the Discrete Commanding of Aerospace Equipment

Neil Yorke-Smith and Christophe Guettier
IC–Parc, Imperial College London, SW7 2AZ, U.K.
{cgue,nys}@icparc.ic.ac.uk

***Abstract*— An increasing requirement for satellites, space probes and (unmanned) aircraft is that they exhibit robust behaviour without direct human intervention. Autonomous operation is required in spite of incomplete knowledge of an uncertain environment. In particular, embedded equipment that processes sensing data must consider uncertain input parameters while managing its own activities. We show how uncertainty may be addressed in constraint-based planning and scheduling functions for aerospace equipment, contrasting with some current practice in Integrated Modular Avionic (IMA) design. We produce a conditional plan that takes account of foreseeable contingencies, so guaranteeing system behaviour in the worst case. Executing a branch of the plan corresponds to synthesising a deterministic finite state automaton capable of discrete event commanding of an avionic sub-system. Experimental results show the feasibility of the approach for realistic aerospace equipment.**

I. Introduction

From the first days of space missions, manned and unmanned, the need to manage uncertainty has been crucial. Uncertainty arises for the same reasons as on Earth — knowledge is incomplete, the environment is changing, the future is unobserved — but its impact is only magnified. When designing and planning for such missions, we cannot avoid the inherent unknowability of what might be encountered.

If future space and aeronautic systems are to achieve more complex missions with less human intervention, a highly automated mission management process will be required [1]. The system must continuously operate in a changing and perhaps ill-known environment, use complicated equipment and instruments, and simultaneously fulfil mission goals and satisfying system requirements (such as timeliness or safety). In space, examples of these systems are probes and planetary orbiting formations, as demonstrated by the Deep Space 1 Remote Agent Experiment [2]. In the aeronautic domain, representative examples are Unmanned Aerial Vehicles for both military and civilian purposes [3].

Current Integrated Modular Avionic (IMA) approaches to behaviour control and planning use finite-state deterministic reactive automata, by means of a formal specification [4]. While this approach does not differentiate planning from control or sensing functions, it does necessitate a perfect knowledge of the environment, and leads to a rigid specification of the system behaviour. Hence these systems are unable to handle uncertainty in an adaptive way.

Our approach models uncertainty within a constraint-based planning framework, in order to improve the robustness of the system behaviour and to reduce operator interventions [5]. We introduce the use of non-deterministic constraint-based automata, and represent each system component by an automaton model. According to the environmental uncertainty, a dedicated automaton is synthesised automatically from the model by a constraint solver. The synthesised automaton corresponds to a branch of a conditional plan. This plan is prepared offline, and the appropriate branch (automaton) is selected online by the on-board system with little overhead.

The Constraint Programming (CP) language used for modelling and solving enables the composition of our planning formulation with other models of the system, such as resource consumption or scheduling constraints. The result, compared to traditional IMA techniques, is a more modular and compositional problem representation, and thus a better representation of global system behaviour. Further, the offline plan generation is complementary to purely reactive control functions. On one hand, the generated plans can be a reference trajectory for an online controller [6], or be part of a cost function for model-predictive control [7]. On the other hand, the plans can parameterise a feedback policy for closed loop control.

II. Modelling with Constraint-Based Automata

We represent component activity over a fixed discrete horizon, using a constraint-based non-deterministic automaton. An example is seen in Fig. 1. This approach has been investigated for several different mission planning domains [3], [8]. For each discrete state of the automaton, we associate a functional approximation that models a physical law (e.g. speed, temperature). Thus, a state in the automaton models the continuous behaviour of the interaction between the component and its environment. Transitions between states in the automaton model abrupt changes in behaviour. In practice, we think of a state of the automaton as corresponding to a mode of operation of the avionic component. From one mode, the modes that could follow in a feasible sequence are specified by possible transitions in the automaton. A transition between states of the automaton is triggered by a composition of events. Events can be of two kinds:

- *Contingent events* are occurrences outside the direct control of the agent. In the model, contingent events are represented by constraints based on physical parameters, formulated for each time point of the horizon.
- *Controllable commands*, in contrast, are events under the direct control of the agent. In the model, controllable

Proceedings of 2003 IEEE International Symposium on Intelligent Control, Houston, TX, October 2003.
Some minor oversights have been corrected in this version.

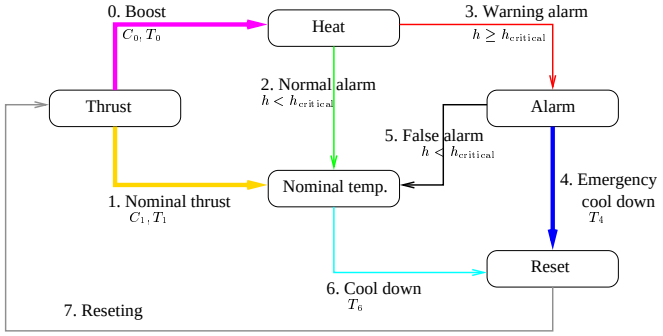


Fig. 1. Discrete automaton representing the behaviour of a thruster sub-system. States corresponds to the edges and transitions to the vertices. The temperature increases will differ in the two thrusting states (*Boost*, *Nominal thrust*), and in the two cooling states. The C_i variables are commands, the T_i variables are state time-outs.

commands correspond to decision variables. These variables are distinguished as either *time-out variables*, which model the command for interrupting a state, or *choice-point variables*, which model the selection of one of several alternative transitions.

In Fig. 1, the transition into state 3 (*Warning alarm*) is contingent on the temperature being above a critical value; whereas the transition from state 7 (*Reseting*) into boosted or nominal thrust is governed by a decision variable.

In the constraint model that forms the automaton, we distinguish between two types of variables: uncontrollable *parameters*, and controllable *decision variables* [9]. The value of a parameter is imposed by the environment; moreover, our knowledge of its value might be incomplete. It is through parameters that we model uncertainty, which we represent as intervals of discrete values.

Planning consists of defining consistent sequences of states in order to reach a given target state. This corresponds to the equipment changing modes of operation, in a feasible way, to reach a target mode. The target mode is specified by mission and operational goals. The sequence (plan) must satisfy the model-based constraints and, possibly, optimise a given performance function. It is natural to use automata to represent the plan, because: first, this is how low-level component IMA are modelled; and second, automata conveniently translate into real-life controllers for the command of equipment.

A. Objective for Robust Planning

The planning problem we address has non-deterministic actions due to contingent events, fully observable states, and ill-known data (the parameters). The uncertainty brings with it the question of what we seek as an outcome. In the first instance, we aim to produce robust plans: plans useful in every realisation of anticipated uncertainty.[1] However, in conjunction with this robust behaviour there may be other objectives, for instance a minimum plan quality; and other requirements, for instance a limit on online computation time.

[1]If unanticipated contingent events occur, or if parameters take realised values outside their domains, a reactive response is necessary. See [10].

In both domains, space and aeronautic, the system requirements are for specified behaviour in the worst case. The autonomous system must guarantee a certain performance, no matter what values the parameters take. This means that seeking one plan, however optimality is measured, is inappropriate for our problem, unless the plan will hold under all *realisations* (i.e. all anticipated scenarios). Our empirical studies, reported below, strongly suggest this is not the case. Therefore, in order to improve existing IMA properties (e.g. reliability, safety), we propose to generate a covering set of feasible plans.

Precisely, let $\mathcal{P}$ be the problem, and let $\mathcal{P}_r$ denote $\mathcal{P}$ under realisation r. We find a set of plans S, such that for every feasible realisation r of $\mathcal{P}$, at least one element of S is a viable plan for $\mathcal{P}_r$. Ideally, of all the sets that cover every realisation in this way (the *complete decisions*), S should have minimal cardinality, i.e. be the smallest covering set.

The outcome of our planning function will thus be a complete conditional plan. This corresponds to (1) synthesising a set of deterministic automata and a timed state sequence for each, and (2) building a discrimination tree to choose which automaton to apply in which realisation. Due to the composition of the constraints, this planning entails solving several related NP-hard subproblems.

For comparison, we will also look at the plan of maximal robustness. That is, the one plan that (simultaneously) covers as many realisations as possible. While a single plan is attractive, such a *universal decision* will not exist in general.

B. Constraint-Based Automata

We now present the mathematical modelling of constraint-based automata, extended to handle environment uncertainty.

Let $H \in \mathbb{N}$ be the finite planning horizon. Any $t \in [0, H]$ corresponds to a discrete time event. Let n be the number of states of the automaton. Each state $\sigma_i,\ i \in [0, n-1]$ is active at instant t if the predicate $\sigma_i(t)$ holds true. By convention, σ_0 is the initial state and $\sigma_0(0)$ is always true.

1) Transitions, events and decision variables: A transition specification $\delta(\sigma_i, \sigma_j),\ j \neq i$, models a change of behaviour between distinct states σ_i and σ_j. The transition $\delta_t(\sigma_i, \sigma_j)$ is triggered at time t if the state σ_i is active at $t-1$ and the event associated to the destination state σ_j holds true at t:

$$\forall i, j, \forall t:\ E_j(t) \wedge \sigma_i(t-1) \Rightarrow \delta_t(\sigma_i, \sigma_j) \qquad (1)$$

An event $E_j(t) \in \{\bot, \top\}$ constrains the activation of a given state. It can be a contingent event, raised due to the component's environment, or a controllable event, or a conjunction of both. The valuation of contingent events depends on physical parameters arising from the environment. In contrast, the valuation of controllable events is decided by the associated decision variables: the commands (C_i) and time-outs (T_i); their values are assigned by the solving process. For the latter time-out variables we impose that the state becomes inactive after a given amount of time:

$$\forall i, \forall t:\ \neg\sigma_i(t-1) \wedge \sigma_i(t) \Rightarrow \neg\sigma_i(t+T_i) \qquad (2)$$

2) Consistent action sequences: Traditional techniques adopted by engineers are based on deterministic reactive automata: in any given state, an automaton can reach exactly one state. These automata cannot be adapted easily to environment changes. Hence we propose to widen the approach by lifting the deterministic assumption, such that multiple transitions can be specified from a given state. However, according to a given environment realisation, the selection of a unique destination state is ensured by the constraint:

$$\begin{aligned} \forall i, \forall t : \sigma_i(t) \wedge \neg\sigma_i(t+1) \Rightarrow \\ \exists! j \text{ s.t. } \sigma_j(t+1) \wedge \delta_t(\sigma_i, \sigma_j) \end{aligned} \tag{3}$$

By (3), exactly one destination state can be active after state σ_i. Further, since we do not consider multiple parallel activities, we have the constraint (4) to ensure only one state can be active at a time:

$$\forall t, \exists! i \text{ s.t. } \sigma_i(t) \tag{4}$$

It follows that the sequence of active states corresponds to the behaviour of the component, and the sequence of intermediate commands and events represent actions.

3) Environmental constraints: Feasibility constraints, entailed by environmental dynamics (e.g. speed, temperature) and cumulative resources (e.g. ergol, power supply), are specified as follows. We assume that in a given state, a physical parameter evolves in a regular way, such that it can be approximated using a cumulative function. This is similar to a resource utilisation formulation in which the resource level at t is a function of the level at $t-1$ alone.

Let $p(t) : [0, H] \rightarrow \mathbb{N}$ be such a physical parameter. A problem-dependent recursion describes the evolution of $p(t)$ in terms of a dynamic function f_i for a given state σ_i:

$$\forall i, \forall t : \sigma_i(t) \Rightarrow p(t) = f_i(p_i(t-1)) \tag{5}$$

Equation (5) approximates the dynamics of $p(t)$. The initial conditions of the system are represented by $p(0)$ and are arbitrarily constant. Contingent events are then defined using physical parameters and a constraint c_i:

$$\forall i, \forall t : E_i(t) \Leftrightarrow c_i(p_i(t)) \tag{6}$$

Both (5) and (6) are problem-dependent. The structure, and hence complexity, of these constraints can lead to very different representations, and hence different solving performance.

4) Parameter uncertainty in the constraint model: The constraint-based automaton described in this section forms a constraint satisfaction problem (CSP) with parameters. Recall that a classical CSP over finite domains is a tuple $\langle \mathcal{V}, \mathcal{D}, \mathcal{C} \rangle$, where $\mathcal{V}$ is a finite set of variables, $\mathcal{D}$ is the set of corresponding domains, and $\mathcal{C}$ is a finite set of constraints. A solution is a complete consistent value assignment. A *mixed CSP* [11] is a tuple $\langle \Lambda, \mathcal{V}, \mathcal{U}, \mathcal{D}, \mathcal{C} \rangle$, where Λ is a finite set of parameters and $\mathcal{U}$ is the set of corresponding domains.

C. Example: Planning the Commands of a Thruster

As a running example for the paper, we introduce a representative planning problem involving a thruster sub-system: the *Constrained Thruster Control Problem* (CTCP). The automaton of Fig. 1 is a simplified version of such a system. The controllable events consist of a set of commands that periodically trigger nominal or boosted stages of thrust for a variable period of time. These different stages can be interrupted using time-out decision variables. Contingent events are hardware alarms that change the equipment mode, according to various temperature limits. Cooling modes, although triggered by contingent events, can be interrupted by a time-out variable. The goal is to achieve a certain thrust performance in a given time window, while maintaining the internal temperature within given limits. Generating a plan for one realisation consists of instantiating decision variables that correspond to commands and time-outs, while satisfying the temperature and thrust requirements.

An instance of the CTCP is in atmospheric entry of a probe. For illustration, we focus on the temperature, neglecting the other parameters. We represent uncertainty in parameter values by discrete, non-stochastic intervals.

For our example CTCP, the set of states and their associated entry events can be summarised as follows:

state	controllable	contingent
0. Boost	T_0, C_0	
1. Nominal thrust	T_1, C_1	
2. Normal boost alarm		¬ warning
3. Warning alarm		warning
4. Emergency cool-down	T_4	¬ false alarm
5. False alarm		false alarm
6. Cool down	T_6	
7. Reseting		

From its graphical representation in Fig. 1, we see that the automaton involves a cycle. A preprocessing function unfolds the automaton states along the horizon; as the number of cycles increases, so does the horizon.

Transitions between states are characterised as follows:

transition	predicate	deterministic
Thrust	$\delta(\sigma_7, \sigma_1), \delta(\sigma_7, \sigma_0)$	no
Heat	$\delta(\sigma_0, \sigma_3), \delta(\sigma_0, \sigma_2)$	yes
Nominal temp.	$\delta(\sigma_2, \sigma_6), \delta(\sigma_1, \sigma_6), \delta(\sigma_5, \sigma_6)$	yes
Alarm	$\delta(\sigma_3, \sigma_4), \delta(\sigma_3, \sigma_5)$	yes
Reset	$\delta(\sigma_4, \sigma_7), \delta(\sigma_6, \sigma_7)$	no

The physical parameters, temperature $h(t)$, and thrust performance $b(t)$, are approximated using families of linear recursions. Each state i is associated with such a function. Thus for this problem, (6) is:

$$\forall i, \forall t : \begin{cases} h(t) = h(t-1) + K_i \\ b(t) = b(t-1) + B_i \end{cases} \tag{7}$$

Constants $h(0)$ and $b(0)$ are arbitrarily set to known steady values, and the performance functions are bounded: $h(t) \in [0, h_{\max}]$ and $b(t) \in [0, b_{\max}]$, where the constant $b_{\max}$ is the maximal thruster performance.

More complex constraints, including non-linear, continuous and disjunctive constraints, can be formulated in the same way. For the purpose of clarity, we restrict the temperature and performance recursions in this paper to be linear; even so, the global problem in the example is non-linear due to the presence of choice-points and the event formulations.

Now we can state the constraints corresponding to events *warning* and *false alarm*:

$$\forall t \in [0, H] : warning(t) \quad \Leftrightarrow h(t) \geq h_{\text{critical}} \tag{8}$$

$$\forall t \in [0, H] : false\ alarm(t) \Leftrightarrow h(t) < h_{\text{critical}} \tag{9}$$

where h_{critical} is a threshold value lower than $h_{\max}$. Finally, the thrust must satisfy a minimum performance $B_{\min}$:

$$\sum_{i=1}^{H} B_i \geq B_{\min} \tag{10}$$

III. Solving Algorithms

In this section, we outline algorithms to solve for the two planning objectives: mcs, finding the minimal covering set of plans; and mrp: finding the single plan of maximal robustness. In the next section we report the experimental results when applied to the Constrained Thruster Control Problem.

Declaratively, the semantics of our approach are described by an operator acting on the uncertain problem to give an element of an algebraic structure. This structure is the subsets of the set of all possible plans (i.e. every plan that is feasible for at least one realisation), which is a boolean algebra under subset inclusion. The set of plans we derive operationally, using the solving methods below, is an instance of the certainty closure approach to data uncertainty in CP [12].

We say that a realisation (scenario) is *feasible* if the constraints of the problem permit it to ever occur (otherwise *infeasible*). A feasible realisation r is *good* if some solution s exists for the decision variables, given that the parameters have taken their values under r (otherwise *bad*); then s is said to *cover* r. By *robust*, we mean that a solution s_1 covers more realisations than a solution s_2.

1) mcs: Minimal Covering Set of Plans: We give two algorithms for the task of finding a set of plans that cover every realisation. Neither guarantees the set of minimum cardinality; the trade-off is that a smaller covering set yields a more compact conditional plan, but might take more time to find.

The first algorithm, `heuristic`, is a naive method: it considers every realisation. The idea is to first compute a heuristic plan $\hat{s}$ that is likely to cover many realisations. For a realisation r, if $\hat{s}$ is feasible, we are done; if not feasible, we compute from scratch a feasible plan for $\mathcal{P}_r$.

A more efficient approach is, for each plan s computed, to remove from future consideration all realisations covered by s. This is the underlying idea of the decomposition algorithm `decomp`, given as Algorithm 1. It is based on the conditional decision method for mixed CSPs with full observability [11]. Central to the method are so-called *environments* — set of realisations — and their judicious decomposition. The result is an anytime algorithm that computes successively closer approximations to a complete decision. If the algorithm is allowed to finish without interruption, it returns a complete conditional decision.

Algorithm 1 Decomposition for covering set of plans

```
B ← ∅ {bad realisations}
D ← ∅ {decision–environment pairs}
E ← U_1 × ··· × U_p {environments still to be covered}
repeat
  Choose an environment e from E
  let 𝓔 be constraints that enforce e
  let P be the CSP ⟨Λ ∪ 𝒱, 𝒰 ∪ 𝒟, 𝒞 ∪ 𝓔⟩
  if P is consistent then
    let s be a solution of P
    let d be s projected onto the variables 𝒱
    R ← covers(d) {realisations covered by d}
    Add the pair d–R to D
    E ← ⋃_{e'∈E} decompose(e', R)
  else
    {all realisations in e infeasible}
    Add e to B
  end if
until E = ∅ {all feasible realisations covered}
return(B, D)
```

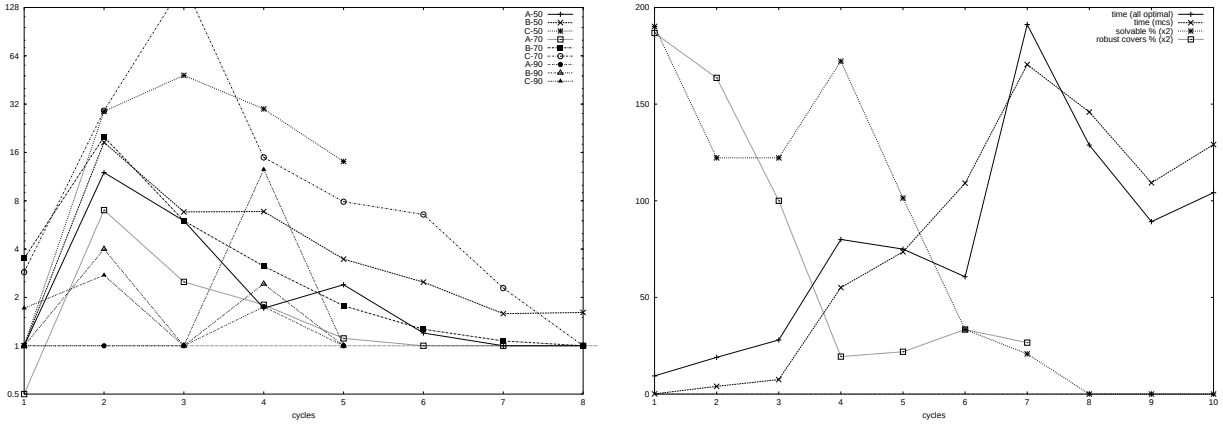
2) mrp: Maximal Robustness Plan: We give two exact algorithms for the task of finding the plan that is feasible for the maximal number of realisations. As before, the first is a naive method. It considers every realisation, computing all feasible solutions for each. When done, it chooses of all the solutions the plan that occurs most frequently. Unsurprisingly, algorithm `naive` is prohibitive in both time and space.

A more efficient approach is to search using branch-and-bound over the space of feasible plans. `b+b`, the second algorithm for mrp, therefore extends the CSP inference technique of *forward checking* within a branch-and-bound search tree, where the value of each leaf node is number of realisations it covers. This type of algorithm is familiar in CP, and in this context is a non-probabilistic version of that for no observability probabilistic CSP [9].

IV. Experimental Results

We modelled the example problem, and implemented the solving methods, using the ECL^{i}PS^e system [13]. We consider the CTCP with three classes of magnitude of uncertainty, and with three different performance requirements. The former, denoted A–C, have intervals of modest, intermediate, and broad width respectively. The latter, denoted by a performance objective as a percentage of the maximum possible, are set at 50, 70 and 90% (contrast with (10)). For each of the nine problems that result, we solve for a number of cycles in $1 \ldots 10$. Many of the instances are infeasible for higher time horizons under any realisation, indicating that if we attempt to thrust for so long, the probe will unavoidably overheat.

1) mcs: Minimal Covering Set of Plans: For the heuristic method, we chose as the heuristic solution $\hat{s}$ that corresponding to the realisation where every temperature increment is maximal. The intuition is that the plan for the worst case may tend to be robust for other cases. Compared with other simple choices, we found this heuristic performed best.



(a) Plan quality for mcs: ratio `heuristic` to `decomp`

(b) Problem hardness for class B, 70%

Fig. 2. Algorithms compared on the constrained thruster control problem.

In Fig. 2(a) we plot the ratio of plan sizes of `heuristic` over `decomp`; thus the greater the value, the greater the advantage to the decomposition algorithm. Note that the vertical axis is on a logarithmic scale. The greatest difference between the methods is seen for each instance to be when the instance is at its hardest; beyond this critical point, the instance tends towards infeasibility, and the ratio of plan sizes in Fig. 2(a) tends towards unity. In some more tightly constrained instances, `decomp` can take longer; but its running time is more consistent than the heuristic method between instances. Moreover, if we measure quality of solution by the size of the set produced, then `decomp` consistently yields better quality solutions across other problem instances.

2) mrp: Maximal Robustness Plan: Fig. 2(b) shows, as the line marked with boxes, the number of realisations covered by the most robust plan. We observe that the most robust plan covers a large majority of the plans when the horizon is short. As the number of cycles increases to 4 or 5, however, and the plan space becomes larger, the percentage of realisations covered drops sharply. This effect is more pronounced as the amount of uncertainty, and so the number of realisations, increases. As anticipated, `b+b` easily outperforms `naive`, which struggles for the harder instances.

3) Discussion: In Fig. 2(b), we also plot the sum of the time (in seconds) to calculate the optimal plan for each realisation, and the time for mcs by `decomp`, together with the percentage of feasible realisations and the percentage of these covered by the most robust plan (the latter two datasets scaled by two). Our results indicate that the hardness of the CTCP jumps, before declining again once all realisations become infeasible: observe the peak in difficulty for 7 cycles; infeasibility occurs at 8 cycles. The time and percentage of feasible realisations appear to be inversely related.

Secondly, Fig. 2(b) shows an inverse relationship between problem difficulty and plan robustness. The maximally robust plan in general covers a small percentage of the feasible realisations, at least for non-trivial cases. Similar trends are seen across all the CTCP classes. This means that there are many potential realisations for which the plan manager would have to take remedial action online.

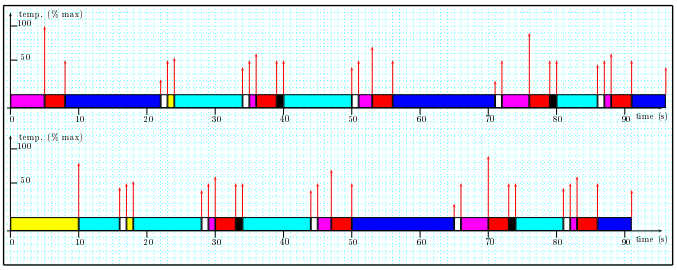


Fig. 3. Example of solutions for two close realisations. The horizontal bars denote the state of the synthesised automata, and the vertical lines denote the temperature at each transition. Observe that the first thrusting stages differ.

Fig. 3 demonstrates the sensitivity of solutions to the CTCP to perturbations in parameter values. It shows optimal plans for two realisations; the realisations differ in the value of only one parameter: $K_1 = 3$ versus $K_1 = 4$. This sensitivity, first, explains why robust plans are hard to find, and second, makes interval reasoning on the parameters difficult to apply.

V. Related Work

Planning in the space domain is reviewed in [2]; here and in aeronautics, active research is ongoing into autonomous systems, and planning is central to their behaviour [14].

An example, noteworthy as the first autonomous system to go into space, is the DS1 Remote Agent. Here, an embedded constraint planner deals with medium-term spacecraft activities while a low-level system provides short-term reactions [10]. Although DS1 is innovative, it handles only limited constraint classes and lacks modelling of ill-known parameters.

Our approach to planning under uncertainty corresponds to *contingent* planning (for mcs) and *conformant* planning (for mrp) [15]. However, application of much existing work on planning under uncertainty to avionic equipment control is difficult. Besides the domain-specific requirements noted

earlier, actions must be scheduled with respect to rich temporal constraints, and the system must cope with large-scale problems. Moreover, for low-level components, behaviour must be guaranteed in the worst case. The latter point, together with the difficulty of estimating probabilities, also hampers the use of Markov Decision Processes.

Of the planners that model incomplete information, MBP uses a language which conceives of a plan as a (deterministic) finite state automaton [16]. MBP accommodates uncertainty in initial state, besides non-deterministic actions and partially-observable effects. In contrast to our approach, it uses disjunctions rather than intervals to represent uncertainty, and is not designed to handle temporal nor heterogeneous constraints.

Despite the development of generic, expressive constraint-based planners, less work considers constraint-based planning under uncertainty. One exception is planning with a class of universally quantified constraints for incomplete information [17]. On the other hand, robust planning with constraints has been successfully shown for simple temporal problems with uncertainty in task durations [18]. Away from the fields of planning and intelligent control, robust computation is well-developed in both engineering and optimisation, e.g. [19].

More generally, handling uncertainty in constraint programming is an emerging area of research [12]. Robust decision making under anticipated future events is considered in [20]. Our search for a conditional decision uses techniques from the mixed CSP framework [11].

In constraint-based control, a generic framework based on multiple constraint solvers is presented in [21], while the advantages of composing logical propositions and constraint formulations in modelling are presented in [6]. In model-predictive control (MPC), constrained optimisation techniques can be used to solve the plant control problem online [7]. Our approach echoes min-max robust MPC, in that satisfaction of the control problem is guaranteed for every realisation. However, in general, uncertainty is not tackled using constraint-based planning approaches.

VI. Discussion and Future Work

This paper illustrates how to model uncertainty in planning the activities of aerospace equipment. We use a constraint-based model approach which allows expressive modelling of equipment and its actions. We have addressed incomplete knowledge in parameter values by providing a conditional plan. Each branch of the plan corresponds to synthesising a deterministic finite state automaton, capable of discrete event commanding of the equipment. This ensures that system behaviour requirements are met. Since the planning is done offline, the response time to (anticipated) contingent events is minimal. On a representative example, experimental results, even with preliminary algorithms, indicate the feasibility of the approach and the robustness of the conditional plan.

Plan generation is only one part of an autonomous system. Execution of the conditional plan our approach provides involves two factors. The first is when the true values of the parameters will be acquired; the second is the interleaving of planning and execution. Besides studying larger and broader examples, future work will look to integrate planning functions into avionic architectures exposed to uncertainty.

Acknowledgements

We thank Gérard Verfaillie for his advice. This work was partially supported by the EPSRC under grant GR/N64373/01.

References

[1] G. Verfaillie, "What kind of planning and scheduling tools for the future autonomous spacecraft?" in *Proc. ESA Workshop on On-Board Autonomy*, Oct. 2001.

[2] A. Jonsson, P. Morris, N. Muscettola, K. Rajan, and B. Smith, "Planning in interplanetary space: Theory and practice," in *Proc. AIPS'00*, 2000.

[3] B. Allo, C. Guettier, V. Legendre, J. Poncet, and N. Strady-Lecubin, "Constraint model-based planning and scheduling with multiple resources and complex collaboration schema," in *Proc. AIPS'02*, 2002.

[4] G. Berry, A. Bouali, X. Fornari, E. Ledinot, E. Nassor, and R. de Simone, "Esterel: a formal method applied to avionic software development," *Science of Computer Programming*, vol. 36, pp. 5–25, 2000.

[5] G. Verfaillie, E. Bensana, C. Michelon-Edery, and N. Bataille, "Dealing with uncertainty when managing an earth observation satellite," in *Proc. 2nd Intl. Symp. on Spacecraft Ground Control and Data Systems*, 1999.

[6] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints," Automatic Control Laboratory, ETH, Zurich, Tech. Rep. AUT98-04, 1998.

[7] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, June 2000.

[8] P. Kim, B. C. Williams, and M. Abramson, "Executing reactive, model-based programs through graph-based temporal planning," in *Proc. IJCAI'01*, Aug. 2001.

[9] H. Fargier, J. Lang, R. Martin-Clouaire, and T. Schiex, "A constraint satisfaction framework for decision under uncertainty," in *Proc. UAI'95*, Aug. 1995, pp. 167–174.

[10] D. Bernard, G. Dorais, E. Gamble, B. Kanefsky, J. Kurien, G. K. Man, W. Millar, N. Muscettola, P. Nayak, K. Rajan, N. Rouquette, B. Smith, W. Taylor, and Y.-W. Tung, "Spacecraft autonomy flight experience: The DS1 Remote Agent experiment," in *Proc. AIAA'99*, June 1999.

[11] H. Fargier, J. Lang, and T. Schiex, "Mixed constraint satisfaction: A framework for decision problems under incomplete knowledge," in *Proc. AAAI-96*, Aug. 1996, pp. 175–180.

[12] N. Yorke-Smith and C. Gervet, "Certainty closure: A framework for reliable constraint reasoning with uncertainty," in *Proc. CP'03*, Sept. 2003, pp. 769–783.

[13] *ECL^{i}PS^e User Manual Version 5.5*, IC–Parc, Nov. 2002.

[14] N. Muscettola, G. A. Dorais, C. Fry, R. Levinson, and C. Plaunt, "IDEA: Planning at the core of autonomous reactive agents," in *Proc. AIPS'02 Workshop on On-line Planning and Scheduling*, Apr. 2002, pp. 49–55.

[15] B. Bonet and H. Geffner, "GPT: A tool for planning with uncertainty and partial information," in *Proc. IJCAI-01 Workshop on Planning with Incomplete Information*, Aug. 2001.

[16] P. Bertoli, A. Cimatti, M. Pistore, M. Roveri, and P. Traverso, "MBP: A model based planner," in *Proc. IJCAI-01 Workshop on Planning under Uncertainty and Incomplete Information*, Aug. 2001.

[17] K. Golden and J. Frank, "Universal quantification in a constraint-based planner," in *Proc. AIPS'02*, Apr. 2002, pp. 61–70.

[18] P. Morris, N. Muscettola, and T. Vidal, "Dynamic control of plans with temporal uncertainty," in *Proc. IJCAI'01*, Aug. 2001, pp. 494–502.

[19] P. Kouvelis and G. Yu, *Robust Discrete Optimization and its Applications*. Kluwer, 1996.

[20] D. W. Fowler and K. N. Brown, "Branching constraint satisfaction problems for solutions robust under likely changes," in *Proc. CP-2000*, Sept. 2000, pp. 500–504.

[21] Y. Zhang, M. P. Fromherz, L. S. Crawford, and Y. Shang, "A general constraint-based control framework with examples in modular self-reconfigurable robots," in *Proc. 2002 IEEE/RSJ Conf. on Intelligent Robots and Systems (IROS 2002)*, Sept. 2002.