ABSTRACT

This paper proposes a distributed approach for planning and scheduling of the on-board activities of a space system flotilla that can achieve several mission goals through their element coordination. Each system is an autonomous agent with its own knowledge base and an own planning-scheduling reasoning capability. The Planning engine maintains a least commitment approach to look for the solution and the Scheduling phase exploits the temporal flexibility during the activity allocation process. Two different negotiation strategies are presented to coordinate the agents during the scheduling process. They emphasize either the team welfare or the individual welfare. A communication protocol to regulate the coordination is also described. Simulation results on a rover exploration scenario are reported.

1. INTRODUCTION

Space missions achieved by means of distributed systems made of autonomous agents that coordinate each others are currently catching an increasing attention; DARWIN, MER and LISA are some of the most famous missions based on multiple elements. The exploitation of an autonomous agent formation makes the overall mission more reliable and its management more flexible. Those benefits are enhanced as soon as a distributed approach rather than a centralized solution is applied to coordinate the agents. On the other hand, those space system formation elements, while having their own resource set, their specific goal to be accomplished, and their own system and functional constraints to answer those tasks, must coordinate each other to accomplish common mission goals. The coordination is the price to be paid for the highlighted benefits as it requires the definition of a negotiation strategy and an efficient communication protocol. A centralized architecture does not care of such a further degree of complexity, by working on the system as a whole with the drawback that if the central scheduler fails the flotilla dies.

This paper presents a distributed architecture solution to cope with the planning/scheduling tasks according to space system formation, both spacecraft and robots. The architecture here proposed has an Inference Knowledge module to formalize the activities in terms of preconditions and effects and to support the Planner. A Constraint Database module describes the activities in terms of resource demand and metric temporal relations to support the Scheduler. The Planning engine is based on a Partial Ordered Planning algorithm, a least commitment strategy is maintained to look for the solution of the pre-post constraint net. No arbitrary choice is made, to contain the backtracking mechanism. The Distributed Scheduling module preserves a temporal flexibility with no instantiation for the time variables, by working on their domain limitations to have the constraints net consistency preserved. As a consequence, the finally scheduled scenario is robust and any sort of uncertainty can be easily coped with. The negotiation is nested in the distributed scheduling process. During the negotiation process, negotiated sequencing constraints are added among activities and propagated in a Distributed Simple Temporal Network. The process keeps posting constraints up to the condition of no capacity demand violation occurrence is gained. This flexible time approach gives the benefit of better visiting the search space: as soon as the value of a time variable is selected, the search space loses a dimension. On the contrary, a constraint settlement just restricts the range of variables, without necessarily decreasing the search space dimension. The potential of leaving a great number of variable assignment possibilities is, then preserved, while the risk for the search process to get lost in blind alleys is reduced.

Two negotiation strategies are applied, to emphasize either the team welfare or the single agent welfare respectively.

The communication protocol is based on the innovation of the communication oriented graph concept. The communication channels are dynamically defined to lead the traffic risen from the information exchange among agents.

The proposed architecture is applied to a rover scenario devoted to planetary exploration. Three problems of different size are considered to verify the validity of the proposed approach.
A critical analysis on the proposed negotiation paradigms is also offered.

2. THE AGENT ARCHITECTURE

The agent architecture is made up of different operational layers each concerning a particular functionality: planning, scheduling, sensing, acting, communicating and preserving a knowledge about the problem. Fig. 1 shows the agent architecture; the arrows are representative for the connections among layers and the information flow direction [1].

Fig. 1. Agent architecture

The Inference Knowledge and the Constraint Database represent the overall knowledge base of the single agent.

The Inference Knowledge contains the logical description of the activities to plan. Such a description concerns with the specification of the necessary preconditions and effects for each activity. The initial and final state description (i.e. mission goal) are specified too.

The Constraint Database is the part of the knowledge concerning the scheduling process. Each activity is described in terms of resource demands and metric temporal relations.

The Planner builds a logical chain of activities by imposing causal links among them to reach the goal state from the initial state. This process aims to satisfy the logical consistency of the solution. The result of this process is an acyclic oriented graph, with nodes for activities and oriented arcs for ordering relations.

The Scheduler works both on the base of the agent plan and on the constraint information contained in the Constraint Database. The Scheduling reasoning is faced by a Simple Temporal Network (STN) solved with an All-Pairs-Shortest-Path algorithm. This strategy allows preserving a temporal flexibility with no instantiation for the time variables, by working on their bounds to gain the constraint network consistency. This approach assures several consistent solutions, useful to deal with environment and execution uncertainties.

The World Interface manages all income and outcome agent information, as results of communication and sensing operation.

Fig. 2 shows the global algorithm performed in parallel by each agent to plan and schedule mission goals.

Fig. 2. Two agents scenario: global algorithm. One color identifies one agent

The following sections explain in detail each phase of the algorithm in Fig. 2.

3. INDIVIDUAL PLANNING

The Planning engine is based on a Hierarchical-Decomposition-Partial-Ordered-Planning (HD-POP) algorithm [2]. A logical reasoning is applied, with a deliberative approach to deal with the decision making problem. HD-POP looks for the solution searching in the space of the plans. HD-POP starts with a partial plan to extend in each iteration till a complete plan is reached. The algorithm extends the partial plans by adding new causal links to satisfy a precondition c of a step S1. This is obtained by choosing a particular step S2 with effects that match the precondition c. The new causal link is registered among the S1 and S2 steps. If both no threat to the set of causal links and no inconsistency is found, the HD-POP goes on with a new iteration, backtracks otherwise. During each iteration, the algorithm selects and decomposes - according to a specific decomposition formula - a non-primitive step of the plan. The algorithm stops as soon as all preconditions of all steps are reached, and a complete plan is gained.

The HD-POP maintains a least commitment strategy in finding solution. No arbitrary choice is made, to contain the backtracking mechanism. During this process, the agents recognize preconditions they cannot satisfy since no activities to reach those preconditions exist in the agent knowledge base. The individual plan satisfies all preconditions that agent can reach with its own
activities. The remaining free preconditions will represent the driving information during the coordination process and, at the end of this phase, they will be reached thanks to the effects of an other agent activity.

4. THE PLAN COORDINATION

The planning coordination takes place as soon as all agents have created their own individual plan. A flotilla agent assumes a coordinator role. No different exists between the coordinator and the other agent architecture, hence any agent can assume such a role, making the overall framework more reliable.

The coordinator receives individual plans and merges them according to a global view of the whole formation. This global view is represented by a global plan. It is formed by the individual plans in parallel (n is the number of agents in the flotilla). In this plan some preconditions can be still not reached. This global plan represents a partial (global) plan to complete. The coordinator invokes the same HD-POP algorithm to complete the global plan by posting causal links among steps of different agent plans to satisfy the remaining unmatched preconditions. This process creates inter-agent ordering constraints. The activities constrained by inter-agent ordering relations are named shared or common activities.

Once the global plan is completed (i.e. all preconditions of all steps are reached), the coordinator extracts and returns to each agent the respective coordinated plan.

A coordinated plan contains more information than an individual plan, the new data are represented by inter-agent constraints and shared steps.

5. THE DISTRIBUTED SCHEDULING PHASE

A DCSP (Distributed Constraint Satisfaction Problem) is proposed to deal with the distributed scheduling phase [3,4]. Several architectures operate by binding values to variables and the solution can be reached by either incrementally extending a partial consistent assignment or repairing a total inconsistent instantiation.

The architecture here proposed adopts an alternative approach: the algorithm maintains a temporal flexibility with no instantiation for the variables that are managed as intervals. The interval bounds are updated to satisfy the consistency of the temporal constraint set thanks to Floyd-Warshall algorithm. The distributed scheduling operates simultaneously with the negotiation process. Instead of binding exact negotiated values to variables, negotiated sequencing constraints are added among activities in resource conflict. The process keeps imposing enough constraints to ensure that at any point in time, the requested capacity does not exceed the available capacity. This flexible time approach has the benefit to better discover the search space [5]. Every time the value of a variable is fixed, the search space loses one dimension. Posting a constraint only restricts the range of variables, without necessarily decreasing their domain dimension.

The coordinated plan is extended in each agent into a temporal flexible constraints net, introducing the problem temporal variables. A STP (Simple Temporal Problem) is, hence, defined [6, 7].

The coordination and inter-agent temporal consistency are satisfied by maintaining equal values of the shared variable domains in the agent memories. During the scheduling process, the single agent modifies its own constraint network to solve the resource conflicts and invokes the Floyd - Warshall algorithm to maintain its own temporal consistency by updating the variables domains. Any change occurred to the domains of the shared variables must be communicated to the other agents in order to keep coordination and inter-agent temporal consistency. Such an information exchange has to be regulated by a suitable communication protocol, explained in a dedicated section.

As shown in Fig. 2, the distributed scheduling is split in two sequential processes: a first phase to solve all conflicts concerning depletable resources and a second phase to solve the remaining resource conflicts (about not-depletable resources). Different strategies are adopted by the agents to solve the detected resource inconsistencies.

5.1 The depletable resource conflicts management

The system deals with renewable-depletable resources. The algorithm orders the activities by using resource R according to the expected execution order. The amount of the resource capacity needed by each activity is supposed to be spent at the starting point. From the first activity listed (i.e. the first activity expected to be executed), the algorithm iteratively evaluates the constraint 1.

\[
\text{cap}(R) - \sum_{i=k}^{n} \text{cap}(A_i, R) \geq 0
\]

\(\text{cap}(R)\) = initial total capacity
\(\text{cap}(A_i, R)\) = amount of capacity consumed (produced) by activity \(A_i\).

If the k-th iteration does not satisfy the constraint, a conflict is detected and all activities from \(A_1\) to \(A_k\) take part to the conflict. The strategy adopted by the agent to deal with the detected inconsistency stays in inserting a regenerative activity with a specific amount of \(R\) capacity. Being the k-th activity listed not legal (i.e. not feasible), the regenerative step is planned according to the ordering relations 2 and 3.
The system manages granular regenerating activities. The entity of granular recharging is previously defined by an external user. Iteratively, the agent inserts granular activities till the conflict is solved. A macro regenerative activity is hence obtained, sized for the detected conflict. The capacity produced is the sum of the single granular capacities.

The insertion of a new granular activity means a different constraints pattern. Therefore, at each iteration, intra-agent and inter-agent consistency must be verified thanks to the Floyd – Warshall algorithm and the correspondence of the shared variable domains must be verified.

5.2 The not-depletable resource conflicts management

The conflict identification algorithm is based on the contemporary graph of the already planned activities. Each node is weighted, according to the amount of capacity required by activity $A_i$: $cap(A_i, R)$. A Maximum Weighted Clique Problem is ran and the following resource constraint is checked for each single clique [8].

$$cap(R) \geq W_{\text{clique}} \cdot W_{\text{clique}} = \sum_i cap(A_i, R)$$

$W_{\text{clique}}$ = weight of the single clique.

The cliques that violate constraint (4) are labelled as conflicts and the correspondent activities (nodes) take part to the conflict solution process. The strategy here proposed to solve those inconsistencies is to post ordering constraints among activities in each not consistent clique. Fig. 3 describes the algorithm flow chart to solve (not-depletable) resource conflicts by introducing negotiated sequencing constraints. The distributed scheduling operates simultaneously with the negotiation process. Instead of binding exact negotiated values to variables, negotiated sequencing constraints are added among activities in the resource conflict. The process keeps posting enough constraints to ensure that at any point in time, the requested capacity does not exceed the available capacity. This flexible time approach has the benefit to better visit the search space.

The algorithm in fig. 11 is based on a Textured-Based heuristic [9, 10]. Two texture measurements are evaluated: Contention and Reliance. The resource $R$ with maximum Contention degree is selected. Two activities, in the same conflict, with maximum Reliance degree are proposed to be constrained by a sequencing relation. There are two ways to order activities $A$ and $B$: $A$ before $B$ and $B$ before $A$. These are two sequencing constraints that each agent proposes in a negotiation session. Each proposal gives rise to different effects of the propagation in each agent STN, and different resource conflict profiles can rise. The effects of the single sequencing constraint is due to the presence of the inter-agent constraints. The proposal effects turn out to be the shared variables domain modifications. The single negotiation session has no more than $2n$ proposals to be evaluated, with a $n$ agents scenario. A unique final agreement exists and all agents finally propagate its effects. The negotiation sessions take place till each agent has no conflicts. No cyclic loop can occur, since the solution search method focuses on incrementally posting constraints rather than an iterative repair strategy.

6. THE NEGOTIATION

Each agent proposes two new constraints to solve a local resource conflict. Each proposal gives new domains for the shared variables. Such information represents both the effects and the object of the proposal. The evaluation step consists of the propagation of the proposed shared variable domains in each agent STN. On the resulting scheduling profile, each agent computes a Fitness Function (FF). The FF consists of three hierarchical criteria: first of all the satisfaction of the temporal consistency, secondly the number of resource conflicts and finally a Resource Global Contention degree. The comparison between FF computed on the actual scheduling profile and the FF coming from the current proposal is the criterion to judge the proposal validity. Whenever the effects of the proposal are temporally consistent the vector in (5) can be defined.

\[
\begin{align*}
S_{\text{igen}} & \text{ before } S_k & (2) \\
S_{k-1} & \text{ before } S_{\text{igen}} & (3)
\end{align*}
\]
\[ \Delta FF = \begin{bmatrix} G_{CONF} & G_{CONT} \end{bmatrix} \] (5)

\(G_{CONF}\) is the Conflict Gain and represents the variation of activities in resource conflict. The greater it is the more the conflict are solved.

\(G_{CONT}\) is the Contention Gain and represents the variation of Resource Global Contention degree due to the proposal. Those proposals that guarantee the temporal consistency for all agents are negotiated and considered. The agent preference is given according to a maximisation strategy for the \(G_{CONF}\). Whenever two proposals assume the same values of \(G_{CONF}\), the agent preference is given to proposal that maximises \(G_{CONT}\). The agent associates to each proposal formulated a precise \(\Delta FF\). After computing all \(\Delta FF\), the agent can define a preference list on proposals. This list is generally different for every agent, hence a negotiation strategy has to be settled to find an agreement. Two different strategies have been here elaborated.

### 6.1 The concurrent strategy

The i-th proposal is described in 6, as an n-dimensional vector in the space of Conflict Gains:

\[(Proposal)_i = [(G_{CONF})_i, (G_{CONF})_2, \ldots, (G_{CONF})_n] (6)\]

\(n\) agents are considered.

Each element is a Conflict Gain associated to a specific agent. In a negotiation session there are no more than \(2n\) proposal vectors to compare. A pairwise comparison is performed by scoring the winner proposal. The agreement is the proposal with maximum score. Agent compares i-th and j-th proposal according to the vector 7. Element k-th of vector 7 is the difference between Conflict Gains of k-th agent referred to proposal i-th and j-th, as described in Eq. 8.

\[ \Delta P_{i-j} = (Proposal)_i - (Proposal)_j \] (7)

\[ \Delta G_{CONF}_{k} = (G_{CONF})_{ik} - (G_{CONF})_{jk} \] (8)

Whenever the positive elements in \(\Delta P_{i-j}\) are more than negative, proposal i-th wins. Whenever the positive and negative elements are equal, analogous comparison is made considering Contention Gains. This strategy represents a sort of majority voting. The proposal that offers an improvement to the greatest number of agents is chosen.

### 6.2. The not-concurrent strategy

In this section an alternative strategy is described. The vectors \(\Delta FF\) of all agents concerning the same proposal i-th are integrated in the global vector 9.

\[ \Delta FF_{i,\text{glob}} = [\sum_{m=1} G_{CONF} m, \sum_{m=1} G_{CONT} m] \] (9)

\(\sum_{m=1} G_{CONF} m\) is named Global Conflict Gain of proposal i-th

\(\sum_{m=1} G_{CONT} m\) is named Global Contention Gain of proposal i-th.

A pairwise comparison is performed scoring the winner proposal. Agreement is the proposal with maximum score. Comparison between two proposals is based on the following criterion. The proposal i-th wins if condition 10 is satisfied or both the conditions 11 and 12 are verified.

\[ \Delta FF_{i,\text{glob}} (2) > \Delta FF_{j,\text{glob}} (2) \] (10)

\[ \Delta FF_{i,\text{glob}} (2) = \Delta FF_{j,\text{glob}} (2) \] (11)

\[ \Delta FF_{i,\text{glob}} (3) > \Delta FF_{j,\text{glob}} (3) \] (12)

It aims to maximize the Global Gain no matter of its distribution among agents.

This strategy emphasizes the agent welfare rather than social welfare as in the concurrent strategy.

### 7. THE COMMUNICATION PROTOCOL

A protocol to manage communications has been defined. Two agents are said neighbours if they share common variables. Each agent is identified by an Intrinsic Label (IL). ILs satisfies a conventional ordering relation, which is reflected on agents. Another label is associated to agents: the Communication Label (CL). CLs satisfies a conventional ordering relations too and are dynamically assigned to agents. They need to define the oriented channels for information flow. In the first scheduling phase, all agents with a depletable resource conflict prime at the same time the propagation mechanism (introducing a new activity in the plan), while in the second phase, propagation is primed by one agent at time (who proposes the sequencing constraint). This feature leads to two different communication protocols for the two scheduling phases.

#### 7.1 The first phase protocol

The agents proceeds simultaneously to identify, select and solve depletable resource conflicts, by introducing in each iteration a regenerative activity. This means new constraints to propagate.

The first scheduling phase is developed in a synchronous fashion, by exchanging at each iteration-shared domains and verifying temporal consistency once quiescence is reached. Quiescence state is reached when all shared domains have equal values in the respective agents and no propagations are primed. Fig. 4 describes the protocol.

New constraints in agent STN prime the propagation. Such a new constraints are given or by the introduction of a regenerative activity (link ‘a’ in fig. 4) or by
reception of closer domains on common variables (link ‘b’ in fig. 4).

![Fig.4 First phase protocol flow chart](image)

The agents communicate to their neighbours the domains of the shared variables. Whenever an agent receives by its neighbours such information, it can deduce whether it is quiescent or not according to them. Two agents are quiescent if they are coordinated, i.e., they have in memory the same values for the common variable domains. Whenever an agent is quiescent (coordinated) with all its neighbours, it is in a local quiescence. When all agents are in local quiescence, no propagation is primed and the overall formation is quiescent. All agents can now verify if own scheduling profile is temporally consistent before starting a new conflict resolution.

### 7.2. The second phase protocol

Each negotiation starts from a quiescence (coordinated) state for the overall formation and the propagations are locally primed by the proposing agent. The order the agents follow in the proposal formulation is defined by the IL. The information exchange is based on a Communication Graph: a node is an agent labelled with a CL and an oriented arc stays between neighbour agents defining the path the information flow. The orientation starts from the smallest CL to the highest CL: sender (labelled with smaller CL) and addressee agents (labelled with greater CL) are, hence, identifiable. The agent in charge responsible for the current proposal has only addressee neighbours. The terminal agents close the protocol and have only sender neighbours. Whenever the proposing agent changes, the agents are re-labelled with CLs to ensure a correct information flow. The proposing agent is the starting point of a propagation and communication chain to manage the domains of the common variables. These communications satisfy the following rule: each agent waits for information on the common domains from its senders, and then it propagates and sends the resulting shared domains to its addressees. Fig. 5 describes the flow chart of the protocol.

![Fig.5 Second phase protocol flow chart](image)

After the propagation occurrence, an agent can recognize whether it is locally quiescent. The local quiescence implies no propagation to its addressees agents. Whenever the propagation accomplished by an agent turns out a tighter domain for the domains of variables shared with its senders, the formation is made apart of that local quiescence lack. A new labelling occurs starting from the not quiescent agent. The overall formation is quiescent as soon as the terminal agents are in local quiescence. The flotilla coordination is reached and both FF concerning the proposal and the respective ΔFF can be computed. This process is iterated with a new proposal.

### 7. THE SIMULATION RESULTS

The scenario adopted to verify the multi agent framework performance concerns with planetary exploration: some rovers start from an initial location to
explore the nearby area. A satellite is considered as data relay. Each rover is identified as an agent. Simulations with both a pair of and three rovers have been performed to verify the system performances whenever not only bi-univocal relations exist. The current work concerns with the prototype definition and its feasibility study; hence the computing efficiency is actually not pursued. Moreover, simulations are limited to three agents because of the low flexibility in applicative scenario reconfiguration. Ongoing work focuses on the architecture efficiency enhancement.

We ran three problems with differently sized: 1-rock, 2-rocks and 3-rocks exploration scenario, with both two and three rovers. Each activity contains two problem variables: the starting and ending time points. The features of each scenario are reported in tab.1 and tab.2. The constraints number reported include the ordering relations posted by the planner, the duration constraints, the synchronizing requirements imposed by an external user and the resource constraints.

Simulations ran on a Pentium 1.6 GHz PC with 512 MB RAM. MatLab has been used to programme the prototype. Tab 3 reports the CPU times both total and dedicated to the negotiation scheduling process according to the specific strategy adopted.

### Tab.1 Problem features in 2 rovers scenario

<table>
<thead>
<tr>
<th></th>
<th>1Rock</th>
<th>2Rocks</th>
<th>3Rocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. activities (Rover1)</td>
<td>14</td>
<td>28</td>
<td>37</td>
</tr>
<tr>
<td>No. activities (Rover2)</td>
<td>15</td>
<td>28</td>
<td>38</td>
</tr>
<tr>
<td>No. coordinated activities (Rover1)</td>
<td>6</td>
<td>12</td>
<td>16</td>
</tr>
<tr>
<td>No. coordinated activities (Rover2)</td>
<td>6</td>
<td>12</td>
<td>16</td>
</tr>
<tr>
<td>No. constraints (Rover1)</td>
<td>42</td>
<td>90</td>
<td>121</td>
</tr>
<tr>
<td>No. constraints (Rover2)</td>
<td>47</td>
<td>90</td>
<td>126</td>
</tr>
<tr>
<td>Starting conflicts for negotiated scheduling</td>
<td>6</td>
<td>22</td>
<td>45</td>
</tr>
</tbody>
</table>

### Tab.2 Problem features in 3 rovers scenario

<table>
<thead>
<tr>
<th></th>
<th>1Rock</th>
<th>2Rocks</th>
<th>3Rocks</th>
<th>2Rocks</th>
<th>3Rocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. activities (Rover1)</td>
<td>15</td>
<td>30</td>
<td>46</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No. activities (Rover2)</td>
<td>14</td>
<td>29</td>
<td>44</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No. activities (Rover3)</td>
<td>12</td>
<td>25</td>
<td>38</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No. coordinated activities (Rover1)</td>
<td>7</td>
<td>16</td>
<td>25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No. coordinated activities (Rover2)</td>
<td>5</td>
<td>14</td>
<td>22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No. coordinated activities (Rover3)</td>
<td>5</td>
<td>14</td>
<td>19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No. constraints (Rover1)</td>
<td>43</td>
<td>81</td>
<td>124</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No. constraints (Rover2)</td>
<td>42</td>
<td>84</td>
<td>126</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No. constraints (Rover3)</td>
<td>33</td>
<td>63</td>
<td>93</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Starting conflicts for negotiated scheduling</td>
<td>4</td>
<td>40</td>
<td>105</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The negotiation process time is related not only to the problem size, but also to the amount of proposals. The amount of conflicts existing at the beginning, that the negotiation must resolve is a feature influencing the

### Tab.3 CPU times per agent, in minutes and seconds

<table>
<thead>
<tr>
<th></th>
<th>1Rock</th>
<th>2Rocks</th>
<th>3Rocks</th>
<th>2Rocks</th>
<th>3Rocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU time negotiated scheduling (concurrent)</td>
<td>10</td>
<td>11</td>
<td>54</td>
<td>6:27</td>
<td>10:36</td>
</tr>
<tr>
<td>Total CPU time (concurrent)</td>
<td>11</td>
<td>16</td>
<td>1:12</td>
<td>7:38</td>
<td>12:11</td>
</tr>
<tr>
<td>CPU time negotiated scheduling (not-concurrent)</td>
<td>10</td>
<td>13</td>
<td>77</td>
<td>5:17</td>
<td>13:25</td>
</tr>
<tr>
<td>Total CPU time (not-concurrent)</td>
<td>11</td>
<td>18</td>
<td>1:35</td>
<td>6:28</td>
<td>15:00</td>
</tr>
</tbody>
</table>
total volume of proposals. Fig. 6 and fig. 7 show a linear approximated increment of the total number of proposals versus the total amount of starting conflicts. By taking into account tab. 3 and fig. 6, a concurrent strategy is expected to show better performances in a bi-univocal relations scenario (i.e. with two rovers). If all \( n \) agents record the resource conflicts, the number of proposals defined for each negotiation session is \( 2n \). The sessions last till all starting resource conflicts are solved. Being the total volume of proposals related to \( 2ns \) (\( n \) is the agent number and \( s \) is the number of negotiation sessions to reach the solution), the (13) can be considered (\( c \) is the starting amount of conflicts).

\[
2ns + c \quad (13)
\]

As a consequence, the number of negotiation sessions to reach the zero conflict scenario is expected to linearly increase according to the total amount of the starting conflicts. The total number of information exchanges among the agents for every proposal has been investigated. Whenever each agent has coordinating constraints with all other agents, the number of communication links \( C \) grows in the size of agent number \( n \), according to (14).

\[
C = 0.5n^2 - 0.5n \quad (14)
\]

The peaks of communication links have been recorded, corresponding to particular recurrent proposals formulated. This feature is present in all problems ran. Fig. 8 shows an example concerning 2-rocks problem with three rovers: peaks 23, 27, 31 (in not-concurrent strategy) and peak 29 (in concurrent strategy) concern with the same activity sequencing proposal formulated by Rover3.

8. CONCLUSIONS

In this paper a distributed planning and scheduling framework has been proposed. The flexibility is maintained during the overall algorithm. The planning is based on HD-POP engine, maintaining a least commitment strategy in finding solution. Scheduling is based on a distributed STN. The variables are not fixed but managed with domains of admissible values. Instead of binding exact negotiated values to variables, negotiated sequencing constraints are added among activities in resource conflict. This flexible time approach has the benefit to better discover the search space. Future work aims to develop the multi-agent architecture in a satellite flotilla scenario, introducing new complex agent interactions. A Consumable Resource Network coupled to the STN (i.e. cRN-STN) is an ongoing work concerning depletable resource management [11]. Mission goals are now distributed among agents by an external user. Future improvement concerns with a negotiated and autonomous distribution of goals.

REFERENCES

1. S. Das, P. Gonsalves, R. Krikorian, W. Truszkowski, Multi-Agent planning and scheduling environment for enhanced spacecraft autonomy Proc. 1° ICMSAS.
7. R. Dechter, I. Meiri, J.Pearl, Temporal Constraint Networks Artificial Intelligence 49(1991) 61-95