Proceeding of the 7th International Symposium on Artificial Intelligence, Robotics and Automation in Space: i-SAIRAS 2003, NARA, Japan, May 19-23, 2003

FUZZY LOGIC AND DECISIONAL PROCESSES TOWARDS A SPACE SYSTEM ACTIVITY AUTONOMOUS SCHEDULER

Lavagna M.R. [1]- Ercoli Finzi A.[1] - Da Costa A.
[1]Politecnico di Milano - Dipartimento di Ingegneria Aerospaziale
Via La Masa 34, 20158 Milano, Italia
lavagna@aero.polimi.it

Keywords: Autonomous Planning-Scheduling, AHP, Fuzzy Logic, Constraint Programming

Abstract
The paper presents a method to face the problem of autonomous activity allocation in time during in flight operations within a space mission framework. Both constructive and repair methods are used, sequentially, to obtain a first attempt allocation scenario and to solve the risen violations in terms of resources. In particular, a backjumping algorithm is applied to achieve a first attempt allocation scenario according to temporal constraints; to manage the multi-level, multi-criteria decision making (MCDM) in solving other resource inconsistencies risen from the first activity allocation the Analytic Hierarchical Process (AHP) technique has been selected as a good choice to identify solutive actions. Moreover, in order to obtain an on-line autonomous scheduler the operators' and engineers' knowledge - needed to fill the matrix of pairwise comparison according to the current system status - has been captured by implementing several Fuzzy Logic (FL) blocks. Simulations - run both by generating random activity demand and by interfacing the system with an on-the-shelf orbit propagator s/w - showed the ability of the proposed method to obtain – from high-level activity requests- a low-level violation free scenario almost ready to be interfaced with an *executer* module with short computational time slots and a high allocated-to-requested activity ratio.

1. Introduction

Within the space community, on-board autonomy enhancement is gaining more and more terrain, as it represents the key to make challenging missions feasible within both the near Earth and interplanetary exploration space mission framework. Increased system autonomy means: a better system utilization because of an on-line system status knowledge before making decisions; a healthier system management because of the timeliness in detecting state parameters deviations from nominal trends and in a prompt system reconfiguration consistent to possible failures; a larger number of space systems to be simultaneously managed by the control centres because of required on ground operation reduction in number, time and effort; the feasibility of space missions far from Earth for which, currently, the TT&C operation management turns out to be unfeasible for the signal time delay because of distance; the feasibility of unknown place explorations because of the possibility to have adaptive intelligent systems able to learn from the current sensored data, and to settle a dedicated rule base. Obviously several drawbacks arise such as the large effort required to implement and initialize systems devoted to the autonomy management, as they normally require a deep base of system knowledge to accomplish tasks normally devoted to humans; algorithms devoted to accomplish autonomy tasks must be highly reliable and robust; they must be easily reconfigurable whenever required during the mission horizon; computational costs must be low because of limited on board resources (in terms of time and h/w).

As it clearly appears, on-board autonomy covers a wide range of application fields, the widest of which are the diagnosis field and the planning/scheduling field.

The current work focuses on the last one and proposes a method to autonomously allocate high-level activities asked to a space system during its in-flight operations. Planning and scheduling features are accomplished in that the system builds the correct activity sequence to achieve state and resource values required to satisfy high level requests (planning) and allocates those activities in time in a way that resources, states, timing constraints are satisfied (scheduling). Re-scheduling performance are also obtained as new high-level requests can be asked to be inserted during the in flight operations, by rapidly revisiting the current scheduling scenario.

According to the available related literature, scheduling problems can be solved by either a constructive or an iterative repair method: the first approach mainly makes use of Constraint Satisfaction (CS) techniques while the second approach is based on dedicated heuristics (the random search included) to search and solve current violations [1], [2], [3], [4], [5], [6]. Space domain applications are quite recent, as the correspondent domain is complex and should be highly adaptive. The most important autonomous schedulers available in literature are the planner/scheduler included in the Deep Space Mission remote Agent (RAX) [7], [8], and the ASPEN system [2], [6], [9], and [10]. The first one is based on the state variable characters expanded in time by the 'token' unit. Based on stochastic simulation to allocate the tokens and to rank identified bottlenecks, it creates new constraints among involved tokens to assure future consistency.

The Aspen system is largely applied (e.g.EO-1, AMM-2, CX-1, UFO-1 space missions). Thought to be a valid help to the on ground operators, one of its targets was represented by a modelization language easily understandable by operators, not expert of the AI domain. The algorithms are based on the iterative repair approach. A possible constructive approach can be selected to generate the first attempt scenario. To the conflict solution several heuristics have been settled: in particular, six for the conflict selection, three for each possible

action to apply to solve the conflict: while the action to apply to solve is randomly found.

Starting from its formalism the current work suggests a different approach to the conflict violation issue based on the human decision making qualitative behaviour simulation.

1. The Proposed Approach

In the followings, the proposed method is presented. An iterative repair method has been chosen to implement the core of the scheduler; a CS approach is applied only to define a first attempt scenario. Space systems are quite complicate in terms of on-board resource and activities have a complex interdependency net both in the time and resource domain; moreover, several constraints, especially in terms of available resources, are highly activity allocation dependent; an example could be the energy resource: a conflict involving that resource typically rises after having already allocated a large part of activities and a backtracking procedure to select the activity responsible of that violation is a quite hard task. On the contrary, the iterative repair methods permit to have a glance of a first (possibly inconsistent) attempt scenario and to select different criteria to guide the conflict solution. That former aspect is a point of weakness and strength: the weakness stays in the first attempt allocation technique definition, that highly influences the consequent conflict scenario, and in the numerous decisional nodes to be solved; the strength stays in its versatility, that is it is better suited to answer re-scheduling skills, as new inserted/deleted high-level requests mean-for sure-new conflict that can be managed as a normal conflict repair situation.

In order to explain better the method, a brief overview of the domain language is firstly given. The knowledge base (KB), the scheduler works on, is then explained; the first attempt allocation and the conflict detection and solution techniques are then exposed. Finally results obtained by simulating an Earth observation mission are given.

1.1. The domain Language

The domain formalism has been settled having – as goal - the same the ASPEN system had: an easily configurable language, near to the human thinking, any time the scenario changes [10].

Three main modules have been settled to model the system both from a static and dynamic point of view: a knowledge base (KB) is asked to be instantiated by the domain experts following a pre-defined formalism; a database (DB) contains informations about the current environment; a decision-maker (DM), strictly related to the adaptive dynamic scheduler, represents the inference motor between the KB and the DB.

1.1.1. The Base of Knowledge

The main characters of the whole representation is the *activity: a*n *activity* is defined as a specific procedure required to the system either by the users, by the system itself or by the external environment: hence, it clearly appears that an activity can be related either to a single, to some or none on-board subsystems: in the space application example, a down-link, a p/l usage request, an eclipse time and an attitude maneuver can be classified as *activities*. Moreover, the *activity* domain is further specified in order to manage with the different nature each *activity* can have: *Activity1+-; Activity2+-; Sub-activity; External.*

Activity: Each high-level procedure, completely independent from any other procedure is an *activity*. Depending on its temporal constraints it is classified as an activity1 or activity2: in particular, *activity1* collects procedures that can be allocated only in particular, fixed time windows because of their dependency from the physics of the system (TT&C links or Earth observations re-enter in that category). *Activity2* can be allocated whenever required, not being related to a particular system position and attitude vector.

The further '+' and '-'taxonomy involves the procedure relationship with particular resource utilization: *activity+* comprehends those procedures that have resource consumption while *activity-* represents procedures devoted to the resource renewal. As an example, a taking-image is an *activity1+*, while a particular attitude maneuver devoted to reduce the sun aspect angle is an *activity2-* procedure.

Each activity is defined by a set of parameters $\underline{X}$, called *state variables (SV);* SVs have specific domain to vary into:

$$(\forall x_i \exists X^*_i \mid x_i \in X^*_i) \quad X_i \subseteq R^n \vee \quad X_i = \text{natural language} \qquad \text{eq. 1}$$

As an example, Fig. 1 shows part of the 'imaging' *activity* settlement and the related SV set definition.

High-level activities can be classified as boundary activities (bound) if they are considered suitable for temporarily define the end of the current scheduling horizon as well as the beginning of the next one. A possible boundary activity is a downlink activity as it is strictly related to the updating of the high-level activity request set (TT&C contact allows the p/l observations up-link to the system). Within the current work, the admissible values for each state are called *attributes* .Legal transitions between different *attributes* must be settled ; recalling the former example legal transitions are given in the *camera* state variable off-line definition as: 'off→warming; warming→on'.

The legacy is given through a square matrix NxN where N is the number of admissible attributes: whenever a high-level activity is required, a particular *attribute* for each related *SV* must be achieved from the current one, through a legal transition path. Legal transition paths, between current and desired *SVs* may involve added activities, here called s*ub-activities.* Just to clarify, an imaging a*ctivity* requires the SV *attribute* to be (see Fig. **1**): *camera-on; attitude-par; vibration-low;* the default *attribute* for the s/c attitude state variable is 'nadir', while its *attribute* during 'imaging' is represented by a particular Euler angle set dictated by the relative position target-camera; hence a transition is activated to achieve the final configuration; in particular the transition *nadir→ par* is not legal ($a_{1,4}=0$) while the $a_{1,5}=1$ it is (nadir-> rotational).

Hence, a further ancillary *activity* with the *SV*= attitude-rotational must be inserted to obtain the initial-to-final transition: in that case the *slew sub-activity* must be settled, in

order to consider a time and resource slot (within the *activity* allocation) dedicated to the target acquisition maneuver whenever a high -level imaging activity is requested . Relationships between activity *SV* attributes and *sub-activity* calling are application dependent and can be settled by initializing the KB once, before the real-time running.

*External*s are particular events requiring a certain time slot within the scheduling scenario, coming from the environment the system is working into. Dealing with the spaceflight application such an *activity* is just the eclipse time span. Fig. 1 gives an example for *sub-activity* and *external activity* domain dependent settlement.

The second important character defined is the r*esource: it* represents a system constraint, and allows the *activities* (and *sub-activities*) to be exploited.

Within the proposed architecture, the *resource* taxonomy is as follows: *Renewable- Aggregated-Integral- Punctual.*

Renewable: it is a resource that can be used by the ac*tivities* as far as it is completely saturated. During the system lifetime, there exist *activities*- devoted to its complete desaturation. In the spaceflight domain, a *renewable resource* is the on-board memory, and its *activity* - is the downlink. Particularly relevant, from the resource conflict solution point of view, is the possibility to split the current scheduling scenario into several *Resource-Windows* (RW); RWs comprehend the time slot between two consecutive *activities*- devoted to make a particular renewable resource again available (e.g. a link contact, as data can be damped, makes the on-board memory renewable resource again usable).

Solving conflicts involving such a *resource* starts from the most congested RW. As soon as the current *renewable resource* timeline is solved, the process restarts by analyzing the following *renewable resource* until the whole *renewable resource* domain is visited.

Aggregate: it is a *resource* that can be used but not necessarily depleted. A typical example, again in the spaceflight domain, is the angular momentum of the momentum wheels. That kind of *resource* does not present a predefined conflict configuration, as it is highly situation dependent.

Integral: it is a *resource* highly dependent not only on the current *activity* allocation but also on the time history of the whole *activity* allocation. It could be partially renewed by a particular *activity*-. The typical example is the energy coming from the on-board batteries.

Punctual: it is a resource that cannot be completely depleted but locally exceeded. Whenever an *activity* stops using a *punctual resource* amount, that amount is again completely available. Within the spaceflight application, the solar panel power production is considered as a *punctual resource.* As it clearly appears from the former definitions, the *resource* type and amount (whenever possible) used by each *activity* must be initialized in the KB; an example is given in Fig. 2 for the memory renewable resource.

According to the conflict solution tactics, the main character is represented by the *action*: a*ction* represents the tool to solve detected allocation inconsistency. An *action* cannot stand-alone but it is always related to an *activity*. Within the current work, the *action* universe is made of: {*to move-to reduce-to delete-to insert*}.

The core of the decision-making process is the selection of the best *activity-action* couple to obtain a consistent scheduling scenario with a good *fitness function* value.

Last but not least within the knowledge base to be instantiated once, off-line, temporal constraints, preferences and some parameters must be settled: A MxM matrix is settled – where M is the number of high-level activity types the system may accomplish-; constraints in terms of admissible time overlapping between activities are given in a 1 to 5 scale, from the softest to the hardest constraint level. Some parameters must be settled to contain the violation solutive loops in terms of maximum allowed actions to solve a conflict; in particular, activities can suffer a maximum number of actions before the whole scenario indexes are refreshed to update the decisional processes.

1.1.2. The DataBase

As already said, the DB represents the dynamic connection with the external environment in the sense that it collects, as soon as they are generated, new high-level activities to be allocated. Within an application to an Earth observation mission, the DB is updated whenever a ground contact uplinks a cluster of images asked by the scientists to be acquired by the payload. The particular formalism is related to the *interval of feasibility* and to the *opportunity windows* (OW) definition. Fig. 3 shows both a graphic representation and an extract of a possible DB. In particular, the OWs are strictly related to the *activity1* type: they represent – within the time interval in which the activity is required to be accomplished – the actual time slot the *activity* can be accomplished because of constraint satisfaction external to the scheduler tasks; just to clarify, a ground contact as well as an observation can be effectively accomplished whenever the space system is in view of the particular Earth zone: this is a constraint rising directly form the relative dynamic system-Earth and must be obviously satisfied to reach the imaging (contact) goal.

2. Time domain partitioning towards re-scheduling

Commonly, scheduling is based on a plan-then-execute tactics that means that operations devoted to schedule and to execute are temporally clearly distinct, as shown in Fig. 4.

In order to maintain the schedule as much as possible adaptive to the actual system status, the scheduling horizons the global mission is divided into must overlap each other to admit new activity allocation depending on the system status updating. A rule to activate the new scheduling must be defined, and it could be either an event-driven or a time-driven rule. In order to be more flexible and realistic an event-driven policy has been here preferred. A re-scheduling is required whenever either new events or deviations from a nominal behaviour are detected; if the re-scheduling activation is maintained time-driven, it may loose particular events determinant either for the health of the system or for the mission success. At the state of the art, activities to be responsible of new scheduling horizon activation are defined in the knowledge base as high-level boundary activities.

No FDIR features are implemented by the system, hence new scheduling horizons computation is required whenever new

activities are asked to the payload. In particular, within an Earth observation mission, boundary activities are, typically, links with the ground stations.
To assure scheduling horizon reciprocal overlapping not all the possible ground contact are considered as a bound, as shown in Fig. 5, but each can be considered as a call to the planner to start a new scheduling horizon. Whenever no bound activities are found or defined, the system autonomously define *activities* devoted to limit the current scheduling horizon, according to a mean horizon time span defined by the user.

3. KB and DB Preprocessing

The pre-processing has two basilar tasks: the first one is related to the *procedure* settlement, the second one to the activity classification generation. A *procedure* is the time slot devoted to accomplish an activity incremented, before and after, by all sub-activities required to achieve the imposed *attribute*s for each *activity* declared *SV*. The *procedure* is the results of all pre/post-requisites implicitly imposed by setting the KB. Fig. 5 shows the *procedure* related to the imaging high-level activity. *Procedure* generation runs once, off-line, before the scheduler start working on the dynamic system.
The *procedure* settlement has the aim to slim the consistent activity allocation as, in solving conflicts for each high-level *activity* a time slot is reserved already comprehensive of the connected low-level *sub-activities*. Moreover, as each current scheduling horizon has a part that has to be reconsidered because of re-scheduling properties, whenever the next scheduling is called, a fine low-level activity granular representation would require an excessive effort and time loss as *activities* have not to be considered definitely allocated yet. On the other side, the *procedure* scheme allows to obtain consistent scenario from all conflict source point of view (i.e. reservations, resources, pre/post-requisites).
During on-line operations, after a scheduling *activity* new calling, *activities* to be allocated within the current scheduling horizon have to be selected: activities not allocated yet, coming from the previous scheduling horizon and *procedures* already allocated in the previous time slot currently overlapping the new horizon must be activated; new activities to be allocated must be analyzed, according to a heuristic based on their *interval of feasibility*, their priority, their sensitivity to be reduced: depending on the match with the current scheduling horizon time span, they are classified as to-be-allocated or to be allocated-next, that means they are collected to be considered in the next scheduling scenario. As soon as the to-be-considered activity class is active, a first allocation is created.

4. The first Attempt scenario

The first attempt scenario is obtained by applying a constructive method; in particular, the backjumping technique, from the CS domain is applied [1]. The constraint net is restricted to time reservations, according to a scale from 1 to 5 to rank the hardness of that constraint. As the problem is over-constrained, a fitness function is settled in order to choose among different final inconsistent scenarios. The constraint net is restricted to the resource independent from the local activity allocation that is time, no matter of the others. The benefit stays in a partially solved first scenario. Another technique has also been applied, based on the allocated activity quality maximisation. A comparison between the two approaches, based on the running time, the loop number dedicated to solve conflicts and to the overall final allocation scenario showed the better behaviour of the CS approach. All allocations (possible OWs and OW reductions) are propagated in order to fill a matrix of possible time conflicts among activities: each activity start being allocated according to its maximum fitness value:

$$Fitness(act) = Priority(act)*Duration(act)*L1(act) \quad \text{eq. 2}$$

Where:
L1= ranking index for the *activity1* OW sorting obtained by a MISO FL block

One *activity* at a time is allocated: by visiting the propagated conflict matrix, whenever a conflict is detected, possible different allocations are considered both for the currently allocated *activity* and for those *activities* conflicting with the current one. Decisions about the *activity* to intervene on is taken by minimizing the loss of fitness either in differently allocating the current one or in re-allocating – by looking backwards -the former *activity* that allows to avoid the detected overlapping without generating any other violation. Whenever either no pre-allocated activity can be moved or reduced without creating new conflicts, or each possible current *activity* allocation is in conflict, the violation is maintained by allocating the current *activity* within the time window with the maximum fitness value. Unsolved conflicts will be solved later.
In particular, the CS technique is applied only for *activity1* type; *activity2*, thanks to their flexibility, are allocated according to a dedicated heuristic.

5. The Conflict Solver

As already said, the conflict solution is based on an iterative-repair approach: conflict taxonomy is created, according to the resource violation: *Renewable – Punctual – Integral*. The driving criterion stays in solving last the conflicts risen in the domain of those resources that are more time-allocation dependent: the *renewable* resource conflicts are solved first as conflicts among *activities* that need that resource type only depend on the RW limit satisfaction: whenever the RW resource global amount is satisfied, those *activities* can be moved (e.g. to solve conflict in other resource domains) within the given RW without creating anymore *renewable* resource conflicts; the *integral* resource conflicts are solved last as they are highly *activity* temporal allocation sensitive: it would be better to analyze those resource timelines once the scenario is almost solved but the *integral* resource conflicts to minimize loops dedicated to that resource consistency acquisition. **Fig. 6** shows the functional architecture of the conflict-solver [11].
Infinite solutive loops are avoided both by adapt the action feasibility meanwhile the scenario is filled, and by setting an upper limit for the solutive loop number. In any case, the presence of the *to-delete* action assures the system always converge to a solution.

As - apart from the dedicated input vectors - tactics are similar for each module, a general explanation is given.
Within the conflict analysis and solution, the following three main tasks must be accomplished: *to select* the worst conflict, *to decide what- to- do* to solve the conflict, *to decide on-which-activity-to-intervene* to solve the conflict. For each decision to be taken a metric is created to rank activities involved in the current conflict. The *what–to-do* decision is split into two waterfall levels: the first judge the validity of the *to-insert action* instead of intervene on currently conflicting ones; the second one is strictly related to the *which-activity-to intervene –on* decision, as it selects the couple *activity-action* to solve the detected conflict. The new *activities* to be inserted are always *activities*$^{-}$ type, strictly related to the current conflict resource domain: just to clarify, within the *renewable* resource domain, *activities* that can be inserted are further ground contacts, just to dump data and get the on-board memory again available; within the *integral* resource domain, particular attitude manoeuvres devoted to augment the incoming power from the solar panels help recharging the on board battery. *What-to-do* is managed as a MCDM problem: the current conflict solutive power, the overall planning scenario, the user preference are the three criteria currently considered. According to the WSM approach [12] the preferred tactic (*to insert* versus *to act*) is selected; the decisional matrix elements are obtained by dedicated MISO blocks implemented by applying the Fuzzy Logic Theory (FL)[13]; FL has been selected to obtain a highly adaptive decisional system as it permits to easily translate causal dependencies expressed in natural language into a mathematical formulation, good to automating the scheduling process; the inference engine that contains rules connecting the current system status to the control quantities (actions) well simulates the operators' expertise in solving rising conflicts during in-flight activities allocation.
As an example in the following the FL module – within the renewable resource conflict domain - devoted to the *what-to-do* ranking index definition for the current conflict solutive power criterion is given:

Inputs: number of that action needed to erase a conflict within the RW
Needed action to available action ratio
Outputs: L_{insert}, L_{move}, L_{reduce}, L_{delete}
Rules: 10:
e.g. 'the higher the action ratio, the higher the actions needed, the lower the preference index is'

The couple *action-activity* to intervene on to solve the current conflict is detected by applying the Analytic Hierarchical Process (AHP) particularly fitting for complex decisional problems [14]: within a given framework (i.e. a conflicting *activity*) alternatives (i.e. solutive *actions*) are easier ranked by comparing them in pair instead of directly assigning an absolute cardinality; to this end, a matrix of pairwise comparison is generated to compare actions in pair, according to each activity belonging to a pre-selected set of conflicting *activities*: by computing the eigenvector of the maximum eigenvalue of that nxn j-th matrix (n= no.of admitted *actions*, $j \in [1\ m]$, m=no.of *activities* in the current conflict) an absolute *action* ranking is obtained, within each *activity* domain: by selecting the highest index of the matrix the columns of which are the computed *activity* eigenvectors, the best couple *activity-action* is obtained. *Action-activity* solutive couples are applied following the decreasing index order a predefined number of times, before the whole scenario is refreshed.
In order to better understand this decisional level, an example is given for the move/reduce *action* comparison for a *procedure* involved in a *renewable* resource conflict.
Within the overall high-level conflict solver, the already mentioned *procedures* are considered. As soon as the high-level scheduling scenario is consistent, the detailed level is activated. That former level is dedicated to the *sub-activity* allocation refinement: *SV attribute* transitions are checked and optimized by erasing, for example, unuseful transitions.

6. Simulation Results

Simulations run with two different tactics for updating the DB. The first one, particularly focused on the system performance evaluations, generates requested activities randomly, while the second one is obtained by interfacing the scheduler with the Satellite ToolKit application, to propagate the real s/c dynamics. For lack of space, only results obtained for a generic Earth observation mission on a sun-synchronous orbit are shown. A camera represents the p/l and high-level *activities* are, firstly, scientists' observation requests.
Fig.8a shows the first attempt allocation scenario – with the on-board resource trend - obtained by applying the backjumping CS algorithm to the first scheduling window that last till the second link activity (red box at $1^{d}22^{h}56.6^{m}$ from the beginning). As expected, as the problem is over-constrained some violations remains, as highlighted in the zoomed box: take_photos *activities* happen during eclipse (the mutual overlapping constraint is as hard as possible, no sun light=no photo); resource different from reservation are violated.
Fig.8b is the final consistent allocation: in particular it is possible to notice that no more take-photo *activities* falls into eclipse time; a lot of recharge mode2 *activities* devoted to maintain the energy level within the threshold are inserted: recharge is obtained by switching off redundant components. Starting from 25 scientists'requests 4 are definitely cancelled, 6 reduced; 26 'move' have been applied, and 10 loops to achieve the renewable resource consistency 36 loops to achieve the punctual resource consistency have been required (see Fig. 6**).**
Starting from a fitness value of 81807 – representing the best allocation choice for the observation windows no matter of inconsistency - the final scheduling presents a fitness of 76588.37. The fitness decrease is basically due to the four activity deletion. The fitness is also used in a learning module currently under development.
The activity plan and the state timelines for the first horizon are not given for lack of space. *Procedures* are here split in the correspondent sequence of pre/post-requisites, in order to assure a low-level consistent scheduling scenario. Activities and attributes of the involved SVs are correctly allocated in time. The next scheduling window allocation is not presented

for lack of space: 10 further observations are uplinked; the scheduling window lasts $1^d\ 20^h\ 26{,}7^m$, starting at $22^h13.3^m$ from the beginning; 1 activity is cancelled and again 6 reduced; to solve resource violations a new ground contact is inserted to dump data and get the on-board memory again available; attitude manoeuvres devoted to augment the incoming power are inserted, as before. The final fitness value is 80397.06, starting from a 81587,87 value.

7. Conclusions

The paper showed a possible approach to the re-scheduling of space system activities during in-flight operations: the proposed method based on an iterative repair approach with the help, for the first allocation setting of a CS algorithm. In particular, the novelty stays in the method implemented to detect, select and solve violations: starting from a MCDM approach, a decisional tree based on the conflict type taxonomy is settled: an AHP is used to balance multiple attributes-multiple tactics to solve violations; but, departing from the classic AHP, the matrix of parwise comparison is filled by devoted FL blocks: that way of dealing allows to capture the ground operators'expertise – based on qualitative relationships- and to automate the decisional process. Simulations run both with a random activity generator and with a connection to an orbital propagator. Results are quite encouraging as activity consistent allocation is always accomplished with both a good fitness in terms of product return (payload activity quality), resource management, running time (O(10^1s)).
Work is still on going to implement a learning module to refine, during the scheduling sessions weights devoted to lead the decisional modules.

8. References

1.]R. Bartak, "Theory and Practice of Constraint Propagation", Proceedings of CPDC Workshop, 2001
2. A. Fukunaga, G. Rabideau, et al., "Towards an application framework for automated planning and scheduling", Proceedings of i-SAIRAS, 1997
3. M. Zweben et al, "Scheduling and rescheduling with iterative repair", Intelligent Scheduling, Morgan Kaufmann editor, 1994
4. G. Rabideau, S. Chien, et al., "Interactive, repair-based planning and scheduling for shuttle payload operations", Proceedings in IEEE Aerospace Conference, 1997
5. G.Rabideau, R. Knight, S. Chien, A. Fukunaga, A. Govindjee, "Iterative Repair Planning for Spacecraft Operations Using the Aspen System", in Proceedings of the i-SAIRAS Congress, Nooordwijk-The Netherlands, June 1999
6.]R. Bartak, "Theory and Practice of Constraint Propagation", Proceedings of CPDC Workshop, 2001
7. A. Fukunaga, G. Rabideau, et al., "Towards an application framework for automated planning and scheduling", Proceedings of i-SAIRAS, 1997
8. M. Zweben et al, "Scheduling and rescheduling with iterative repair", Intelligent Scheduling, Morgan Kaufmann editor, 1994
9. G. Rabideau, S. Chien, et al., "Interactive, repair-based planning and scheduling for shuttle payload operations", Proceedings in IEEE Aerospace Conference, 1997
10. G.Rabideau, R. Knight, S. Chien, A. Fukunaga, A. Govindjee, "Iterative Repair Planning for Spacecraft Operations Using the Aspen System", in Proceedings of the i-SAIRAS Congress, Nooordwijk-The Netherlands, June 1999
11. S. Chien, R. Knight et al., "Using iterative repair to improve the responsiveness of planning and scheduling", Proceedings of the 5th International conference on Artificial Intelligence Planning and Scheduling, 2000
12. N. Muscettola, P. Nayak et al., "Remote Agent: To boldly go where no AI system has gone before", Proceedings in IJCAI Congress, 1997
13. N. Muscettola," HSTS: integrating planning and scheduling", Intelligent Scheduling, Morgan Kaufmann editor, 1994
14. S. Chien, G. Rabideau, et al., "ASPEN – Automated Planning and Scheduling for Space Mission Operations", Proceedings of Space Operations, 2000
15. B. Smith, R. Sherwood, et al., Representing spacecraft mission planning knowledge in ASPEN , AIPS, Workshop on knowledge engineering and acquisition for planning, 1998
16. M. Lavagna, A.E.Finzi "On-board dynamic scheduling with a Fuzzy Logic Inference Engine", Proceedings of i-SAIRAS, Montreal, 2001
17. E. Triantaphyllou, B. Shu, et al., "Multi-Criteria Decision Making: an Operations Research approach", Encyclopedia of electrical engineering, John Wiley and Sons, pagg 175-186, 1998
18. E. Triantaphyllou, S. Mann, "Using the AHP for decision making in engineering applications: some challenges", International journal of industrial engineering: applications and practice, vol. 2, n° 1, pagg 35-44, 1995
19. G. J. Klir, B. Yuan, "Fuzzy Sets and Fuzzy Logic", Prentice Hall, 1995

[activity]
name: take-photo
OW storage: 0
priority:4
duration:230
resource:memory,66.5,
energy, NaN, power 18
State variables: camera, on,
attitude, par, vibration, low
action permission: [5,1,,1]
reduction:1.5e-1
bound:0
periodic:0

[state]
name: camera
attribute: on, off, warming, stand_by
default: off
link: stand_by
transition:[1, 0.5 0 0.5; 0 1 1 0; 0.5 0 1 0;0.5 0.5 0 1]
type:2

[state] name: attitude	attribute: par, SAA_up, SAA_down, nadir, rotational, sun default: nadir link: nothing	transition:[0 0 0 0 1 0; 1 0 0 1 0 ; 0 0 1 0 1 0 ; 0 0 0 1 1 0; 1 1 1 1 1 1; 0 0 0 0 1 1] type:5

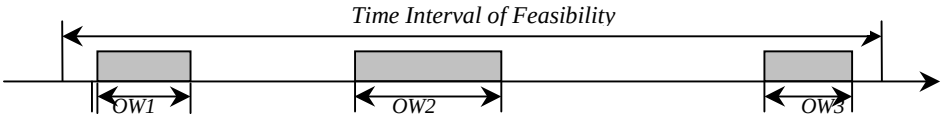
Fig. 1 Examples for the activity and state definitions within the static KB

[sub-activity]
name: slew
duration:60
resource: energy, NaN, Power 17
State variables: attitude, rotational, vibration, high, CPU, busy
Subroutine: sbrtne_slew
Estimate duration: [30,5,60]
Calling state: attitude

[external]
name: eclipse
type:propagated

[resource]
name: memory
type: renewable
range: [0 1000]
do not exceed: [0 1]
sensible to reduce: 0

Fig. 2 Examples for the sub-activity, external activity and resource definitions



Imaging:[’10 Aug 2002 04:15:00’’11 Aug 2002 0:30:00’] Target, pos,[-82.3°; 10.5°]

Imaging:[’10 Aug 2002 01:00:00’’10 Aug 2002 9:30:00’] Target, pos,[-42.3°; 110.8°]
link:[’11 Aug 2002 06:00:00’’11 Aug 2002 08:45:00’]Fixed

Fig. 3 OW definitions and DB extract

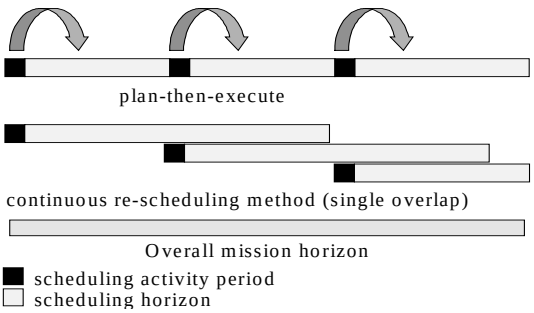


Fig. 5 Scheduling horizons in a re-scheduling framework

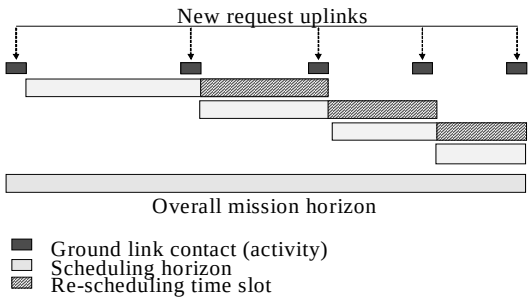


Fig. 4 re-scheduling window activation

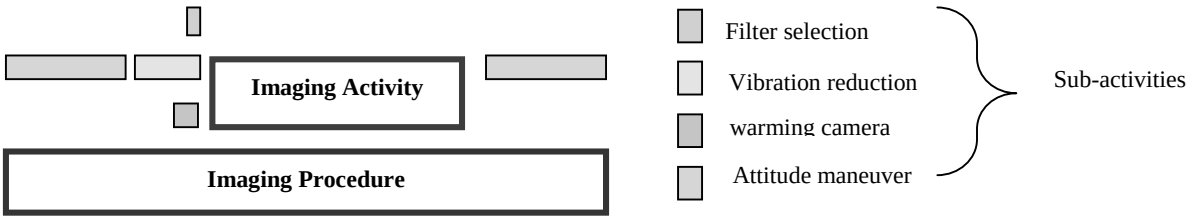


Fig. 5 Procedure Generation for the high –level *activity* ‘imaging’

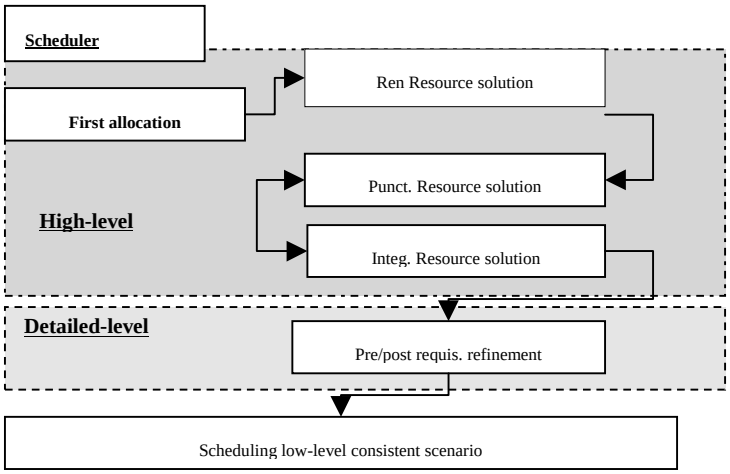
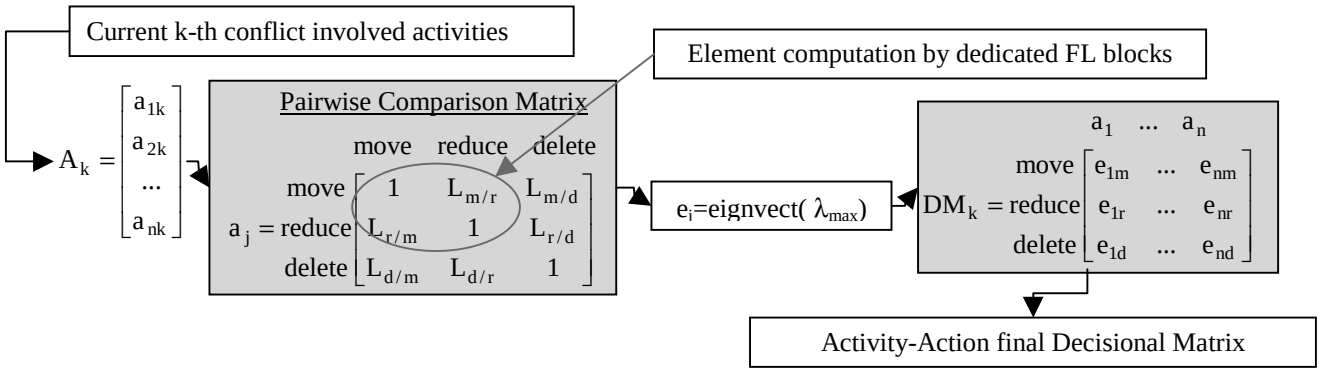
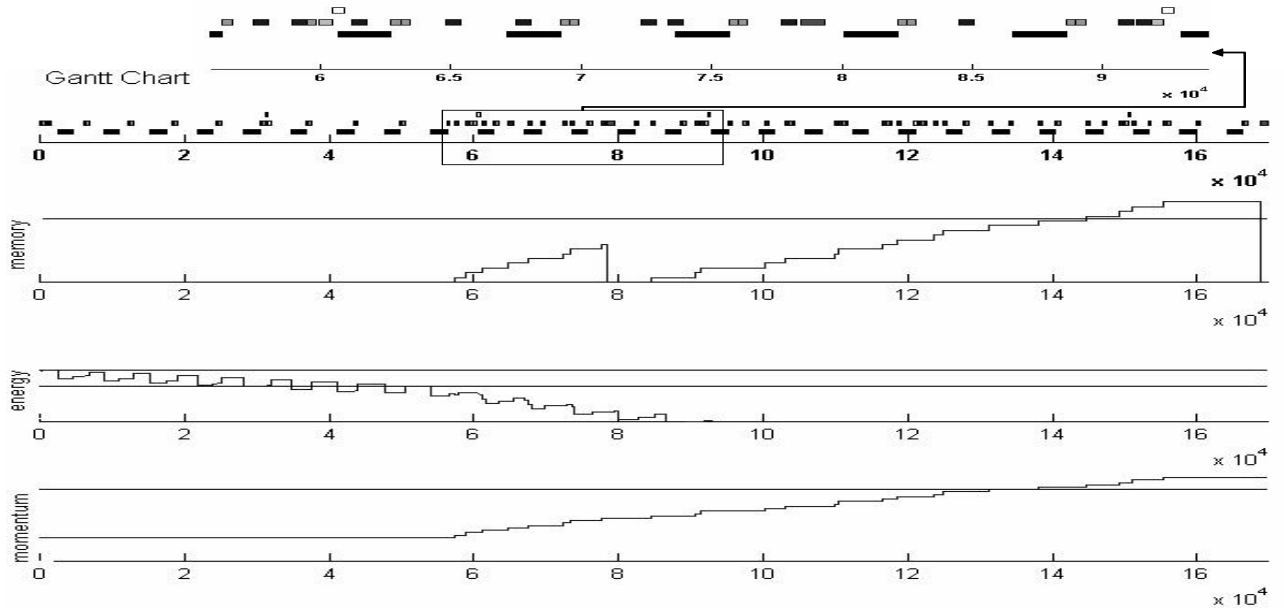


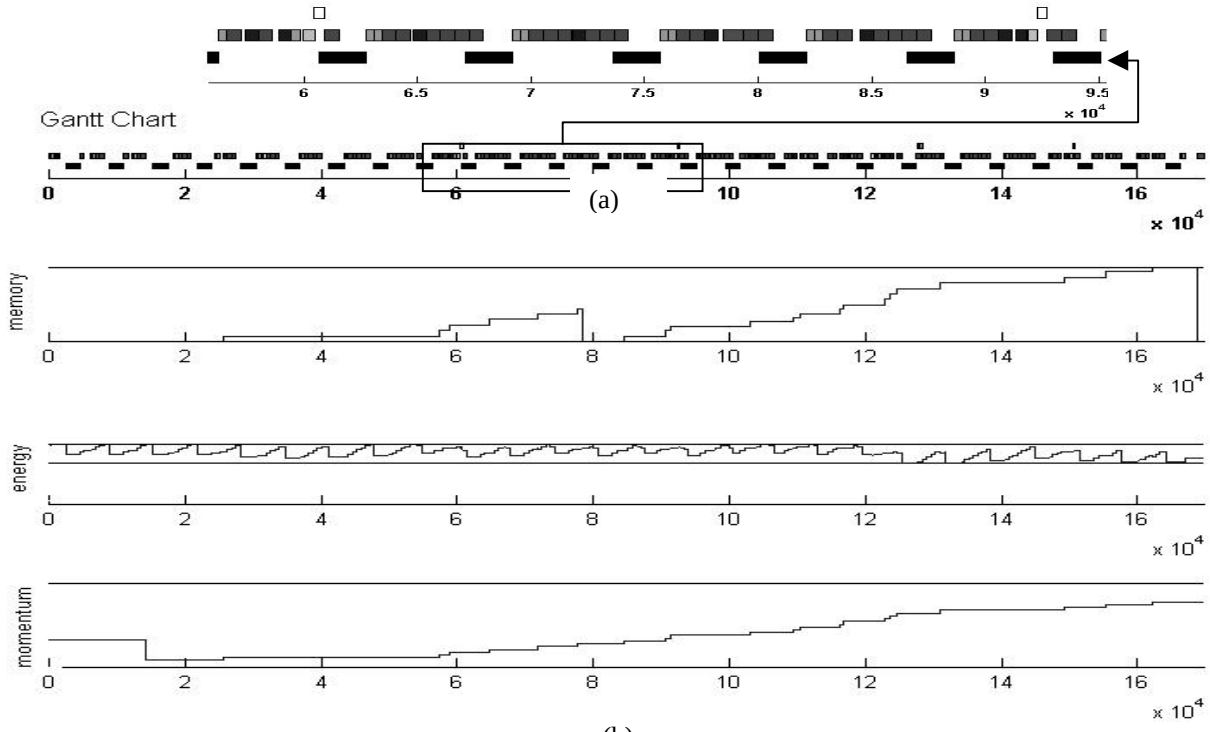
Fig. 6 Conflict solver functional architecture



$L_{m/r}$ inputs: -ratio between resource gain in applying the 'move' instead of the 'reduce'
-ratio between the fitness loss in applying the 'move' instead of the 'reduce'
-overlapping time for the current activity- ratio between no. of remaining actions to be done within each action domain
- L_1 activity quality index
Outputs: $L_{m/r}$ Rules: 19

Fig. 7 activity-action couple selection functional diagram and Lm/r dedicated FL for the j-th activity within the k-th RR conflict





(a)

(b)

Fig. 8 First scheduling horizon: first attempt (a) and final consistent allocation (b) and resource timelines