A Cognitive Engineering Approach for Showing Feasibility Margins on an In-Flight Planning

Sami Lini

Bruno Vallespir

Sylvain Hourlier

Franck Labat

Pierre-Alexandre Favier

A COGNITIVE ENGINEERING APPROACH FOR SHOWING FEASIBILITY MARGINS ON AN IN-FLIGHT PLANNING

Sami Lini[1,2,4]
Bruno Vallespir[1,3]
Sylvain Hourlier[4]
Franck Labat[1,2,4]
Pierre-Alexandre Favier[2,4,5]

[1]Univ. Bordeaux, IMS, UMR 5218, F-33400 Talence, France
[2]CNRS, IMS, UMR 5218, CIH/ISCC team, F-33000 Bordeaux, France
[3]CNRS, IMS, UMR 5218, LAPS, F-33400 Talence, France
[4]HEAL (Thales Avionics/ENSC), F-33000 Bordeaux, France
[5]Univ. Bordeaux, IPB, ENSC, F-33000 Bordeaux, France

The purpose of the ASAP (Anticipation Support for Aeronautical Planning) project was to design an anticipation support for civilian pilots. In this context, we undertook a cognitive engineering approach. Interviews with pilots and in-situ tasks analysis were performed. Helping pilots better anticipate can consist in showing them the room for maneuver for every single task to perform. At first an activity modeling serves as a basis for describing these tasks during a specific phase of the flight (descent/approach). It confirms a need for a visual representation of temporal feasibility margins. According to the constraints of the flight plan (e.g. speed, altitude…) our algorithm dynamically computes the local extreme values for every main flight variables. The tasks performed in this so-defined multivariate tunnel are guaranteed to meet the flight path requirements. The design process and the canvas of our algorithm are presented in this paper. Directions are discussed to evaluate such an algorithm.

The purpose of the ASAP (Anticipation Support for Aeronautical Planning) project was to design an anticipation support for civilian pilots. In this context, we undertook a cognitive engineering approach. Interviews with pilots and in-situ tasks analysis were performed. That led to a model of anticipation which is described as a metacognitive process aiming at providing more temporal and resources margins to perform tasks. On that basis, it is possible to help pilots better anticipate by showing them the room for maneuver for every single task to perform as well as tasks dependencies.

At first an activity modeling serves as a basis for describing the various tasks to be performed during a specific phase of the flight (descent/approach). That model highlights how constrained are these tasks along the main defining flight variables (time, altitude, speed…) and how pilots tackle these constraints. It also confirms a need for a visual representation of feasibility margins (see Lini et al., 2013).

The elaborated tasks graph is used by implementing our algorithm. According to the constraints of the flight plan (e.g. speed, position, altitude…) it dynamically computes the local extreme values for every main flight variables. The tasks performed in this so-defined multivariate tunnel are guaranteed to meet the flight path requirements. A demonstrator of the functioning of this algorithm is implemented. It simulates some of the flight variables

computation and shows how the tasks graph is accordingly modified dynamically all along the flight phase.

The design process and the canvas of the algorithm are presented in this paper. Directions are discussed to evaluate such an algorithm.

Activity modeling

For logistic reasons (data availability), we choose to study the descent and approach phases on Rio de Janeiro international airport. At first, an interview is conducted with a flight instructor familiar with this flight. The pilot is required to put into words all the tasks performed in the defined phases. Using the MASK knowledge management methodology (Matta, Ermine, Aubertin, & Trivin, 2002) we conducted a hierarchical task analysis (HTA, Stanton, 2006) aiming at building a knowledge base.

At first, we capitalized the knowledge of an expert pilot on a defined flight phase: the descent and approach phases of a commercial flight are known to be representative of the troubles pilots have to face (Boeing, 2011). This capitalization consists in a recorded semi-directed interview where the subject is asked to describe as accurately as possible his tasks and activity. Questions are asked from time to time to disambiguate grey areas. A camera video recording of the cockpit was also made during an actual flight phase with the expert.

These two sources of information form the material we have been using for building the task diagram. We stressed the inputs and outputs of every single task and their links to anticipation. This allows building a representation of pilots' activity over time. A second model, the activity diagram, gives a higher level of details about it: processes and used resources are detailed. We interviewed a second expert to double check the modeling. Pursuing the goal of designing an anticipation support, both the HTA and pilots-in-the-loop process led to several trends to help better anticipate.

The first one is about helping to compute the feasibility margins (in terms of time and other flight variables) of the most critical tasks to be performed. Two reasonings are considered. The first one is as follows: "given an arrival airport and an ETA, until what time can I consider performing every task in the graph while flying safely and satisfying all the coming constraints". The second one is as follows: "given the current situation, what is the earliest term until which I can consider performing the tasks while still satisfying all the coming constraints".

Another trend is to help dealing with "what-if": to help implementing hypotheses about the flight, in case of diverting for instance. All of this will necessitate a real-time and up-to-date representation of the situation or the ability to dynamically build and represent a projection of an alternative scenario.

Planning algorithm

Objectives

The objective of the presented algorithm is the following: to offer the possibility to visualize and interact with an improved flight plan (predefined but not necessarily specific to a particular flight) which is dynamically synchronized with the current flight situation (position, altitude, speed).

The requirements are as follow:

- Representation of tasks feasibility margins

- To specify a flight plan to a particular airport;
- Automatic spatial and temporal adjustment (ahead/delay regarding an ETA given the current situation)
- What if (simulation and validation)
- Dynamic re-planning in case of diversion

Data structure

The first step of this algorithm consists in modeling a flight plan: elements we want to show and on which computation will be performed. A *flight plan* is an oriented graph in which *flight steps* are vertices while the *change of state* between two flight steps are edges. This plan is built during a two steps process. Firstly it's instantiated from a generic flight plan (built from our HTA previously described) which defines generic procedures for a standard approach. It's then specified through a transformation process which exploits both this generic instance and a constraint list specific to the concerned airport in order to produce a dedicated flight plan specific to the current flight...

Each flight step is defined by:

- a name: it describes the task to be performed;
- the aerodynamic state of the aircraft: it is a mandatory property in order to take into account the plane's shape in the time margins computation. What is the flaps' level? Is the landing gear in or out? Knowing this information will allow adjusting the maximum and minimum plane's evolution speed in the computation of time margins.

Thus the structure of the flight plan describes everything that needs to be performed during the flight and the "shape" of the plane during these very tasks.

A list of constraints can be tied to a flight step. A *constraint* can be either generic (shared between all flight plans) or specific to a flight plan. Managing a flight is not specific to a particular trajectory. Generic rules are used to fly the plane. Like a mental representation, these general rules need to be fulfilled with local and specific information. We then need to distinguish what is generic to any flight and what is specific to a trajectory.

A constraint is defined by:

- a name: it describes the constraint,
- a set of variables (altitude, position, speed, …): depending on the source of this constraint, one or more flight variable can be constrained. The aerodynamic state of the plane will for instance give speed constraints. On the flight path, one might also encounter other speed limitations as well as altitude or position constraints.
- a range of values to reach for each variable: the variable can be constrained to either a single value (pretty scarce) or to a range of values (for instance, speed between 250 kts and 280 kts).

Two kinds of constraints exist:

- a *postrequisite* is a satisfied constraint, the new state of the world, after performing the task, which is necessary in order to define if it is possible or not to move to a next step
- a *prerequisite* is an entry constraint without which the flight step cannot be considered.

Tied to a flight step are a list of its *predecessors* (flight steps) and a list of its *successors* (flight steps). These predecessors and successors allow from every vertex to have a representation of both where it came from and where it goes.

An *arc* is defined as a pair (*p*,*q*) where *p* and *q* are two distinct vertices. A *transition* makes possible to move from a vertex *p* to a vertex *q* if and only if the constraints list of *p* is satisfied and there is an arc between *p* and *q*. Two kinds of arcs are defined:

- a *forward arc* model the progress of the flight plan
- a *backward arc* model the updates of constraints according to the flight plan (aerodynamic state) and the pilot's anticipation.

Given a pair (*p*,*q*), the activation of *p* towards *q* consists in the following steps:

1. *p* is the active flight step
2. Postrequisites are validated: constraints satisfaction is checked
3. If there is an arc between *q* and *p*, *q*'s prerequisites are checked.
4. If *q*'s prerequisites are satisfied, *p* is inactivated and *q* is activated.

Spread algorithms

Two spread algorithms are distinguished:

- The forward spread is used every time the current state of the aircraft is modified. It can also be used in the case of a change of the execution time of a flight step (anticipation).
- The backward spread is always used after a forward spread in order to spread to every flight step the new calculated time limits.

For each vertex, the updated values for each variable are thrown to all of its predecessors/successors (depending on the direction of the spread). The Breadth First Search (BFS) algorithm is used to perform this spread throughout the whole plan.

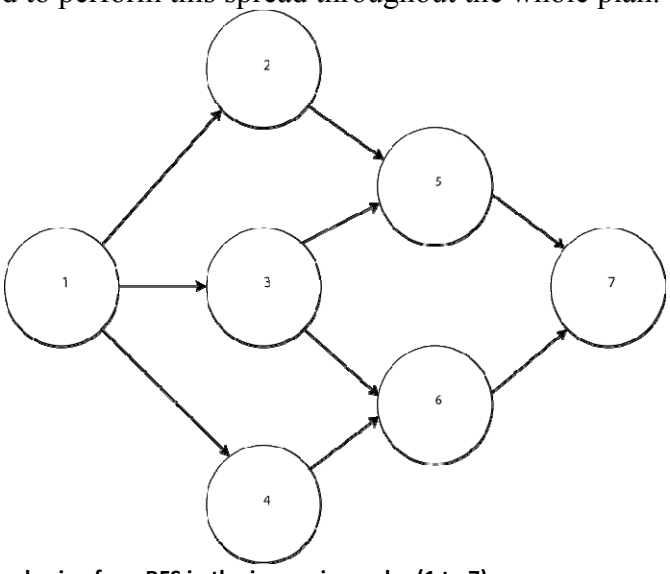

Figure 1 : Steps numbering for a BFS in the increasing order (1 to 7)

The forward spread then consists in:

- A BFS to all forward arcs from an activated vertex

- For each successor, an update operation of all variables using the MaxMin algorithm (see below)

The backward spread consists in:

- A BFS to all backward arcs from an activated vertex
- For each predecessor, an update operation of all variables using the MinMax algorithm (see below)

MaxMin and MinMax algorithms.

The philosophy of the MinMax algorithm is the following: given a vertex of which we know every single characteristic (temporal, spatial, and aerodynamic), we observe each of its predecessors. Every predecessor has one or more constraints.

These constraints are successively and hierarchically analyzed. They are considered in the following order of importance: altitude, speed, vertical speed, position (from HTA). The time to satisfy each of them is calculated using a simplified model of evolution: a constraint defined a state of the world (speed, position…).

We calculate the time it takes to move from this state of the world to the state of its successor. Following the adage "*he who can do more can do less*", we keep the lowest calculated value. The algorithm is then spread to the predecessors using the BFS.

The MaxMin algorithm follows the same philosophy in a forward way.

The actual tasks graph is the following:

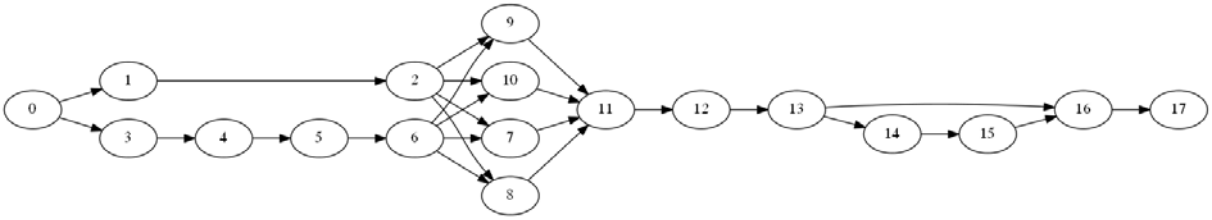

Figure 2: Actual tasks graph

Prototyping

In order to test both the spread equations and the spread models, a prototype HMI was designed. Three panels were defined: a temporal one, presenting all tasks and their time margins, a speed panel, presenting the speed margins in time for each task, and an altitude panel presenting the altitude range for each task.

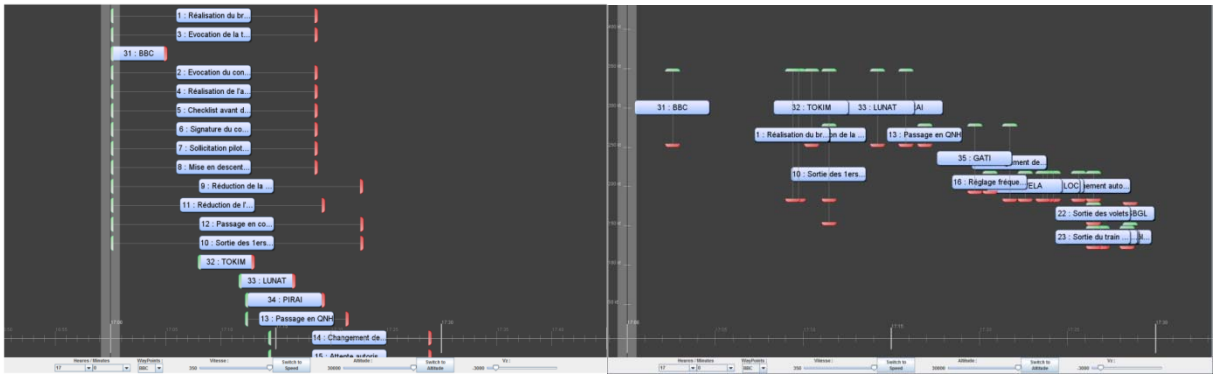

Figure 3: Tasks and speed panels

Conclusion

This algorithm allows displaying a visualization of both the flight path and the tasks to be performed in an inter-linked and up-to-date way. It has been implemented in the ASAP interface which has been evaluated with a panel of 36 commercial pilots in a simulated flight context and showed benefits on both cognitive load and situation awareness (see Lini et al., 2013).

Further improvements of this algorithm shall focus on the plane's energy rather than its independent flight variables.

References

Boeing. (2011). *Statistical summary of commercial jet airplane accidents - Worldwide operations 1959 2011*.

Lini, S., Bey, C., Hourlier, S., Vallespir, B., Johnston, A., & Favier, P.-A. (2013). Evaluating ASAP (Anticipation Support for Aeronautical Planning): a user-centered case study. *Proceedings of the 17th International Symposium on Aviation Psychology*. Dayton, OH.

Matta, N., Ermine, J. L., Aubertin, G., & Trivin, J. Y. (2002). Knowledge Capitalization with a knowledge engineering approach: the MASK method. *Knowledge management and organizational memories*, 17–28.

Stanton, N. A. (2006). Hierarchical task analysis: developments, applications, and extensions. *Applied ergonomics*, *37*(1), 55–79. doi:10.1016/j.apergo.2005.06.003