages 417–426, 2003.

[30] P. Modi and M. Veloso. Bumping strategies for the multiagent agreement problem. In *Proceedings of the 4th AAMAS*, pages 390–396, 2005.

[31] J. B. Mueller, D. M. Surka, and B. Udrea. Agent-based control of multiple satellite formation flying. In *Proceedings of the 6th ISAIRAS*, 2001.

[32] J. Odell, H. Parunak, and B. Bauer. Extending UML for agents. In *Proceedings of the Agent-Oriented Information Systems Workshop at the 17th AAAI*, 2000.

[33] B. Pittel. On spreading a rumor. *SIAM Journal of Applied Mathematics*, Vol. 47:213–223, 1987.

[34] B. Polle. Autonomy requirement and technologies for future constellation. *Astrium Summary Report*, 2002.

[35] T. Sandholm. Contract types for satisficing task allocation. In *Proceedings of the AAAI Spring Symposium: Satisficing Models*, pages 23–25, 1998.

[36] T. Schetter, M. Campbell, and D. M. Surka. Multiple agent-based autonomy for satellite constellation. *Artificial Intelligence*, Vol. 145:147–180, 2003.

[37] O. Shehory and S. Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, Vol. 101(1-2):165–200, 1998.

[38] D. M. Surka. ObjectAgent for robust autonomous control. In *Proceedings of the AAAI Spring Symposium*, 2001.

[39] W. Truszkowski, D. Zoch, and D. Smith. Autonomy for constellations. In *Proceedings of the SpaceOps Conference*, 2000.

[40] R. VanDerKrogt and M. deWeerdt. Plan repair as an extension of planning. In *Proceedings of the 15th ICAPS*, pages 161–170, 2005.

[41] B. Werger. Cooperation without deliberation : A minimal behavior-based approach to multi-robot teams. *Artificial Intelligence*, Vol. 110:293–320, 1999.

[42] P. Zetocha. Satellite cluster command and control. *IEEE Aerospace Conference*, Vol. 7:49–54, 2000.