

# Multi Agent Conflict Partition Scheduling for Coordinated Space Systems

**Andrea Brambilla and Michèle Lavagna**

Politecnico di Milano - Department of Aerospace Engineering  
via La Masa, 34  
20156 Milano, Italy  
brambilla@aero.polimi.it

## Abstract

The Conflict Partition Scheduling (CPS) is a methodology that builds solutions by repeatedly identifying bottlenecks and posting constraints to solve them. As a result, a temporally flexible schedule is gained, that offers an envelope of executable legal behaviours. CPS has been focused to single agent (e.g. a single rover or spacecraft) scenarios so far, according to a centralized scheduling strategy. The growing interest on space missions composed by coordinated space systems (e.g. spacecraft flotillas and rover teams) makes distributed scheduling technologies even more attractive to develop. The paper extends the CPS approach to a distributed multi-agent scheduling environment. Running isolated and parallel CPS processes does not guarantee the consistency of inter-agent constraints. The here proposed methodology concerns with controlling the bottleneck conflict partition of each agent through a negotiation mechanism. In particular, each sequencing constraint becomes an agent proposal to negotiate according to a specific strategy. Negotiation mechanism models a majority voting system and reduces the ongoing conflict partitions to propose. The constraint propagations are performed in a Distributed Simple Temporal Network (DisSTN) to keep temporal consistency of the agent society. An analysis on differently sized problems and the results of a two rovers scenario are also offered.

## Introduction

A growing interest is focused on space missions based on multiple systems working in a coordinated fashion. As examples, we remind ongoing ESA missions with spacecraft flotillas (DARWIN and LISA) and all future explorations with robotic teams. Two possible strategies can be defined to schedule activities of multiple systems.

The first one is a centralised approach. All the knowledge base (i.e. all problem variables and constraints) resides on a single unit (agent) which is devoted to solve the global scheduling problem. The schedule components are extracted from the global solution and distributed to the respective agents for execution.

The second one is a distributed approach. Each agent has an own local and partial knowledge base and runs a local scheduling problem. A coordination mechanism regulated by a communication protocol has to be defined among the

units in order to satisfy inter-agent constraints (e.g. the execution of operations of different robots at the same moment or the reservation for a unique central antenna).

The centralised approach shows to be less robust than the distributed approach. The loss of the central scheduling unit means the failure for the overall system. According to a distributed technology, each agent has an own autonomy degree and reconfiguration capability whenever a unit is loss.

Conflict Partition Scheduling (CPS) is a methodology to cope scheduling problem by exploiting temporal flexibility (Muscuttola (1992)). CPS has been encoded in HSTS framework (Muscuttola (1994)). HSTS is the planning/scheduling kernel of the Remote Agent software (Muscuttola *et al.* (1998)). It ran as autonomous control system on the on-board computer of Deep Space 1. CPS builds solutions by repeatedly identifying bottleneck conflicts and posting constraints to solve them. Every time a variable is assigned during problem solving, the search space loses one dimension. Posting a constraint only restricts variable domains without necessarily decreasing the search space dimension. As a result, a temporally flexible schedule can be gained and an envelope of legal behaviours at execution time is offered. CPS methodology was defined and studied with a centralized approach and only one agent.

This paper extends CPS to distributed scheduling problems. In this way, we aim to propose a new methodology maintaining flexibility (given by a least commitment scheduling approach like CPS) and robustness (given by a distributed framework).

We have also to highlight that a CPS based approach is effective only with not depletable resource conflict. So, we will consider only this class of resource in the remainder of the paper (e.g. power generated from solar arrays, antenna and unary devices). An analogous algorithmic basis has been adopted in Brambilla *et al.* (2005) and Armellin *et al.* (2005) to schedule activities respectively of a rover team and of a spacecraft flotilla simulation model.

According to a distributed problem, each agent is devoted to schedule its own set of tasks. Temporal and resource capacity constraints may involve activities coming from different agents. Therefore, the scheduling problem solving of each agent can not be performed with no team interaction as the scheduling decisions of CPS may affect the other agents schedule envelope too. We need to introduce

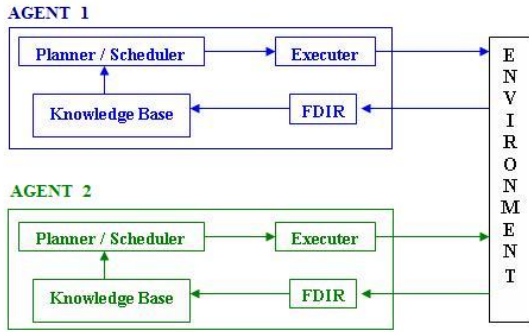


Figure 1: Framework overview. One colour identify one agent.

Distributed Simple Temporal Network (DisSTN) to propagate single constraint settlement to the Simple Temporal Networks (STNs) of all the agents (for STN description see Dechter, Meiri, & Pearl (1991)). We will define in the following sections how to maintain the temporal coherence of the agents while managing the constraint settlements through a negotiation process. Negotiation strategy models a majority voting, so common benefit is pursued during problem solving.

Result of the proposed methodology is a schedule envelope for each agent that can be made dispatchable for efficient execution (Muscettola, Morris, & Tsamardinou (1998)). Sensitivity of different sizing problems is critically analysed by discussing simulation results.

## Framework Overview

This section introduces the overall framework developed to encode the proposed distributed CPS methodology, focusing on Planner/Scheduler module. For more details see Brambilla *et al.* (2005).

The same framework runs in each agent of the agency. Fig.1 reports two agents even if the architecture can manage a greater number of agents.

Planner/Scheduler (PS) module takes as input the sets of constraints and variables in the Knowledge Base formalising the operational scenario. Output is sent to Executer under dispatchable temporal network. A Dispatchable Execution Controller generates commands for actuators in the Executer module. Failure Detection Identification and Reconfiguration (FDIR) is devoted to build expected state from sensors feedback, detecting and identifying displacements from nominal state. In this way, FDIR updates constraints in the Knowledge Base closing the control loop among agent modules.

PS is composed by two layers. The first one is the Planner Layer and the second one is the Scheduler Layer.

Planner Layer is based on the Hierarchical Decomposition-Partial Ordered Planning algorithm (HD-POP). Each activity is described with preconditions and effects and can be considered as a state transition operator. A decomposition in a sub-activities plan can also be associated to every activity. Input of the HD-POP is an initial partial plan encoding

the initial state (as effect of the 'start' activity) and the goal state (as precondition of the 'end' activity). The algorithm builds causal link chains among preconditions and effects of the activities.

Each agent run a first local planning process to build a local plan.

All local plans are sent to the agent elected as *Coordinator*. The coordination role is limited only at Planner level, working on causal links between different agents. Criteria to elect the Coordinator can be defined in many different ways but the framework of the Coordinator is not different by other units. So any agent can be elected as Coordinator, maintaining the robustness of the agency. The coordination algorithm invoked is the HD-POP itself, but taking as input the global partial plan instead of the local initial partial plan. The global coordinated plan is formalised as an acyclic oriented graph, where nodes are planned activities and oriented arcs are ordering relations.

In fig.2 is reported an example with two robots (two agents) devoted to assist crew during exploration. This case will be used in the paper to better explain the multi agent CPS methodology. Robots receive samples collected by crew and coordination constraints guarantee the alternation of robots in storing samples next to the crew. 'Link' is considered as an uplink demanding for the reservation of the orbiter antenna, modeled as a common and unary resource.

Coordinator returns to agent  $A_i$  the respective coordinated plan, that is the original local plan augmented with activities of other agents sharing a constraints with  $A_i$  (in fig.2(b) 'link' activity shares the resource antenna and 'receive sample' is temporally constrained with the other agent).

Coordinated plan is the input to the scheduling process of each agent and is transformed in a STN with resource allocation.

The Scheduler Layer is devoted to reach the consistency of both metric temporal constraints and resource constraints. It is governed by the multi agent CPS methodology described in detail in the remainder of the paper. We need to introduce Distributed Simple Temporal Network (DisSTN) to propagate local constraint partition to the Simple Temporal Networks (STNs) of all the agents. We will define in the following sections how to maintain the temporal coherence of the agents. We will also describe the management of the constraint partition through a negotiation process during problem solving. Negotiation strategy models a majority voting, so common benefit is pursued during problem solving.

## Distributed Simple Temporal Network

Each planned activity generates two variables for the STN problem: the start time point and the end time point. A finite temporal domain is associated to each variable. Start time point ranges between bounds  $[est, lst]$  *est: earliest start time, lst: latest start time*. End time point ranges between bounds  $[eet, d]$  *eet: earliest end time, d: deadline*.

Coordinated plan is the input to the scheduling process of each agent and is transformed in a STN. The single STN includes not only variables executable by the agent itself but also variables controlled by external agents (e.g the starting and ending time points of 'link' activity of agent-2 in the

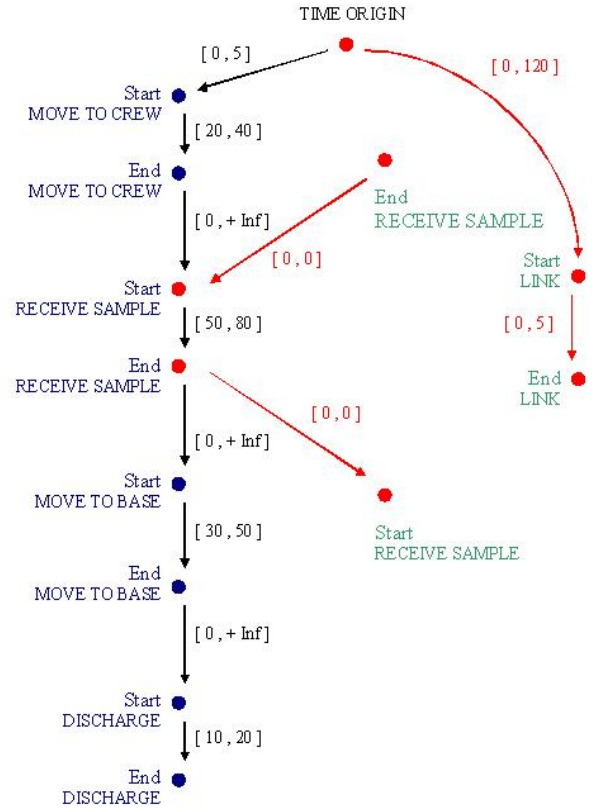
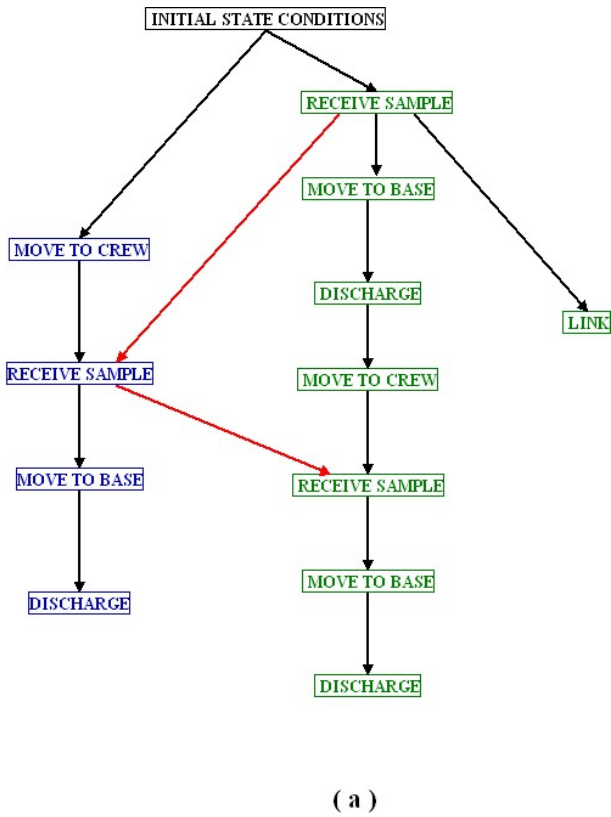


Figure 3: Agent-1 STN.

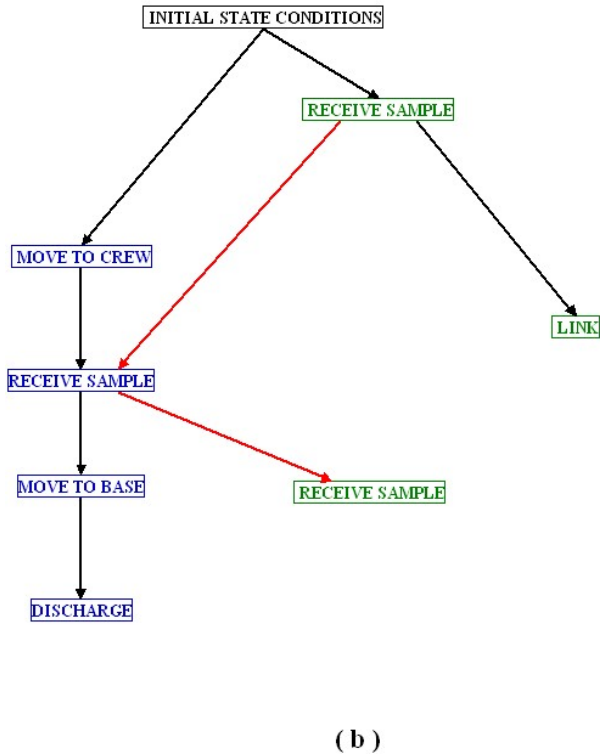


Figure 2: (a) global coordinated plan. (b) agent-1 coordinated plan. Activities of agent-1 local plan are labeled in blue. Activities of agent-2 local plan are labeled in green.

STN of agent-1, see fig. 3). A same variable is present in STN both of agent  $A_i$  and of agent  $A_j$  whenever it is involved in an inter-agent constraint between  $A_i$  and  $A_j$ . For instance, the starting and ending time points of a link activity performed by the robot  $A_i$  to a central shared antenna are present in the STN of robot  $A_j$ , even if  $A_j$  has not the execution control on those time points. This class of variables will be referred as *shared variables*. Fig.3 reports agent-1 STN obtained from plan in fig.2(b). It is augmented with variables controlled by agent-2 (labelled in green). Blu nodes represents agent-1 own variables, red nodes are shared variables.

Shared variables make all coordinated events (e.g. temporal bounds of a link start time) visible to each agent. This feature provides to the agent all the informations to locally reason on inter-agent constraint satisfaction. Shared variables represent the specific part of the agent Knowledge Base involved in not local constraints. Each shared variable is present in more local STNs and represents the same physical event. For this reason it must be synchronised. *Synchronisation* requirement means the maintenance of the same temporal domain in each STN for every shared variable. For instance, two different rovers have to not consider the starting time of the same communication link at different temporal intervals for the execution. This is an important

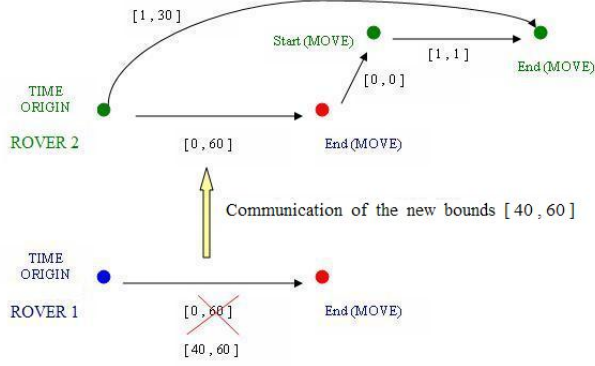


Figure 4: Interaction of local STNs. Red node is a shared variable. Blue and green identify the two agents

issue to satisfy the temporal coherence.

### Temporal Coherence Maintenance

The maintenance of temporal coherence of distributed scheduling processes is here addressed. A constraint propagation in a STN makes the synchronisation of shared variables no more satisfied because of the adjustment of the variable bounds. Whenever a new constraint is propagated in  $STN_i$ , then agent  $i$  has to communicate to agent  $l$  the new domain of the shared variable. This information means new constraints to propagate in  $STN_l$ . In this way, any propagation in  $STN_i$  is extended to  $STN_l$ . This kind of interaction is invoked until synchronisation is reached, determining a new schedule envelope for each agent. We have to highlight that a local propagation of a new constraint may bring inconsistency to a STN of a different agent. Fig.4 shows an example. The end time point of rover-1 moving activity is contemporary to the starting time of the moving activity of rover-2, lasting 1 temporal units. The deadline of the rover-2 moving activity is set at 30 units. Let us suppose that a propagation of a new constraint in the rover-1 STN shifts the *est* of rover-1 to 40 time units. This new bound for the shared variable brings an inconsistency in the rover-2 STN. *est* of rover-2 moving activity becomes 41 units but the deadline is 30 units.

A protocol to manage the communications of the shared variable domains among agents is proposed in (Brambilla *et al.* (2005)).

### Bottleneck Conflict Partition

In previous sections the temporal consistency problem has been addressed. The synchronisation mechanism assures the maintenance of temporal coherence.

In the remainder of the paper the resource constraints satisfaction will be addressed in a distributed way through multi agent CPS.

Running isolated and parallel CPSs does not guarantee the consistency of inter-agent constraints (e.g. synchronisation

of shared variables or satisfaction of common resources capacity as the reservation of a same antenna). The here proposed methodology concerns with running CPS processes in parallel and controlling the bottleneck conflict partition of each process through a negotiation mechanism. In particular, each sequencing constraint becomes an agent proposal to negotiate according to a specific strategy. This condition do not enforce the instantiation of the problem variables. The algorithm is required to generate a set of constraints which are proved compatible one with the others and represent not a single schedule but a set of possible schedules. This become useful when the results for each agent are put into execution. While unforeseen events (e.g. an activity lasting longer than expected) would normally invalidate the unique schedule, the same events may only reduce the set of schedules represented by the current constraints. The distributed scheduling process iteratively performs the following steps:

1. Conflicts identification
2. Bottlenecks detection
3. Proposals definition on conflict partitions
4. Negotiation of proposals and agreement election

Process ends as soon as no conflicts are identified for all the agent schedule envelopes. Following subsections describe each step. Our algorithm is based on CPS and introduces some different algorithmic choices if compared to (Muscettola (1992)). They will be highlighted in the remainder of the paper.

### Conflict Identification

In this work we consider not depletable resources, both with unary and cumulative capacity. As a result of scheduling process we have for each agent an STN that can be made dispatchable and satisfying not depletable resource constraints. The possibility to fall into a (not-depletable) resource conflict during dispatching execution must be prevented during conflict partitioning. A conflict on resource  $r$  is defined whenever a set of activities requiring a total amount of  $r$  greater than available capacity may be scheduled in a common temporal interval. Algorithm reasons about this possibility creating a weighted graph of contemporary activities. Nodes of the graph are activities demanding for  $r$ , respective weight is the capacity required by the activity itself and an arc is posted between activities  $A_k$  and  $A_j$  whenever condition (1) is verified and an existing constraint that specifies that the two actions are ordered is not present.

$$[est_k, let_k] \cap [est_j, let_j] \neq \emptyset \quad (1)$$

*est* is the earliest start time of activity.

*let* is the latest end time of activity.

A weighted cliques problem is solved on the previous graph (Carraghan & Pardalos (1990)). Only cliques having a total weight greater than the available capacity of the resource are considered as conflicts (fig.5).

In the remainder of the paper we will consider cliques and conflicts in the same way. Each clique is a set of activities not prevented to be executed in a common temporal interval

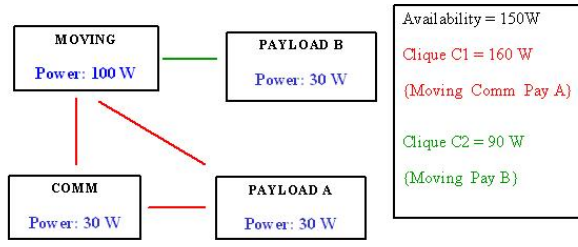


Figure 5: Conflict detection on a weighted graph of contemporary activities

and giving a not consistent demand for  $r$  whenever executed in contemporary moments.

### Bottleneck Definition and Partition Proposals

Original CPS process determines bottleneck conflict using a Stochastic Capacity Analysis. It runs a stochastic simulation of the STN for  $N$  times, gathering statistical metrics (i.e. token demand and resource contention) (for more details see Muscettola (1992)).

In a distributed environment this could bring a lot of computational effort: the single value assignment during stochastic simulation invokes a chain of STN propagations among agents to maintain the temporal coherence of the shared variables. We adopted a different solution for the commitment on the activities to sequence. This method is based on two texture measurements: Reliance and Contention that are computed through an a priori estimation and no propagation is required, as proposed in Beck, Devenport, & Sitarski (1997).

Reliance is estimated by an activity's probabilistic individual demand for a resource over time, while contention is the aggregation of the individual activity demands.

To calculate an activity's individual demand, a uniform probability distribution over the possible start times of the activity is assumed: each start time has a probability of  $1 \div (lst - est)$  ( $est$  is the earliest start time,  $lst$  is the latest start time). In Beck, Devenport, & Sitarski (1997) an event-based representation and a piece-wise linear estimation of the Reliance curve is proposed. A uniform probability distribution is the "low knowledge" default. It may be possible to use some local propagation in the constraint graph to find a better estimate of the individual demand (see Sadeh (1991), Muscettola (1992)). This approach has not been adopted because each propagation brings new communications to synchronise shared variables.

To estimate Contention, the individual demands of each activity are aggregated for each resource. Aggregation is done by summing the individual activity curves asking for that resource.

Once computed all contention curves for each resource  $r$ , agent integrates every single function over the Scheduling Horizon determining a global contention index for each resource (represented with  $C_r$ ). Commitment to make a sequencing proposal focuses on the resource with the greatest  $C_r$  and on the clique whose activities contribute to  $C_r$  the

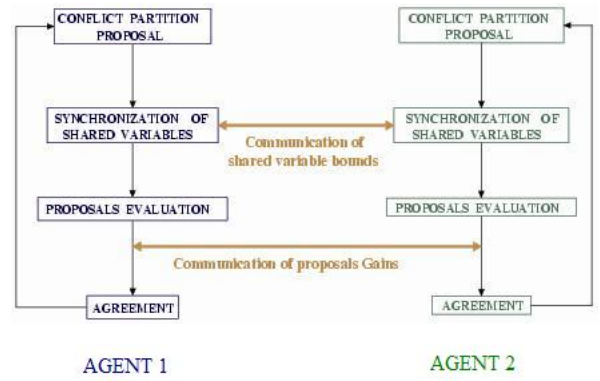


Figure 6: Multi-agent CPS logical flow. One colour identify one agent

most. The single activity contribution (i.e. reliance degree) to the contention degree for  $r$  can be estimated with the integral of the individual demand on the interval  $[est, lst]$ . The two activities most reliant on  $r$  in the critical clique,  $A$  and  $B$ , take part to the sequencing proposal. Propagation of the ordering constraint  $A$  before  $B$  (or  $B$  before  $A$  as well) in a local STN determines new propagations in STNs of other agents to synchronise shared variables. The consequence is a new schedule envelope for each agent. This new schedule envelope so obtained is referred with the attribute *proposed*. Schedule envelopes of the agents before any proposal propagation is referred with the attribute *current*.

### Negotiation

Fig.6 reports the logical flow of the distributed scheduling algorithm.

Only proposals guaranteeing temporally consistent synchronisation are negotiated. Every agent evaluates consequences of each proposal (i.e. schedule envelope consequent to the partition) according with two criteria. The first one is based on the definition of the Conflict Index Gain ( $G_{conflict}$ ) which is formalised in (2):

$$\left( \sum_{c \in CL} Card(c) \right)_{Current} - \left( \sum_{c \in CL} Card(c) \right)_{Proposed} \quad (2)$$

$Card(c)$  is the cardinality of the conflict  $c$  (i.e. the number of the clique nodes).

$CL$  is the set of all cliques.

The second one is based on the definition of the Global Contention Index Gain ( $G_{contention}$ ) which is formalised in (3):

$$\left( \sum_{r \in R} C_r \right)_{Current} - \left( \sum_{r \in R} C_r \right)_{Proposed} \quad (3)$$

$R$  is the set of all agent resources.

The greater those indexes are, the greater the preference for the proposal is. In general, each proposal gives rise to different consequences for every agent in terms of  $G_{conflict}$  and  $G_{contention}$ . This means different preferences over the set of proposals.

In the remainder of this section we define a negotiation strategy to reach an agreement on constraint settlement. Agreement is computed as soon as every proposal of the session has been evaluated and every agent has informed the agency about own proposal gains ( $G_{conflict}$  and  $G_{contention}$ ). In the memory of each agent the  $i$ -th proposal is stored as a couple of vectors. Elements of the first vector are Conflict Index Gains:

$$(Proposal)_i = [(G_{conflict})_1 \dots (G_{conflict})_n] \quad (4)$$

Elements of the second vector are Global Contention Index Gains:

$$(Proposal)_i = [(G_{contention})_1 \dots (G_{contention})_n] \quad (5)$$

Each element is associated to a specific agent.

Agreement is locally computed by each agent in the same fashion (i.e. with the same strategy). In this way, the agreement is univocally determined without any further communication. Backtracking occurs whenever all proposals formulated for a given session are temporally inconsistent for at least one agent. A pair-wise comparison is performed among all proposals. The score of the winning proposal for every comparison is augmented by one. The agreement is the proposal with maximum score. The comparison between two proposals is based on vector (6) where vectors  $Proposal_i$  and  $Proposal_j$  are defined as (4) or (5).

$$\Delta P_{ij} = Proposal_i - Proposal_j \quad (6)$$

Whenever the positive elements in  $\Delta P_{ij}$  are more than negative, proposal  $i$ -th wins. Whenever the positive and negative elements are equal, an analogous comparison is made considering vectors  $Proposal_i$  and  $Proposal_j$  as in (5). Hence a hierarchical preference criterion between  $G_{conflict}$  and  $G_{contention}$  for the agency point of view is modelled.

This strategy models a majority voting system and reduces the ongoing number of proposals to negotiate. In fig.7 is shown an example. Oriented arcs are ordering constraints. Solid line identifies the selected clique (i.e. conflict) to partition. Dashed lines are other identified cliques. Fig.7(a) reports the current conflict situation. According with the partition proposal in fig.7(b)  $G_{conflict}$  is 1. According with the partition proposal in fig.7(c)  $G_{conflict}$  is 3. Preference is given to (c) presenting only one ongoing clique to solve.

Success of negotiation session is guaranteed and no cycle loops in finding agreement occur. Each proposal formulation means not only a computational effort but also communication among agents to maintain temporal coherence.

Next section will show the sensitivity analysis with respect the number of negotiation sessions established and will report the results on a multi rovers scenario for crew assistance.

## Results

Each negotiation session means proposals to compute and communications to establish. So we ran test problems with

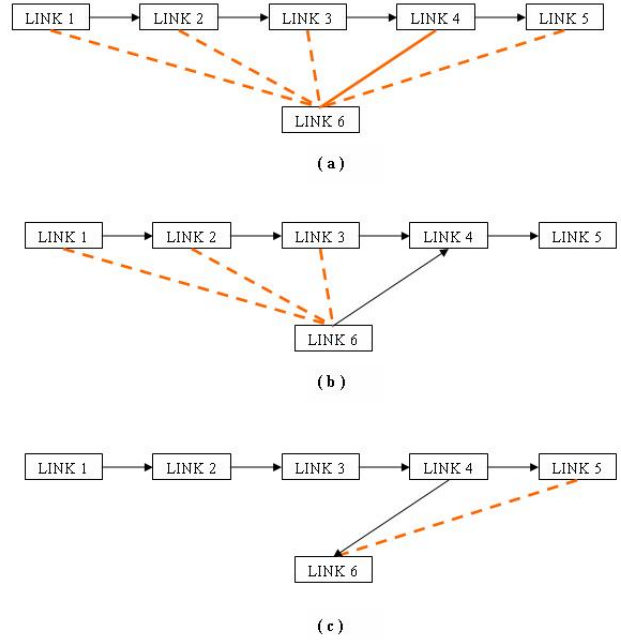


Figure 7: Example of proposals. (b) and (c) show two sequencing proposals on conflict situation reported in (a)

agency growing from 2 to 8 agents ( mission DARWIN of ESA is composed by a flotilla of 8 spacecrafts) in order to analyse the trend of the number of total negotiation sessions. We considered one common resource shared by the agency and one local resource. Each agent has ten activities to schedule and about 20 % of synchronised variables.

Fig.8 shows the conflict index ( $\sum_{c \in CL} Card(c)$ ) trend per agent along the negotiation session for the largest problem (8 agents). The number of sessions to reach the solution is related to the amount of the starting conflicts index and to the presence of local resource conflicts. We can highlight that agreements of final sessions reduce conflicts of only one agent (i.e. from session 13). This feature is characterised by agreements resolving local resource conflicts.

Each evaluation of a proposal means an amount of communications established among agents to synchronise variables. The single communication involves two agents and concerns with shared variable domains exchanging. Fig.9 reports the total number of communications established per proposal for the largest problem (8 agents). Low degree of communications established concerns with a local resource conflict partition proposal

Fig.10 shows the trend of total number of negotiation sessions with respect the starting conflict index. Increment of the starting conflict index per agent is determined by introducing a new agent every time (i.e. more conflicts on the common resource).

The red results in fig.10 consider only one common resource in the problem formulation. Blu results take into account also the introduction of a local resource for each

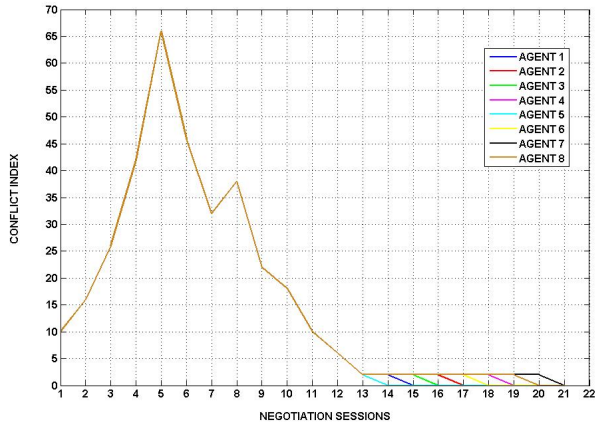


Figure 8: Conflict Index trend per agent (8 agents problem)

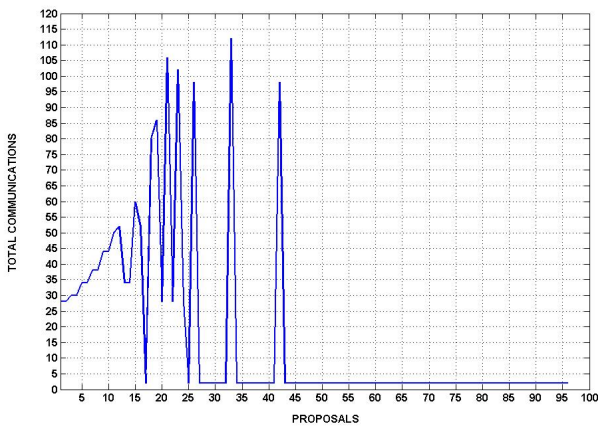


Figure 9: Total communications to synchronise variables after each proposal (8 agents problem)

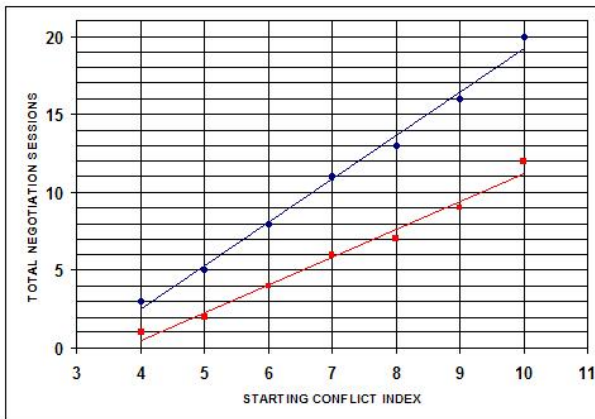


Figure 10: Sensitivity of the total number of negotiation sessions

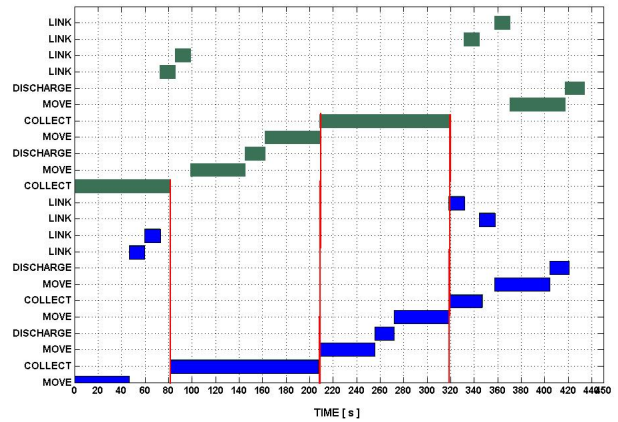


Figure 11: Earliest schedule profile. One colour identifies activities of one agent

agent. A linear trend is maintained in both the situations. Fig.8 and fig.10 highlight the strong impact of only one local resource with respect to the number of negotiation sessions. This suggests to modify the strategy by accepting multiple agreements for the same session whenever more agents propose local conflict partitions.

We ran a study case scenario of two robots devoted to assist crew during exploration (fig.11). Robots receive samples collected by the crew and carry the samples to a base. Each transporting rover moves from crew to base and viceversa. Coordination constraints guarantee the alternation of robots in storing samples next to the crew. As soon as one rover leaves the crew to back base, the other rover takes its place and vice versa, so as to guarantee a permanent presence of one of the two transporting systems next to the crew. Red lines in fig.11 show the satisfaction of this coordination requirement. 'Link' is considered as an uplink demanding for the reservation of the orbiter antenna, modeled as a common and unary resource. Fig. 11 shows that link activities are all ordered satisfying the resource constraint.

## Related Work

In Hunsberger (2003) is proposed an alternative approach to control distributed temporal networks. The basic idea is to decouple a global STN in independently controllable subnetworks. If each agent maintains the consistency of its subnetwork, the consistency of the global network will be ensured. The decoupling algorithm computes a set of constraints to add to the network sufficient to temporally decouple specified subnetworks. On the one hand temporally decoupled networks can sometimes be tightly constrained losing in flexibility, on the other hand we do not need of communications to keep temporal coherence among the networks.

This kind of approach is suited for applications where agents may be out of communication for long period of time (e.g. surveying a large terrain area that has hills or large rocky areas that can obstruct communication).

In other applications agents may all operate in a general area

where communication is relatively inexpensive (e.g. rovers working in close range to build a structure or habitat) and the flexibility is a viable issue to cope the great amount of coordinating constraints and execution uncertainty. The methodology proposed in this paper is suited for this last kind of applications. In fact, the temporal coherence maintenance mechanism preserves the flexibility of agent STN and no temporal decoupling constraint is added to the network. However, this advantage is paid by overhead of communication to synchronise the shared variables.

## Conclusion

In this work we proposed a development of the CPS methodology to solve distributed scheduling problems. We extended the STN formalisation to a DisSTN to keep temporal coherence of the agency. Conflict partitions are coordinated through a negotiation process of the sequencing constraints settlements. Negotiation mechanism models a majority voting system and reduces the ongoing conflict partitions to propose. Algorithm reaches the solution without fall in cyclic loops.

This work will be developed taking into account depletable resources during problem solving (e.g. on-board memory, battery).

The mechanism to elect the agreement will be revisited in order to better cope local resource conflicts.

Analysis of the proposed methodology on benchmark problems will also be considered.

## References

- Armellin, R.; Brambilla, A.; Lavagna, M.; and Finzi, A. 2005. A distributed approach for space system formation operation autonomous planning and scheduling. *Proc. XVIII AIDAA*.
- Beck, C.; Devenport, A.; and Sitarski, E. 1997. Texture-based heuristics revised. *Proc. of AAAI97*.
- Brambilla, A.; Lavagna, M.; Costa, A. D.; and Finzi, A. 2005. Distributed planning and scheduling for space system flotillas. *Proc. iSAIRAS'05*.
- Carraghan, R., and Pardalos, P. 1990. An exact algorithm for the maximum clique problem. *Oper. Res. Lett.* 9:375–382.
- Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal constraint networks. *Artificial Intelligence* 49:61–95.
- Hunsberger, L. 2003. Distributing the control of a temporal network among multiple agents. *Proc. AAMAS-03*.
- Muscettola, N.; Nayak, P.; Pell, B.; and Williams, B. 1998. Remote agent: To boldly go where no ai system has gone before. *Artificial Intelligence* 103:548.
- Muscettola, N.; Morris, P.; and Tsamardinos, I. 1998. Fast transformation of temporal plans for efficient execution. *Proc. AAAI'98*.
- Muscettola, N. 1992. Scheduling by iterative partition of bottleneck conflicts. CMU-RI-TR-92-05, Carnegie Mellon University.

Muscettola, N. 1994. *Integrating Planning and Scheduling*. Morgan Kaufmann Publishers. 195–196.

Sadeh, N. 1991. Lookahead techniques for micro-opportunistic job-shop scheduling. Ph.D. Thesis, CMU-CS-91-102, Carnegie Mellon University.