Synthesis of plans or policies for controlling dynamic systems

G. Verfaillie, C. Pralet, F. Teichteil, G. Infantes, C. Lesire

G. Verfaillie, C. Pralet, V. Vidal,
F. Teichteil, G. Infantes, C. Lesire
(Onera)

E-mail: gerard.verfaillie@onera.fr

Synthesis of plans or policies for controlling dynamic systems

To be properly controlled, dynamic systems need plans or policies. Plans are sequences of actions to be performed, whereas policies associate an action to be performed with each possible system state. The model-based synthesis of plans or policies consists in producing them automatically starting from a model of the physical system to be controlled and from user requirements on the controlled system. This article is a survey of what exists and what has been done at Onera for the automatic synthesis of plans or policies for the high-level control of dynamic systems.

A high-level reactive control loop

Aerospace systems have to be controlled to meet the requirements for which they have been designed. They must be controlled at the lowest levels. For example, an aircraft must be permanently controlled to be automatically maintained at a given altitude or to go down to a given speed. They must be controlled at higher levels too. For example, an autonomous surveillance UAV must decide on the next area to visit at the end of each area visit (highest navigation level). After that, it has to build a feasible trajectory allowing it to reach the chosen area (intermediate guidance level). Then this trajectory is followed using an automatic control system (lowest control level).

At any level, automatic control takes the form of a control loop as illustrated by figure 1. At any step, the controller receives observations from the dynamic system and sends it back commands; commands result in changes in the dynamic system and then in new observations and commands at the next step.

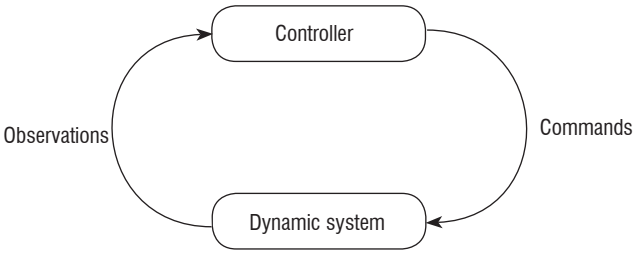

Figure 1 - Control loop of a dynamic system

At the lowest levels, for example for the automatic control of an aircraft, the dynamic system is usually modelled using a set of differential equations (domain of continuous automatic control [29]). However, at the highest levels, for example for the automatic navigation of an UAV, it is more conveniently modelled as a discrete event dynamic system (domain of discrete automatic control [14]): instantaneous system transitions occurring at discrete times; if the system is in a state s and an event e occurs then it moves instantaneously to a new state s' and remains in this state until the next event occurs.

In some cases, these transitions can be assumed to be deterministic: there is only one possible state s' following s and e. In other cases, due to uncertain changes in the environment, actions of other uncontrolled agents, or uncertain effects of controller actions, they are not deterministic: there are several possible states s' following s and e. For example, the result of an observation action by an UAV may depend on atmospheric conditions.

In some cases the controller has access to the whole system state at each step (full observability). However, in many cases it only has access to observations that do not allow it to know exactly the current system state (partial observability). For example, an automatic reconfiguration system may not know the precise subsystem responsible for faulty behavior but it must act in spite of its partial knowledge. In certain particular cases no observation is available (null observability).

In many cases, the system is assumed to run infinitely: in fact, its death is not considered. However, in some cases it is assumed to stop when reaching certain conditions. This is the case when we consider a specific mission for an UAV which ends when the UAV lands at its base.

User requirements on the behavior of the controlled system may take several forms. Most generally, they take the form of properties

(constraints) to be satisfied by the system trajectory (sequence of states followed by the system), or of a function of the system trajectory to be optimized. Usual properties are safety and reachability properties: a safety property must be satisfied at any step whereas a reachability property must be only satisfied at a certain step. The requirement that the energy level on board a spacecraft must remain above a minimum level is an example of a safety property. The fact that an UAV must visit a given area is an example of a reachability property.

The most general form of a controller is a policy which associates with each pair consisting of an observation o and a controller state cs, another pair consisting of a command c and a new controller state cs' (see figure 2).[1] The controller state is useful for recording relevant features of past observations and commands, as well as current objectives. Note that command c and controller state updating cs' are deterministic whereas system state transition s' may not be.[2] In the particular case of a finite trajectory with determinism or with non-determinism without observability (when observability is useless or impossible), a controller may take the simpler form of a plan which is a sequence of commands.

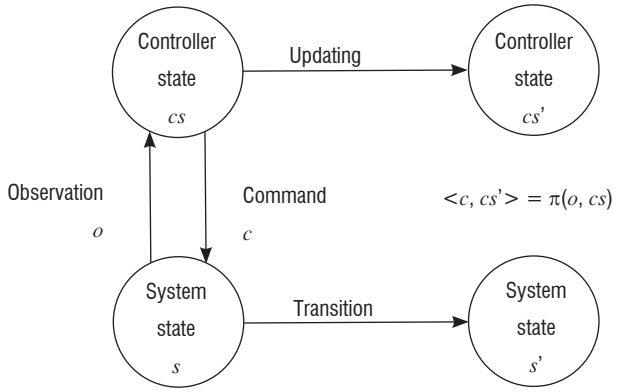

Figure 2: Controller implementing a policy π

The main question of discrete automatic control is then "how to build a controller (a policy or a plan) that guarantees that requirements on the controlled system are satisfied, possibly in an optimal way?".

To do this, three main approaches can be distinguished: (1) manual design, (2) automatic learning, and (3) automatic synthesis. The first approach, which is by far the most usual, assumes the existence of human experts or programmers who are able to define a controller under the form of decision rules which point out what to do in each possible situation, for each pair (observation, controller state). Model checking techniques [18] can then be used to verify that the resulting controller satisfies requirements. The second approach, which is being used more and more, consists in automatically learning a controller from experience, that is, from a set of tuples (observation, controller state, command, effects), which can be obtained by running the system or by simulating it [47]. The third approach, which is the most used at Onera for the control of aerospace systems because it offers the best guarantees in terms of requirement satisfaction, consists in automatically synthesizing a controller from a model of the dynamic system, the controller and the requirements on the controlled system. This third option is the approach we develop in this article.

Control synthesis modes

When it is embedded in an aircraft or spacecraft, the controller may be strongly limited in terms of memory and computing power. In spite of these limitations it has to make reactive decisions or at least make decisions by some specified deadlines. For example, the next area to be visited by an UAV and the trajectory that allows this area to be reached must be available by the end of the visit of the current area.

To satisfy these requirements the most usual approach consists in synthesizing the controller off-line, before execution. As soon as it has been built, the controller can be implemented and then reactively executed on board.

However, synthesizing a controller off-line requires that all the possible situations be taken into account. Because the number of possible situations can quickly become astronomical (settings with 10^{100}, 10^{500}, or 10^{1000} possible situations are not rare), difficulties quickly arise in terms of controller synthesis, memorization, and execution.

To overcome such difficulties an option is available that consists in synthesizing the controller on-line on-board: given the current context, the number of situations to be considered can be dramatically reduced; if computing resources are sufficient on-board, the synthesis of a controller dedicated to the current context may be feasible by the specified deadlines; as long as the current context remains unchanged, this controller remains valid; as soon as the context changes, a new controller is synthesized. See [60] for a generic implementation of such an approach. Anytime algorithms [72], which quickly produce a first solution and try to improve on it as long as computing time is available, may be appropriate in such a setting.

When the state is fully observable and the possible state evolutions can be reasonably approximated by a deterministic model, at least over a limited horizon ahead, an option consists in (i) synthesizing a simpler controller under the form of a plan over a limited horizon, (ii) keeping with this plan as long as it remains valid (no violated assumptions and a sufficient horizon ahead taken into account), and (iii) replanning as soon as it becomes invalid. Such an option (planning/replanning) is widely used because of its simplicity and efficiency. See [31] for an example of application to the control of Earth observation satellites.

Another option, close to the previous one and to the model predictive control [24] approach developed in the domain of continuous automatic control, consists, at each step, in (i) observing the current state, (ii) synthesizing a controller in the form of a plan over a limited horizon (reasoning horizon), (iii) applying only the first command of this plan (decision horizon), and (iv) starting again at the next step over a sliding horizon. A simple decision rule (valid, but non optimal) is applied when no plan has been produced. See [44] for a generic

[1] This approach of control differs from the approach adopted by the pioneering works on discrete control [53] which consider that the role of control is not to specify a command, but to restrict the set of acceptable commands.

[2] We do not consider in this article the case of non deterministic controllers which select commands in a non deterministic stochastic manner.

implementation of such an approach and [5] for an example of application to the autonomous control of an Earth observation satellite.

Models

The automatic controller synthesis approach requires models of the dynamic system and of the requirements on the controlled system.

Model of the dynamic system

Dynamic systems are usually modelled using:

- a finite set S of possible states;
- a finite set O of possible observations;
- a finite set A of possible actions;
- an initialisation model I which defines possible initial states;
- an observation model Om which defines possible state-observation pairs;
- a feasibility model F which defines feasible state-action pairs;
- a transition model T which defines possible state transitions.

When transitions are deterministic, T is a function from $S\times A$ to S. When they are not deterministic, T is a relation over $S\times A\times S$. When probabilistic information is available, T is a function which associates with each element of $S\times A$ a probability distribution over S. Similarly, Om may be defined by a function from S to O, by a relation over $S\times O$, or by a function which associates with each element of S a probability distribution over O. I may be defined by an element of S, a subset of S or a probability distribution over S. Finally, F is defined by a relation over $S\times A$.

It must be emphasized that the transition model is here assumed to be Markovian: the next state s' depends only on the current state s and action a; it does not depend on previous states and actions. When this assumption is not satisfied it is necessary to add relevant features of past states and actions in the state definition to get it satisfied. It must be stressed too that, in this presentation, a state includes the system state and the controller state. Hence, with regard to figure 2, the transition model includes system state transition and controller state updating.

Requirements on the controlled system

Finally, user requirements on the controlled system are modelled using a requirement model R which may take several forms:

- R may be a subset of S which defines the set of acceptable final states, also called goal states (reachability property);
- R may be a relation over $S\times A\times S$ which defines the set of acceptable transitions (safety property), not to be mistaken for the set T of possible transitions, previously defined in the model of the dynamic system;
- R may be a function from $S\times A\times S$ to the set of reals which associates a local reward with each transition; the total reward associated with a trajectory is then the sum of the local rewards associated with the successive transitions.

More complex requirements on the system trajectories may be expressed using temporal logics [20] or non Markovian rewards that are rewards on state trajectories [63].

Compact representations

Usually, sets S, O, and A of states, observations, and actions are compactly defined using finite sets of state, observation, and action variables whose domains of value are finite (factored representation). For example, if Sv is the set of state variables, S is defined as $\prod_{v\in Sv} D(v)$, where $D(v)$ is the domain of v. Similarly, initialisation, observation, feasibility, and transition relations are compactly defined using constraints [54] or decision diagrams [13]. Initialisation, observation, and transition probability distributions may also be compactly defined using Bayesian networks [40], valued constraints [55] or algebraic decision diagrams [2].

Usual frameworks

For example, in the classical AI planning framework (Artificial Intelligence planning [28]), the initialisation model is defined by a state (only one possible initial state), the feasibility model by action preconditions, the transition model by deterministic action effects, and the requirement model by a set of goal states. The objective is to build a plan that allows a goal state to be reached. Observation is useless because of determinism. Specific languages, such as PDDL [23, 27], have been developed in the context of the International Planning Competition (IPC) to allow users to express models in a compact and natural way.

Another example, in the classical MDP framework (Markov Decision Processes [52]), is where initialisation and transition models are defined by probability distributions, there is no observation model (assumption of full observability) and the requirement model has the form of additive local rewards. The objective is to build a policy that maximizes the expected total reward over finite horizons or the expected discounted total reward over infinite ones (reward geometrically decreasing as a function of the step number). In the POMDP framework (Partially Observable MDP [37]), the observation model is defined by a probability distribution.

In the goal MDP framework [71, 61], which is a hybrid between AI planning and MDP, a set of goal states is added to the MDP definition and local rewards are replaced by costs. The objective is to build a policy that minimizes the expected total cost to reach a goal state.
Finally, in a framework that can be referred to as the logical MDP framework [8, 51], initialisation, observation, transition, and requirement models are defined by relations. The objective is to build a policy that guarantees that reachability and/or safety requirements are satisfied in spite of non determinism and partial observability.

Other models

It must be emphasized that, although these models are generic, many real problems of synthesising plans or policies in the aerospace domain are more conveniently modeled using different frameworks, popular in the Operations Research community: scheduling, resource assignment, knapsack, shortest path, or traveling salesman problems [3, 39, 43]. For example, the main objective of plan synthesis is to build a sequence of actions allowing a goal to be reached. However, when planning activities for an Earth observation satellite, the problem is not to discover the sequence of basic actions allowing an area a to be observed: one knows that is necessary to set the instrument ON,

to point the satellite towards the beginning of a during a's visibility window, to start observing, to memorize data, and then to download it using a station visibility window. The HTN framework (Hierarchical Task Network [21, 56]) may be used to describe such a breakdown of a task into sub-tasks. The main problem is to organize observations over time and resources in order to perform a maximal subset of them of maximum value. In these problems, time and resource management is central and an explicit representation of the system state may be useless. See [45] for an example.

It must be however stressed that standard Operations Research problems rarely allow real problems to be completely and precisely modelled. For example, many problems are over-constrained scheduling problems with complex time and resource constraints to be satisfied and a complex optimization function to be optimized. See [30] for an example.

It must be emphasized too that many problems in the aerospace domain combine action planning [28] and motion planning [42]. For example, inserting the visit of an area into the activity plan of an UAV requires checking the feasibility of the trajectory allowing the UAV to reach this area and to compute the effects of this movement, for example in terms of energy. See [33] for the proposal of a generic scheme for cooperation between action and motion planning.

To sum up, many real problems appear to be complex hybrids of action planning, motion planning, and task scheduling. The CNT framework (Constraint Network on Timelines [67]) has been developed to try and model them as well as possible. It extends the basic CSP framework (Constraint Satisfaction Problem [54]) by defining horizon variables that represent the unknown number of steps in system trajectories and by defining dynamic constraints as functions which associate a set of classical CSP constraints with each assignment of the horizon variables.

Optimality equations

Optimality equations, also called Bellman equations [6], allow satisficing or optimal policies to be characterized. In the MDP framework over infinite horizons, they can be defined as follows.

Let $I(s)$ be the probability of being initially in state s, $F(s,a)$ be a Boolean function which returns true when action a is feasible in state s, $T(s,a,s')$ be the probability of being in state s' after applying action a in state s, $R(s,a,s')$ be the local reward associated with this transition, and $\gamma \in [0,1[$ be the discount factor.

Let $V^*(s,a)$ be the optimal expected total reward it is possible to obtain when starting from state s and applying action a, $V^*(s)$ be the optimal expected total reward it is possible to obtain when starting from state s, V^* be the optimal expected total reward it is possible to obtain taking into account the possible initial states, and π^* be an optimal policy. It can be shown that the following equations must be satisfied:

$$\forall s \in S, \forall a \in A / F(s,a) : V^*(s,a) = \sum_{s' \in S} T(s,a,s').(R(s,a,s') + \gamma.V^*(s'))$$

$$\forall s \in S : V^*(s) = \max_{a \in A/F(s,a)} V^*(s,a)$$

$$\forall s \in S : \pi^*(s) = \arg\max_{a \in A/F(s,a)} V^*(s,a)$$

$$V^* = \sum_{s \in S} I(s).V^*(s)$$

While the values $V^*(s,a)$, $V^*(s)$, and V^* are the only solutions of this set of equations, several associated optimal policies are possible.

In the goal MDP framework, there is no discount factor, $V^*(s)=0$ for any goal state s, and maximization is replaced by minimization.

In the logical MDP framework, these equations can be reformulated as follows.

Let $I(s)$ be true when s is a possible initial state, $F(s,a)$ be true when action a is feasible in state s, $T(s,a,s')$ be true when s' is a possible state after applying action a in state s, and $R(s,a,s')$ be true when this transition is acceptable (safety properties).

Let $V^*(s,a)$ be true when it is possible to satisfy the safety properties when starting from state s and applying action a, $V^*(s)$ be true when it is possible to satisfy the safety properties when starting from state s, and π^* be a satisfying policy. It can be shown that the following equations must be satisfied:

$$\forall s \in S, \forall a \in A / F(s,a) : V^*(s,a) = \wedge_{s' \in S}(T(s,a,s') \rightarrow (R(s,a,s') \wedge V^*(s')))$$

$$\forall s \in S : V^*(s) = \vee_{a \in A/F(s,a)} V^*(s,a)$$

$$\forall s \in S : \pi^*(s) = \text{argtrue}_{a \in A/F(s,a)} V^*(s,a)$$

$$\forall s \in S : I(s) \rightarrow V^*(s)$$

Algorithms

Dynamic programming

Dynamic programming algorithms [6] make direct use of the optimality equations to produce satisfying or optimal policies. The most popular variant, referred to as value iteration, approximates better and better values $V^*(s)$, starting from any initial values $V_0(s)$. In the classical MDP framework, at each algorithm iteration step $i>0$, values are updated in the following way:

$$\forall s \in S, \forall a \in A / F(s,a) : V_i(s,a) = \sum_{s' \in S} T(s,a,s').(R(s,a,s') + \gamma.V_{i-1}(s'))$$

$$\forall s \in S : V_i(s) = \max_{a \in A/F(s,a)} V_i(s,a)$$

It can be shown that values $V_i(s)$ asymptotically converge to $V^*(s)$. Practically, the algorithm stops when $\max_{s \in S} |V_i(s) - V_{i-1}(s)|$ is below a given threshold. When it stops at step i, a policy π can be extracted using the following equation:

$$\forall s \in S : \pi(s) = \arg\max_{a \in A/F(s,a)} V_i(s,a)$$

In the logical MDP framework, initial values $V_0(s)$ are all true and convergence is reached in a finite number of iterations.

These algorithms are the most natural way of getting an optimal policy when the number of states remains reasonably small. However, because the number of states is an exponential function of the number of state variables, they may quickly become impracticable.

To overcome such a difficulty, special structures can be used inside dynamic programming algorithms. In the logical MDP framework, binary decision diagrams (BDD [13]), allowing a Boolean function of Boolean variables to be represented, can be used to represent relations compactly and manipulate them [17]. This technique can be extended to classical MDP using algebraic decision diagrams (ADD [2]), allowing a real function of Boolean variables to be represented.

Heuristic search

Whereas dynamic programming algorithms consider all possible states, heuristic search algorithms consider only those that are reachable from the possible initial states by following the policies that are considered: a potentially very small subset of the set of possible states.

Although these algorithms are not limited to AI Planning problems, they can be easily presented in this framework. In AI Planning, we have one possible initial state, positive action costs, deterministic transitions, and a set of goal states. The problem can be formulated as a shortest path problem in a weighted oriented graph where nodes are associated with states, arcs with transitions, positive weights with action costs: a shortest path is sought from the node associated with the unique initial state to any node associated with a goal state.

Efficient algorithms exist to produce shortest paths from an initial node n_0 to any node in weighted graphs with positive weights, such as the well-known Dijkstra algorithm [19]. Let $W(n,n')$ be the weight of the edge from node n to node n'. This algorithm incrementally builds a tree of shortest paths from n_0 to any node n, rooted in n_0. At each algorithm step, with any node n, are associated an upper bound $V(n)$ on the minimum length to go from n_0 to n, and the parent node $P(n)$ of n in the current tree. Values $V(n)$ are initialized with $+\infty$, except 0 for the initial node. Parent nodes $P(n)$ are initialized with $\varnothing$. At each algorithm step, the algorithm visits a new node. The selected node is a node of minimum value $V(n)$. For each node n' that has not been visited yet and can be reached directly from n, such that $V(n)+W(n,n')<V(n')$, $V(n')$ is updated to $V(n)+W(n,n')$ and $P(n')$ to n. The algorithm stops when all nodes have been visited. It is guaranteed that, for each node n, $V(n)$ is then the minimum distance from n_0 to n. The associated shortest path can be built backward using $P(n)$. This algorithm visits only the nodes that are reachable from n_0.

However, when we search for shortest paths from an initial node to a set of goal nodes, more efficient algorithms exist, such as the well-known A^* algorithm [35]. This algorithm works as the Dijkstra algorithm does, except that values $V(n)$ are replaced by values $V'(n)$ which are the sum of two values: a value $V(n)$ which is an upper bound on the minimum distance from n_0 to n, and a value $H(n)$ which is a lower bound on the minimum distance from n to any goal node. Values $V(n)$ are initialized and updated the same way as they are in the Dijkstra algorithm. As for values $H(n)$, they are assumed to be given by an admissible (optimistic) heuristic function, null for any goal node. At each algorithm step, the selected node is a node of minimum value $V'(n)$. If the heuristic function is monotone $(\forall n,n': H(n)\leq H(n')+W(n,n'))$ then a node cannot be revisited, but if it is not then a node may be visited several times. The algorithm stops when the selected node is a goal node. It is guaranteed that the value $V(n)$ of this node is the minimum distance from the initial node to any goal node. With regard to the Dijkstra algorithm, the main advantage of this algorithm is its focus on goal states via the heuristic function. Its efficiency strongly depends on this heuristic. The better the heuristic function $H(n)$ approximates the minimum distance from n to any goal node, the fewer nodes are visited. The Dijkstra algorithm is a particular case of A^* where the heuristic function is null for any node.

The most efficient algorithms for solving AI planning problems, such as HSP (Heuristic Search Planner [11]) or FF (Fast Forward [36]), combine sophisticated variants of A^* with powerful heuristic computations.[3] The heuristic function is automatically built by solving specific problem relaxations at each node of the search. Some of these heuristics are admissible (thus yielding optimal plans) as the optimum of any problem relaxation is a lower bound on the optimum of the original problem. The YAHSP planner (Yet Another Heuristic Search Planner [68]) uses variants of these principles as well as a lookahead strategy which causes the planner to focus more quickly on promising parts of the graph. It must be stressed that the graph is never explicitly built. It is only explored as and when required by the effective search.

Heuristic search can be generalized to planning under uncertainty. For example, the LAO^* algorithm [34] solves goal MDP problems via a combination of heuristic forward search and dynamic programming. The RFF algorithm (Robust FF [59]) solves the same problems using successive calls to FF: a deterministic model of the goal MDP is first built by considering the most likely initial state and, for each state and each feasible action, the most likely transition; a plan is built using this deterministic model; then, this plan, which is a partial policy, is simulated using the original non deterministic model; for each reachable state s that is not covered by the current policy, a plan is built from s using the deterministic model, and so on until all reachable states or nearly of them are covered by the current policy. If all reachable states are covered then the resulting policy guarantees goal reachability but may not be optimal.

Greedy search

Greedy search is a very simple technique for dealing with combinatorial optimization problems. It consists in making successive choices, following a given heuristic, without ever reconsidering previous choices. For example, in AI planning, it is possible to systematically choose as a next node a node n that minimizes the heuristic function $H(n)$. It is clear that this method offers no guarantee in terms of goal reachability and optimality.

However, repeated greedy searches, combined with learning, can offer these guarantees. For example, the $LRTA^*$ algorithm (Learning Real Time A^* [41]) solves shortest path problems by performing a sequence of greedy searches. Each search starts from the initial node n_0 and greedily uses a heuristic function V which is a lower bound on the minimum distance to a goal node. V is initialized by any admissible (optimistic) heuristic. When the current node is n, a node n' that minimizes $W(n,n')+V(n')$ is selected, $V(n)$ is updated to $W(n,n')+V(n')$, and n' becomes the current node. The search stops when the current node is a goal node. A new search can start using the current heuristic function V. It can be shown that, search after search, this function converges to the minimum cost to reach a goal node. This algorithm can be straightforwardly generalized to planning

[3] In fact, FF combines tree search with hill climbing local search.

under uncertainty. See the *RTDP* algorithm (Real Time Dynamic Programming [4]) which solves goal MDP problems and can be seen as a special way of exploiting Bellman optimality equations. This use of simulations of the dynamic system to learn good policies is generalized by reinforcement learning techniques [57].

Iterated stochastic greedy search [12] is another interesting variant where heuristic choices are randomized and greedy searches are repeated. See [48, 5] for examples of application to planning for Earth observation satellites.

Local search

Local search [1] is a very powerful technique for the approximate solving of combinatorial optimization problems. Starting from any solution, it improves on it iteratively by searching for a better solution in the neighbourhood of the current solution: a small set of solutions that differ slightly from the current one. Although it cannot guarantee optimality, this method is generally able to produce high quality solutions quickly thanks to so-called meta-heuristics such as simulated annealing, tabu search, or evolutionary algorithms which allow a search to escape from so-called local minima: no better solution in the neighbourhood of the current solution.

It has been successfully applied to AI planning [26]. In [9], classical planners and evolutionary algorithms are combined to produce high quality plans. It is widely used to deal with scheduling problems with time and resource constraints. See [45] for an example of application to the problem of scheduling observations performed by an agile satellite.

Constraint-based approaches

The SAT and CSP frameworks (Boolean SATisfiability [10] and Constraint Satisfaction Problem [54]) are widely used to model and solve problems where one searches for assignments to a given set of variables that satisfy a given set of constraints and optimize a given criterion. To model AI planning problems, we can associate one SAT/CSP variable with each state or action variable at each step. However, the difficulty is that the number of steps in a plan, and thus the number of SAT/CSP variables, is unknown.

This is why SAT and CSP techniques have been first used to solve planning problems over a given number of steps [38, 64]. One can start with only one step and increment the number of steps each time a plan has not been found, until a plan is finally found.

In the CNT framework (Constraint Network on Timelines [67]), this unknown number of steps is taken into account inside the constraint-based model via the use of so-called horizon variables. See [49, 50] for optimal and anytime associated algorithms.

In [69] an alternative constraint-based formulation is proposed with variables associated with each possible action, present or not in the plan.

These constraint-based approaches are not limited to AI planning problems. They can be used for planning under uncertainty. See [51, 66] for two approaches applicable to the logical MDP framework.

Applications

In this section we show some selected examples of plan or policy synthesis problems we have had to deal with in the aerospace domain.

Search and rescue mission with an unmanned helicopter

For some years, Onera has been using the ReSSAC platform (see figure 3) for studying, developing, experimenting, and demonstrating the autonomous capabilities of an unmanned helicopter [22]. In such a setting, Onera researchers considered a mission of search and rescue of a person in a given area.

Figure 3 - An Onera ReSSAC unmanned helicopter

After taking off from its base and reaching the search area, the helicopter takes, at high altitude, a wide picture of this area and extracts possible landing sub-areas. Before landing in any of these sub-areas, it must explore it at lower altitude in order to be sure that landing is safe. After landing in any sub-area, the searched person reaches the helicopter, which takes off to come back to its base.

The mission goal is to come back to the base with the searched person. Fuel is limited. For any sub-area, there is uncertainty about the fuel consumed in its exploration and about the possibility of landing safely in it. The problem is to determine in which order sub-areas are visited.

Because of uncertainties, producing a plan off-line to be executed by the helicopter is not a valid approach. A policy has to be produced either off-line from the initial state, or on-line from the current state.
The problem has been modeled in the MDP framework. State variables include discrete Boolean variables pointing out whether or not a given area has already been explored and a continuous variable representing the current level of fuel. Action variables include a discrete variable representing the next sub-area to be visited. Transition probabilities are assumed to be available. There is no reward except when the helicopter comes back to its base with the searched person. All state variables are observable, but some of them are continuous. The result is thus a hybrid MDP [32].

To solve it, a hybrid version of the RTDP algorithm (Real Time Dynamic Programming [4]), called HRTDP for Hybrid RTDP [58], has

been developed. HRTDP works as RTDP does, using greedy search, sampling, and learning, except that value functions which associate a value with each discrete-continuous state are approximated using regressors, and policies which associate an action with each discrete-continuous state are approximated using classifiers.[4]

We are currently working on more complex missions involving target recognition, identification, and tracking by an unmanned helicopter, for which we are making use of POMDP techniques [15].

Planning airport ground movements

At airports, aircraft must move safely from their landing runway to their assigned gate and, in the opposite direction, from their gate to their assigned runway.

To assist airport controllers, safe ground movement plans can be automatically built, taking into account a given set of flights over a given temporal horizon ahead.

To build such plans, the airport (see figure 4) is modelled as an oriented weighted graph where vertices represent runway access points, gates, or taxiway intersections, arcs represent taxiways, and weights represent taxiway lengths. In the proposed approach [46], flight movements are planned flight after flight, according to their starting time order. For each flight, a movement plan is built, taking into account the plans built for the previous flights. For each flight, the problem is to find a shortest path in the graph in terms of time, taking into account time separation constraints between aircraft at each vertex of the graph. An algorithm is used to solve it optimally. The result is a path in the graph, with a precise time associated with each vertex in the path.

These plans are too rigid and do not take into account the uncertainty about aircraft arrival and departure times and about aircraft ground speed. To overcome such a difficulty, in a second version of the algorithm, precise times associated with each flight and each vertex are replaced by time intervals. The resulting plan is flexible and remains valid as long as these time intervals are adhered to by the aircraft.

Management of an autonomous Earth surveillance and observation satellite

For some years, Onera, CNES, and LAAS-CNRS have been involved in a joint project called AGATA [16] which aims at developing techniques for improving spacecraft autonomy. A target mission in this project was a fictitious mission, called HotSpot, using a constellation of small satellites for surveillance and observation of hotspots (forest fires or volcanic eruptions) at the Earth's surface [48] (see figure 5).

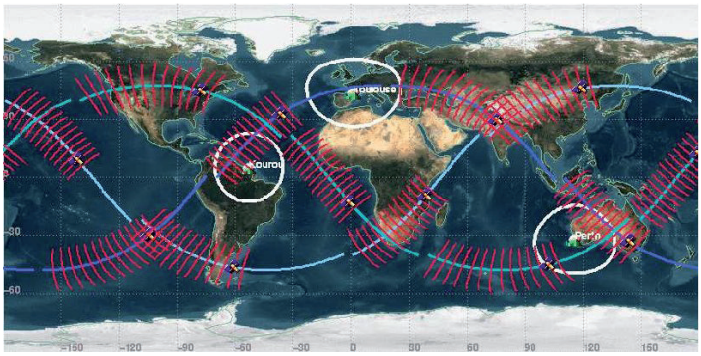

Figure 5 - Track on the ground of the HotSpot detection instrument (12 satellite constellation) within a 25 minute period

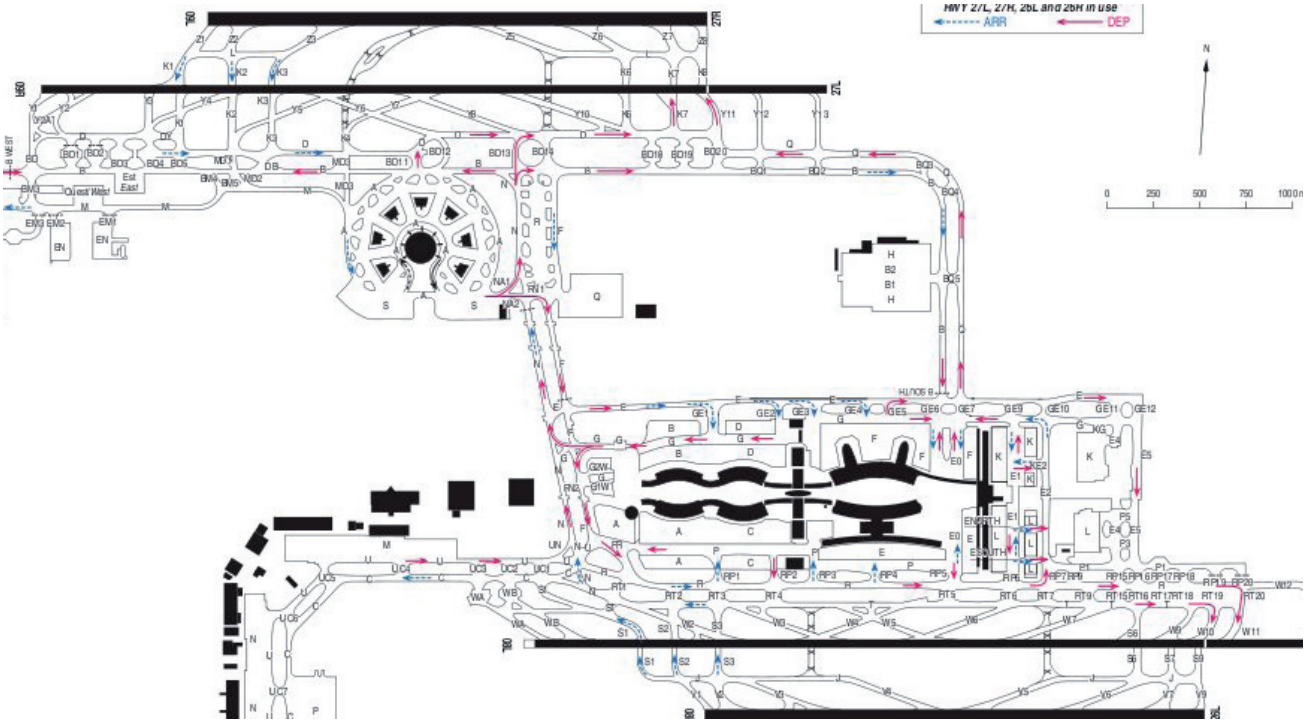

Figure 4 - A map of the Roissy Charles de Gaulle airport

[4] A regressor allows a function to be approximated. When this function takes discrete values, one speaks of classifier.

Each satellite is assumed to be equipped with a large aperture detection instrument, able to detect hotspots at the Earth surface. In the event of detection an alarm is sent to the ground via relay geostationary satellites. Each satellite is also equipped with a small aperture observation instrument, able to observe areas where hotspots have been detected or of any other areas whose observation is required by the ground mission center. Observation data is recorded on board and downloaded when the satellite is within a visibility window of a ground reception station. Decisions must be made on board on which areas to be observed, which data to be downloaded, and when downloads occur (when selected, observations occur at specific times with no temporal flexibility).

In this problem, uncertainty is due to the possible presence of hotspots. However, we do not have at our disposal any model of this uncertainty. This is why we adopted a planning/replanning approach (see § "Control synthesis modes"). Each observation and download plan is built over a given temporal horizon ahead, takes into account the known observation requests, and ignores the possible future requests that might follow hotspot detection. In case of detection of any unexpected event, a new plan is built.

To implement such an approach, we developed a generic reactive/deliberative control architecture [44] where a reactive module receives information from the environment, triggers a deliberative module with all the necessary information (starting state, requests, temporal horizon), and makes final decisions, and where the deliberative module performs anytime planning [72] and sends plans to the reactive module each time a better plan is found.

Because plans must be produced quickly, we developed an iterated stochastic greedy search. Each greedy search is performed chronologically from the starting state and produces a candidate plan. At each step, the algorithm makes a heuristic choice and checks that all the physical constraints are met. Observations and data downloads which can be performed in parallel are taken into account, as well as energy and memory profiles. Heuristic choices are randomized to explore plans in a neighbourhood around a reference plan (the plan that is produced by strictly following heuristics).

Autonomous decision about data downloading

Onera has been studying the problem of data downloading for a CNES electromagnetic surveillance mission using a constellation of satellites (see figure 6). In this mission, ground electromagnetic sources are tracked by satellites, data is recorded on board and then downloaded to ground reception centers [65].

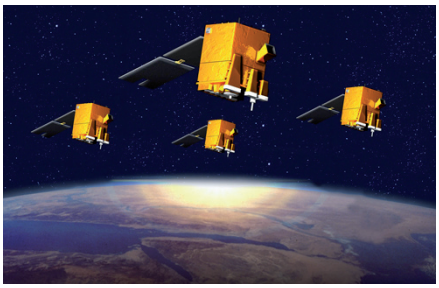

Figure 6 - The Elisa satellite constellation designed as a technological demonstration of electromagnetic surveillance capabilities from space

The main difficulty in this problem is that the volume of data that results from the tracking of a ground area is uncertain and that the variance of the probability distribution is very large. In such conditions, building data downloading plans off-line on the ground, which is how this is usually done, may be problematic. If maximum volumes are taken into account then downloading windows may be underused due to actual volumes being less than their maximum value. If mean volumes are taken into account then some downloads may be impossible due to actual volumes being greater than their mean value. In this problem, it is assumed that, for each ground area, a probability distribution on the volume of data generated by its tracking is available. In such conditions, if the tracking plan is known, an MDP model of the data downloading problem can be built. Solving it would produce a policy which would say which data is to be downloaded as a function of current time and memory state (data currently present in memory). However, the fact that the resulting MDP is a hybrid MDP [32] with a huge number of continuous variables (volume of each data in memory) has prevented us, at least for the moment, from following this approach.

More pragmatically we adopted a planning/replanning approach, close to the one used to solve the previous HotSpot problem (see the previous subsection). Each plan is built over a given sequence of downloading windows ahead and takes into account the known volumes for the data already in memory and the mean volumes for the others. Each time the tracking of a ground area ends and the generated volume is known, a new plan is built. Plans are built greedily by inserting data downloads one after the other. At each step, a download of the highest priority and, in the case of equality, of the highest ratio between its value and its duration is selected and inserted in the plan at the earliest possible moment (classical heuristics used to solve knapsack problems [39]).

Simulations show the superiority in terms of actual downloads of online on-board planning/replanning compared to off-line planning on the ground.

Challenges

Algorithmic efficiency

Algorithmic efficiency is the key issue for dealing with plan or policy synthesis problems. Most of the problems we address are not polynomial, but NP-hard or Pspace-hard according to the complexity theory in computer science [25]. This means that the worst-case time complexity of any known optimal algorithm grows at least exponentially with problem size (what is usually referred to as the combinatorial explosion) and that there is no serious hope of discovering polynomial algorithms. Thus, if the combinatorial explosion is unavoidable, the priority becomes to delay it as far as possible.

This is the role of many of the techniques we are working on, as a large number of other researchers are, such as efficient data structures, intelligent search strategies, intelligent sampling, high quality heuristics, constraint propagation and bound computing, explanation and learning, decomposition, symmetry breaking, incremental local moves, portfolios of algorithms, and the efficient use of multi-core processor architectures.

Generic vs. specific algorithms

In fact, in most of the applications we have had to deal with we did not use generic algorithms but developed specific ones, tuned for solving the specific problem at hand. The two main reasons for that are that (i) generic frameworks and algorithms are often unable to handle specific features of the problem and (ii) generic algorithms do not take into account problem specificities and are thus often too inefficient. However, this approach is very consuming in terms of engineer working time. Moreover, any small change in the problem definition may compel engineers to revisit the whole algorithm.

As a consequence, one of the challenges we have to face is the design of really generic modeling frameworks and of associated efficient generic algorithms, for at least some important problem classes to be identified. These algorithms should be tunable as much as possible as a function of the problem at hand. If we succeed then engineers could limit their work to problem analysis and modeling and to algorithm tuning.

Constraints and criteria

User requirements on the controlled system are usually of the form of constraints to be satisfied or of criteria to be optimized on all the possible system trajectories. However, some modeling frameworks such as temporal logics, classical AI planning, or logical MDP focus on constraint satisfaction, whereas other frameworks such as classical or goal MDP focus on criterion optimization. It would be very interesting to build a more general framework where various kinds of constraints and criteria on trajectories could together be represented and handled, in order to be able, for example, to synthesize safe optimal controllers.

Discrete and continuous variables

In most of the work on the problem of plan or policy synthesis for the high-level control of dynamic systems, time, state, and action variables are assumed to have discrete and finite domains of values. On the other hand, for work in the domain of continuous automatic control, it is continuous, time, state, and command variables that are considered. A challenge would be to put up a bridge between the discrete and continuous worlds in order to address problems of control of hybrid systems that can only be modeled using discrete/continuous time, state, and command variables.

Centralized and decentralized control

In this article, we have limited ourselves to problems where the control is centralized: whatever its dimension is, the physical system is controlled by a unique controller which receives all the observations and sends all the commands. However, in many situations, distributed control is either mandatory or desirable. This is the case when a fleet of vehicles (aircraft, spacecraft, ground robots, ground stations …) needs to be controlled in spite of non-permanent inter-vehicle communications. In this case, local decisions must be made by each vehicle with only a local view of the system. This kind of problem has already been formalized, using the Dec-MDP framework (Decentralized Markov Decision Processes [7]), where each agent has a local view of the system state and can only make local decisions, or the DCSP framework (Distributed Constraint Satisfaction Problem [70]), where decision variables are distributed among agents. Nevertheless, a lot of work remains to be done in this domain in terms of relevant modeling frameworks and efficient algorithms.

Human beings in the control loop

Finally, the presence of human beings who want to have the best view of the system state, want to control the system at the highest level, and want to be able to make their own decisions at any moment, is another challenge. Indeed, while it is sensible to assume that we have at our disposal models of the dynamics of artificial physical systems, this is no longer the case with human beings who may intervene as they wish, within the limits of the man-machine interaction system. See [62] in this issue of Aerospace Lab ■

References

[1] E. AARTS, J. LENSTRA, eds. - *Local Search in Combinatorial Optimization*. John Wiley & Sons, 1997.

[2] R. I. BAHAR, E. FROHM, C. GAONA, G. HACHTEL, E. MACII, A. PARDO, F. SOMENZI - *Algebraic Decision Diagrams and their Applications.* Proc. of the IEEE/ACM International Conference on Computer-aided Design (ICCAD-93), 1993.

[3] P. BAPTISTE, C. LE PAPE, W. NUIJTEN - *Constraint-based Scheduling: Applying Constraint Programming to Scheduling Problems.* Kluwer Academic Publishers, 2001.

[4] A. BARTO, S. BRADTKE, S. SINGH - *Learning to Act using Real-time Dynamic Programming.* Artificial Intelligence, 72 (1995), pp. 81–138.

[5] G. BEAUMET, G. VERFAILLIE, M. CHARMEAU - *Feasibility of Autonomous Decision Making on Board an Agile Earth-Observing Satellite.* Computational Intelligence, 27 (2011), pp. 123–139.

[6] R. BELLMAN - *Dynamic Programming.* Princeton University Press, 1957.

[7] D. BERNSTEIN, R. GIVAN, N. IMMERMAN, S. ZILBERSTEIN - *The Complexity of Decentralized Control of Markov Decision Processes.* Mathematics of Operations Research, 27 (2002), pp. 819–840.

[8] P. BERTOLI, A. CIMATTI, M. ROVERI, P. TRAVERSO - *Planning in Nondeterministic Domains under Partial Observability via Symbolic Model Checking*. Proc. of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01), Seattle, WA, USA, 2001, pp. 473–478.

[9] J. BIBAI, P. SAVÉANT, M. SCHOENAUER, V. VIDAL - *An Evolutionary Metaheuristic Based on State Decomposition for Domain-Independent Satisficing Planning.* Proc. of the 20th International Conference on Automated Planning and Scheduling (ICAPS-10), Toronto, Canada, 2010.

[10] A. BIERE, M. HEULE, H. VAN MAAREN, T. WALSH, eds. - *Handbook of Satisfiability*. IOS Press, 2009.

[11] B. BONET, H. GEFFNER - *Planning as Heuristic Search.* Artificial Intelligence, 129 (2001), pp. 5–33.

[12] J. BRESINA - *Heuristic-Biased Stochastic Sampling*. Proc. of the 13th National Conference on Artificial Intelligence (AAAI-96), Portland, OR, USA, 1996, pp. 271–278.

[13] R. BRYANT - *Graph-Based Algorithms for Boolean Function Manipulation*. IEEE Transactions on Computers, C-35 (1986), pp. 677–691.

[14] C. CASSANDRAS, S. LAFORTUNE - *Introduction to Discrete Event Systems*. Springer, 2008.

[15] C. P. C. CHANEL, J.-L. FARGES, F. TEICHTEIL-KÖNIGSBUCH, G. INFANTES - *POMDP Solving: What Rewards do you Really Expect at Execution?* Proc. of 5th European Starting AI Researcher Symposium (STAIRS-10), Lisbon, Portugal, 2010.

[16] M.-C. CHARMEAU, E. BENSANA - *AGATA: A Lab Bench Project for Spacecraft Autonomy*. in Proc. of the 8th International Symposium on Artificial Intelligence, Robotics, and Automation for Space (i-SAIRAS-05), Munich, Germany, 2005.

[17] A. CIMATTI, M. PISTORE, M. ROVERI, P. TRAVERSO - *Weak, Strong, and Strong Cyclic Planning via Symbolic Model Checking*. Artificial Intelligence, 147 (2003), pp. 35–84.

[18] E. CLARKE, O. GRUMBERG, D. PELED - *Model Checking*. MIT Press, 1999.

[19] E. DIJKSTRA - *A Note on Two Problems in Connection with Graphs*. Numerische Mathematik, 1 (1959), pp. 269–271.

[20] E. EMERSON - *Temporal and Modal Logic*. Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics, J. van Leeuwen, ed., Elsevier, 1990, pp. 995–1072.

[21] K. EROL, J. HENDLER, D. NAU - *UMCP: A Sound and Complete Procedure for Hierarchical Task-Network Planning*. Proc. of the 2nd International Conference on Artificial Intelligence Planning and Scheduling (AIPS-94), Chicago, IL, USA, 1994, pp. 249–254.

[22] P. FABIANI, V. FUERTES, G. LE BESNERAIS, A. PIQUEREAU, R. MAMPEY, F. TEICHTEIL-KÖNIGSBUCH - *Ressac: Flying an Autonomous Helicopter in a Non-Cooperative Uncertain World*. Proc. of the AHS Specialist Meeting on Unmanned Rotorcraft, 2007.

[23] M. FOX, D. LONG - *PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains.* Journal of Artificial Intelligence Research, 20 (2003), pp. 61–124.

[24] C. GARCIA, D. PRETT, M. MORARI - *Model Predictive Control: Theory and Practice*. A Survey, Automatica, 25 (1989), pp. 335–348.

[25] M. GAREY, D. JOHNSON - *Computers and Intractability : A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, 1979.

[26] A. GEREVINI, A. SAETTI, I. SERINA - *Planning through Stochastic Local Search and Temporal Action Graphs in LPG.* Journal of Artificial Intelligence Research, 20 (2003), pp. 239–290.

[27] E. GEREVINI, P. HASLUM, D. LONG, A. SAETTI, Y. DIMOPOULOS - *Deterministic Planning in the Fifth International Competition: PDDL3 and Experimental Evaluation of the Planners.* Artificial Intelligence, 173 (2009), pp. 619–668.

[28] M. GHALLAB, D. NAU, P. TRAVERSO - *Automated Planning: Theory and Practice*. Morgan Kaufmann, 2004.

[29] F. GOLNARAGHI, B. KUO - *Automatic Control Systems*. John Wiley & Sons, 2009.

[30] R. GRASSET-BOURDEL, G. VERFAILLIE, A. FLIPO - *Building a Really Executable Plan for a Constellation of Agile Earth Observation Satellites*. Proc. of the 7th International Workshop on Planning and Scheduling for Space (IWPSS-11), Darmstadt, Germany, 2011.

[31] R. GRASSET-BOURDEL, G. VERFAILLIE, A. FLIPO - *Planning and Replanning for a Constellation of Agile Earth Observation Satellites.* Proc. of the ICAPS-11 Workshop on «Scheduling and Planning Applications» (SPARK-11), Freiburg, Germany, 2011.

[32] C. GUESTRIN, M. HAUSKRECHT, B. KVETON - *Solving Factored MDPs with Continuous and Discrete Variables*. Proc. of the 20th International Conference on Uncertainty in Artificial Intelligence (UAI-04), Banff, Canada, 2004.

[33] J. GUITTON, J.-L. FARGES - *Towards a Hybridization of Task and Motion Planning for Robotic Architectures.* Proc. of the IJCAI-09 Workshop on «Hybrid Control of Autonomous Systems», Pasadena, CA, USA, 2009.

[34] E. HANSEN, S. ZILBERSTEIN - *LAO*: A Heuristic Search Algorithm that Finds Solutions with Loops*. Artificial Intelligence, 129 (2001), pp. 35–62.

[35] P. HART, N. NILSSON, B. RAPHAEL - *A Formal Basis for the Heuristic Determination of Minimum Cost Paths*. IEEE Transactions on Systems Science and Cybernetics, 4 (1968), pp. 100–107.

[36] J. HOFFMANN, B. NEBEL - *The FF planning system: Fast Plan Generation through Heuristic Search*. Journal of Artificial Intelligence Research, 14 (2001), pp. 253–302.

[37] L. KAELBLING, M. LITTMAN, A. CASSANDRA - *Planning and Acting in Partially Observable Stochastic Domains*. Artificial Intelligence, 101 (1998), pp. 99–134.

[38] H. KAUTZ, B. SELMAN - *Planning as Satisfiability*. Proc. of the 10th European Conference on Artificial Intelligence (ECAI-92), Vienna, Austria, 1992, pp. 359–363.

[39] H. KELLERER, U. PFERSCHY, D. PISINGER - *Knapsack Problems*. Springer, 2004.

[40] K. KORB, A. NICHOLSON - *Bayesian Artificial Intelligence*. Chapman and Hall, 2004.

[41] R. KORF - *Real-Time Heuristic Search*. Artificial Intelligence, 42 (1990), pp. 189–211.

[42] S. LAVALLE - *Planning Algorithms*. Cambridge University Press, 2006.

[43] E. LAWLER, J. LENSTRA, A. R. KAN, D. SHMOYS, eds. - *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. John Wiley & Sons, 1985.

[44] M. LEMAÎTRE, G. VERFAILLIE - *Interaction between Reactive and Deliberative Tasks for on-line Decision-Making.* Proc. of the ICAPS-07 Workshop on «Planning and Plan Execution for Real-world Systems», Providence, RI, USA, 2007.

[45] M. LEMAÎTRE, G. VERFAILLIE, F. JOUHAUD, J.-M. LACHIVER, N. BATAILLE - *Selecting and Scheduling Observations of Agile Satellites*. Aerospace Science and Technology, 6 (2002), pp. 367–381.

[46] C. LESIRE - *Iterative Planning of Airport Ground Movements*. Proc. of the 4th International Conference on Research in Air Transportation (ICRAT-10), Budapest, Hungary, 2010, pp. 147–154.

[47] T. MITCHELL - *Machine Learning*. McGraw Hill, 1997.

[48] C. PRALET, G. VERFAILLIE - *Decision upon Observations and Data Downloads by an Autonomous Earth Surveillance Satellite.* Proc. of the 9th International Symposium on Artificial Intelligence, Robotics, and Automation for Space (i-SAIRAS-08), Los Angeles, CA, USA, 2008.

[49] C. PRALET, G. VERFAILLIE - *Using Constraint Networks on Timelines to Model and Solve Planning and Scheduling Problems*. in Proc. of the 18th International Conference on Automated Planning and Scheduling (ICAPS-08), Sydney, Australia, 2008, pp. 272–279.

[50] C. PRALET, G. VERFAILLIE - *Forward Constraint-based Algorithms for Anytime Planning*. Proc. of the 19th International Conference on Automated Planning and Scheduling (ICAPS-09), Thessaloniki, Greece, 2009.

[51] C. PRALET, G. VERFAILLIE, M. LEMAÎTRE, G. INFANTES - *Constraint-based Controller Synthesis in Non-deterministic and Partially Observable Domains*. Proc. of the 19th European Conference on Artificial Intelligence (ECAI-10), Lisbon, Portugal, 2010, pp. 681–686.

[52] M. PUTERMAN - *Markov Decision Processes, Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 1994.

[53] P. RAMADGE, W. WONHAM - *The Control of Discrete Event Systems.* Proc. of the IEEE, 77 (1989), pp. 81–98.

[54] F. ROSSI, P. V. BEEK, T. WALSH, eds. - *Handbook of Constraint Programming*. Elsevier, 2006.

[55] T. SCHIEX, H. FARGIER, G. VERFAILLIE - *Valued Constraint Satisfaction Problems : Hard and Easy Problems.* in Proc. of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95), Montréal, Canada, 1995, pp. 631–637.

[56] P. SCHMIDT, F. TEICHTEIL-KÖNIGSBUCH, P. FABIANI - *Taking Advantage of Domain Knowledge in Optimal Hierarchical Deepening Search Planning*. Proc. of the ICAPS-11 Workshop on «Knowledge Engineering for Planning and Scheduling» (KEPS-11), Freiburg, Germany, 2011.

[57] R. SUTTON, A. BARTO - *Reinforcement Learning*. MIT Press, 1998.

[58] F. TEICHTEIL-KÖNIGSBUCH, G. INFANTES - *Anytime Planning in Hybrid Domains using Regression, Forward Sampling and Local Backups.* Proc. of the the 4th ICAPS Workshop on «Planning and Plan Execution for Real-World Systems» (ICAPS-09), Thessaloniki, Greece, 2009.

[59] F. TEICHTEIL-KONIGSBUCH, U. KUTER, G. INFANTES - *Incremental Plan Aggregation for Generating Policies in MDPs*. Proc. of the 9th Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-10), Toronto, Canada, 2010.

[60] F. TEICHTEIL-KÖNIGSBUCH, C. LESIRE, G. INFANTES - *A Generic Framework for Anytime Execution-driven Planning in Robotics.* Proc. of the IEEE International Conference on Robotics and Automation (ICRA 2011), Shanghai, China, 2011.

[61] F. TEICHTEIL-KÖNIGSBUCH, V. VIDAL, G. INFANTES - *Extending Classical Planning Heuristics to Probabilistic Planning with Dead-Ends*. Proc. of the 25th National Conference on Artificial Intelligence (AAAI-11), San Francisco, CA, USA, 2011, pp. 1017–1022.

[62] C. TESSIER, F. DEHAIS - *Authority Management and Conflict Solving in Human-Machine Systems*. Aerospace Lab, issue 4, AL04-08 (2012).

[63] S. THIÉBAUX, C. GRETTON, J. SLANEY, D. PRICE, F. KABANZA - *Decision-Theoretic Planning with non-Markovian Rewards*. Journal of Artificial Intelligence Research, 25 (2006), pp. 17–74.

[64] P. VAN BEEK, X. CHEN - *CPlan: A Constraint Programming Approach to Planning.* Proc. of the 16th National Conference on Artificial Intelligence (AAAI-99), Orlando, FL, USA, 1999, pp. 585–590.

[65] G. VERFAILLIE, G. INFANTES, M. LEMAÎTRE, N. THÉRET, T. NATOLOT - *On-Board Decision-Making on Data Downloads*. Proc. of the 7th International Workshop on Planning and Scheduling for Space (IWPSS-11), Darmstadt, Germany, 2011.

[66] G. VERFAILLIE, C. PRALET - *Constraint Programming for Controller Synthesis.* Proc. of the 17th International Conference on Principles and Practice of Constraint Programming (CP-11), Perugia, Italy, 2011, pp. 100–114.

[67] G. VERFAILLIE, C. PRALET, M. LEMAÎTRE - *How to Model Planning and Scheduling Problems using Timelines.* The Knowledge Engineering Review, 25 (2010), pp. 319–336.

[68] V. VIDAL - *A Lookahead Strategy for Heuristic Search Planning*. Proc. of the 14th International Conference on Automated Planning and Scheduling (ICAPS-04), Whistler, Canada, 2004, pp. 150–159.

[69] V. VIDAL, H. GEFFNER - *Branching and Pruning: an Optimal Temporal POCL Planner Based on Constraint Programming*. Artificial Intelligence, 170 (2006), pp. 298–335.

[70] M. YOKOO, E. DURFEE, T. ISHIDA, K. KUWABARA - *The Distributed Constraint Satisfaction Problem: Formalization and Algorithms*. IEEE Transactions on Knowledge and Data Engineering, 10 (1998), pp. 673–685.

[71] N. ZHANG, W. ZHANG - *Fast Value Iteration for Goal-Directed Markov Decision Processes.* Proc. of the 13th International Conference on Uncertainty in Artificial Intelligence (UAI-97), Providence, RI, USA, 1997, pp. 489–494.

[72] S. ZILBERSTEIN - *Using Anytime Algorithms in Intelligent Systems.* AI Magazine, 17 (1996), pp. 73–83.

Gérard Verfaillie graduated from École Polytechnique (Paris) in 1971 and from SUPAERO (French national engineering school in aeronautics and space, Computer science specialization, Toulouse) in 1985, Gérard Verfaillie is now research supervisor at Onera (The French Aerospace Lab). His research activity is related to models, methods, and tools for combinatorial optimization and constrained optimization, especially for planning and decision-making.

Cédric Pralet graduated from SUPAERO (French engineering school in aeronautics and space) in 2003. He passed his PhD in Computer Science in Toulouse in 2006. He is now working as a research engineer at Onera. His research interests concern constraint-based combinatorial optimization and automated planning and scheduling. His research activities are applied to space and aeronautics.

Vincent Vidal graduated from the University of Toulouse where in 2001 he defended a PhD in Computer Science in the field of Artificial Intelligence. After being an associate professor at the University of Artois from 2003 to 2009, he is now a research scientist at Onera. His research field is mainly focused on automated planning and scheduling, artificial intelligence, constraint programming and distributed algorithms.

Florent Teichteil holds an engineering degree and a master of science in control theory, as well as a PhD in artificial intelligence planning. He has been working at Onera since 2002 on the design of mission controllers for autonomous UAVs under uncertainty, and since 2009 on the quantitative assessment of critical probabilistic aeronautical systems. He is globally interested in the combinatorial optimization and the analysis of complex probabilistic dynamic systems.

Guillaume Infantes graduated from ENSEEIHT (a French national engineering school, computer science and applied mathematics specialization) in 2002, he obtained his PhD from the University of Toulouse in 2006 in the field of automated learning and decision making for autonomous robots. After one year as a research assistant at UMIACS (University of Maryland, Institute for Advanced Computer Studies), he joined Onera in 2008. His research activities are related to decision making under uncertainty, from modelling to problem solving algorithms.

Charles Lesire has graduated from ENAC (the French Aviation University, in Toulouse) in 2003 and obtained a PhD in Computer Science at the University of Toulouse in 2006. Since 2007, Charles has been a research scientist at Onera. His research activities are related to autonomous decision-making and software engineering for robotics.