AN ADAPTATIVE MULTI-AGENT MODEL OF DELAYS PROPAGATION IN THE AIR TRAFFIC SYSTEM

Georges Mykoniatis, Laurent Lapasset, Andrija Vidosavljevic

AN ADAPTATIVE MULTI-AGENT MODEL OF DELAYS PROPAGATION IN THE AIR TRAFFIC SYSTEM

First A. Georges Mykoniatis, Second B. Laurent lapasset, and Third C. Andrija Vidosavljevic

Ecole Nationale de l'Aviation Civile

Keywords: Delay Propagation, hub-and-spoke airline modeling, adaptive multi-agent modeling,

Abstract. In the air transportation system, airports are connected by companies operating flights in different ways. One is to use a shuttle system, with a high number of evenly scheduled daily flights. In such a case, a delay occurring will be smoothed out using the buffer effect of the remaining rotating flights. On the other hand, companies operating a single rotation will only be able to mitigate the delay using the turnaround duration. It is the same for transit flights. In existing simulation environments, such a fine structure is not taken into account, even if some tools were designed in the context of rail transportation. The purpose of the present work is to provide the air transportation community with an advanced delay propagation simulator using an adaptive multi-agent model. The main contribution is the introduction of a learning procedure mimicking the airlines behavior as a response to a delay. The mitigation effects will thus be made dependent on the rotation typology on the various legs[1] of the network and will adapt according to the outcomes of a decision. Furthermore, a priori knowledge about a specific airline behavior (low cost, regular) may be encoded in the learning algorithm to closely adhere to reality.

1. INTRODUCTION

In the commercial sector of Air Transportation System, delays are of the great concern as they generate disruptions and costs for the airlines, ANSPs, airport operators, ground handlers and passengers. Delay of a flight may cause delay of consecutive flight that is operated by the same aircraft and, in the case of hub airport, delay of connected flights due to the late transfer passengers. In 2016, around 2 million flights (20%) have been delayed in the ECAC area, causing increase by more than 1 milliard euros of direct operating costs to the airlines [1]. To cope with this problem, airlines pad their flight schedules including extra time between two flights. However this increase airline's direct cost since flight crew is being paid for more time than a flight actually takes, and due to lower aircraft exploitation. In the similar way, uncertainty of flight schedules due to the delays cause inefficient use of the sparse resources such as airspace and airports, which already operating at their limits. In order to decrease delays, efficiently plan operating schedules and measures for delay mitigation, it is important to identify delay origins but also understand mechanisms of their formation and propagation.
Study of ATC delays, delays caused by ATFM regulations in the case of airspace capacity shortage, have been done extensively, however studies of non-ATC delays are mainly data-driven without complete understanding of delay propagation due to the lack of airline actions as a respond to initial delays.
The current models and simulation tools used in the air traffic management domain are not able to identify clearly what is the result or impact of a delay on the network efficiency. This situation is clearly due to the lack of information on the mode of connections between the various legs of the flights. It is only in the last years that the flight plan information sent to the network manager shall contain a registration number, allowing to the network manager to identify which aircraft is used on which leg and to correlate the information. Since then a set of studies allows us to get a better picture of the delay propagation mechanism. In early 2017, S. Belkoura, JM Peña, and M. Zanin [2], shown that analyzing a large set of flights operating over the European airspace, that airports can be classified into two groups: those in which outbound delays get independent of inbound ones, and those in which large inbound delays are propagated, on average, under the form of large outbound delays. These considerations has been motivation for the more detailed analysis of such behaviors based on airline predominant network design, and the mechanisms for delay compensation and reduction. The objective of this paper is to present the initiative to provide to the air transport community with an advanced delay propagation simulator using an adaptive multi-agent model. We will start by giving some useful definitions and setting the scene in term of airline network and types of operations, and then we will introduce the multi agent approach in how it will be used in our context. We will then describe the State estimation and adaptive decision rules that has been used to set up our model and simulations.

[1] A flight leg is basically flight from one point to another point, a flight plan could have several legs

2. DELAY DEFINTION AND PROPAGATION

By the most general definition, “delay is the time lapse which occurs when a planned event does not happen at the planned time” [3]. However the way and the moment when delay is measured depend on the context. For an airline schedule departure or arrival delay is how late a flight departs or arrives compared to an airline’s schedule, and it is measured on the ramp. Delays could be introduce during the various phase of operations, during the taxiing, while airborne, en-route, during arrival sequencing or departure sequencing, and recording delay is not always an easy task. Delay propagates throughout the Air Traffic Management network because of the interdependencies between different schedule flights. The interdependency between flights is introduced if: performed successively using same resources such as aircraft and crews, sharing transfer passengers or sharing airspace/airport resources. For example, a late arrival of one flight may cause a late departure of the consecutive flight, introduce a delay in the schedule sequence of arrivals at a given airport or delay a next flight using same apron. Therefore the main delay categorization is into concepts of Primary and Secondary delay. A primary delay is delay that affects the initiation of the flight and it is unaffected by any earlier event. However, a secondary or reactionary delay is accumulated, and is imposed as a consequence of the unavailability of aircraft, crew or passengers due to disruption earlier in the day. The cause of this earlier disruption could be itself either a primary delay at the start of the previous flight, or a reactionary delay arising from an even earlier incident.
The most common strategy for the delay prevention is to build schedules more robust to the delays by padding a flight schedules including extra time between two flights. However, when experiencing a delay airlines take measures for the delay mitigation. Delay may be recovered while aircraft is airborne or during turn-around process. Each of the mentioned strategies have their costs, and to choose optimal solution to initial problem it is important to have a knowledge of delay effect and its propagation. Due to various causes of flight interdependencies and a network it is very complex task to model delay propagation. For the modelling of the delay propagation an Approximate Network Delays model (AND) has been described [4] by a stochastic and dynamic queuing model designed to compute approximately delays at each of the individual airports in a network and, more important, how these delays propagate from one airport to another over the course of a day or other time period of interest. It treats the airports in the network as a set of interconnected individual queuing systems. To be able to measure the propagation of the delay the notion of **effective Delay multiplier** (EDM)has been introduced by Felix Mora-Camino in early 2017 [5]. In our work we will use the EFD [5] not use the AND [4] approach. We will consider the rotation time for an aircraft at a given airport. This rotation time will be function of the type of aircraft, of the availability of the various resources needed for the rotation at the airport..

3. NETWORK

For the network modelling, we will start by using what have been done in 2007 by M. Alderighi, A. Centro P. Nukamp and P. Rietveld [6]: “There is no unique or even widely used definition of what exactly constitutes an HS (Hub and Spoke) or a PP (point to point) network. Instead, a number of definitions coexist. From a network design perspective the HS or PP network can be described by using a simple network of four nodes:”

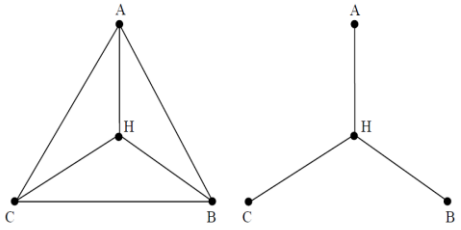

Figure 1. Point-to-point network versus a hub-and-spoke network

In their paper they found some evidence that the FSCs (Full-Service Carrier) have developed their networks as mixed multi-HS and PP systems with a strong dominance of the HS. These configurations vary from Iberia, which has the most spatially concentrated HS network with a two-hub radial network (Barcelona and Madrid), to British Airways, which has the most mixed HS and PP network configuration. In particular, British Airways network is organized such that London-Heathrow is the main hub, and Manchester, Glasgow and Edinburgh are bases with several direct connections to European and domestic destinations. The Lufthansa network developed into a two-HS with mixed PP structure. In particular, the hubs are Munich and Frankfurt; and the bases with PP connections are Berlin, Hamburg and Dusseldorf. Finally, the Air France network (before the KLM merger) is classified as a single HS configuration with Charles de Gaulle as the hub for intra- European and intercontinental traffic, and Paris-Orly acting as a PP airport base for domestic traffic within France. Their analysis shows variations among

LCCs (Low Cost Carriers) network configurations. While Ryanair and EasyJet have developed a pure PP structure, Virgin Express and Air Berlin offer a modest percentage of connecting flights in Brussels and Berlin. However, Virgin's connectivity ratio has grown in the last few years, and it is possible that the bases of this LCC can turn into small hubs if this trend continues in the years to come. In this work we will classified Airlines by Type (FSC, LCC) and we will modelled their network by a set of legs. This modelling will allow to define different Network configurations.

4. MULTI-AGENT SYSTEMS

Multi-agent systems have been used to solve many problems in operations research, like regulation of urban transportation networks [7] design of mechanical systems [8], or path-finding problems [9].This paradigm is often regarded as a kind of distributed artificial intelligence. Multi-agent systems are made of autonomous agents interacting among themselves and with their environment [10]. Usually, agents have a limited perception of environment and they partially know the internal state of their neighbors, via message exchanges.

Their behavior can either be simple (whereby reactive agents are only influenced by environmental changes) or complex (whereby cognitive agents try to fulfill an objective).

Self-organization is a key aspect of multi-agent systems. If the rules that direct agents are carefully chosen, a complex behavior can emerge at the system level from local interactions and behavior of agents. When multi-agent systems are used to solve operations research problems, a carefully chosen set of agent behaviors can help to find an overall solution to the problem (system level) by only using local rules (agents level).

Multi-agent systems can be implemented either within a computer simulation, or as a physical system that is composed of robots that are able to communicate and to interact with their environment. When agents are implemented within a computer simulation, computations of agents can be done in parallel, exploiting modern hardware architectures (multi- core processors, computations on graphic card). A multi-agent system can also run on a cluster of computers. Those systems have several advantages compared to centralized decision methods. When correctly designed, they exhibit a good resilience when facing disruptive events [11]. Agents try to fulfill a goal and act in order to become closer to this objective. When they are confronted to local perturbations in their environment, they adapt their actions to take those changes into account, enabling the system to get back to a new stable state. Since decisions are decentralized at agent's level, the failure of an agent will not impact the whole system. In centralized decision processes, a central regulation entity failure may prevent the system to work. In the field of information technologies, such a central point would be defined as a Single Point of Failure (SPOF).

4.1. AMAS Approach

In the AMAS theory (Adaptive Multi-Agent Systems) [12], agents are considered as autonomous and cooperative entities, having a partial knowledge on their environment and searching to reach a local objective. Agents of these systems interact locally in a cooperative manner producing partial functions. Cooperation is defined as the capacity of the agents to work together in order to reach a common objective. Thus, any activity between agents is complementary and solidarity links exist between them. Using cooperation, the system self-adapts to stay in cooperative state. The cooperation of all parts of the system makes the adequate function the system was designed for "*emerges"*.

Local interactions allow the system to self-adapt to perturbations and so to handle dynamics without challenging the already reached solution. Perturbations produce "**Non Cooperative Situations**". To repair those situations, agents possess mechanisms to autonomously adapt their behavior to the context [12]:

- **Tuning**: the agent adjusts its internal state to modify its behavior,
- **Reorganization**: the agent modifies the way it interacts with its neighborhoods,
- **Evolution**: the agent can create other agents or self-suppress when there is no other agent to produce a functionality or when a functionality is useless.

The algorithm of an adaptive agent can be described as follow: if a Non Cooperative Situation is detected, agent uses one or more self-adaptation mechanisms to come back to a cooperative state where it performs its nominal behavior.

To ease the design of agents' behavior and interactions for solving optimization problems under constraints based on the AMAS Theory, the **AMAS4Opt** agent model has been proposed [13]. This model provides design patterns for two cooperative agent roles: "**constrained role**" and "**service role**". One agent can have one or both roles and switches at runtime between them depending on the situation it faces. The agents having the "constrained role" manage the constraints and must be satisfied, while the agents having the "service role" are skilled to help the

agents under the "constrained role". This model uses the notion of so called criticality of agents with the "constrained role" as an engine for the cooperation between agents. We have used and extended this model to design the agents and the general architecture of our system.

4.2. Applied to air traffic management

In the Adaptive Multi-Agent System Theory [12] agents are defined as autonomous, adaptive and cooperative entities that possess local objective. The local cooperation between agents allows the system to self-adapt in order to realize the function for which it is designed. Cooperation is defined as the ability agents have to work together in order to realize a common global goal. It implies that the activities of the agents are supplementary, and dependency links and solidarity exist between them. To deal with dynamic environments, agents possess mechanisms enabling them to autonomously modify their organization. In order to identify the agents of the ATLAS system, we followed the ADELFE toolkit (French acronym for "Toolkit to develop software with emergent functionality" - Atelier de Developpement de Logiciels à Fonctionnalité Emergente).
ADELFE allows to develop software with emergent functionality [12]. The goal of this methodology is to guide the development of adaptive multi-agent systems, through several work definitions, from preliminary requirements to design and fast prototyping. Thus, ADELFE allows to qualify the environment of the system, the exchanged messages and to identify the system entities.
To ease the design of agent's behavior and interactions for solving optimization problems based on the AMAS Theory, the AMAS4Opt agent model has been proposed [12]. This model provides design patterns for two cooperative agent roles: "constrained role" and "service role". One agent can have one or both roles and switches at runtime between them depending on the situation it faces. The agents having the "constrained role" manages the constraints and must be satisfied, while the agents having the "service role" are skilled to help the agents under the "constrained role". This model uses the notion of so-called criticality of agents with the "constrained role" as an engine for the cooperation between agents. We have used and extended this model to design the agents and the general architecture of our system. In this section, the identified agents, their behavior, interactions and the criticality measure of agent under "constrained role" are detailed. Then, the extension of the AMAS4Opt model, the cost measure, representing the criticality of agents under the "service role" is presented.

4.3. Agents

Given the problem description and using the ADELFE toolkit, we identified three kinds of entities:

- Cooperative agent's types: the aircraft agent, the airline agent, the network manager…
- Active entities agent's types: the meteorological coverage and passenger demand. These entities do not have a goal to satisfy, but they still interact and influence agent activities and decisions,
- Passive entities agent's types: the airport agent, aircraft agent, leg agent .These entities are considered passive because they do not interact with other entities, they are resources.

The AMAS4Opt model allows to design the behavior and interactions between agents. In our case and given the AMAS4Opt agent roles definition, the aircraft, airport and airline agents play the "service role", and the network manager agents and meteorological agents play the "constrained role".
The second essential element of our agents is the **set of links** between it and others local's agents. We will discuss later this notion of locality. At this point, we would like to focus on the necessity to maintain a coherent set of links between agents during all the simulation time. A link is a direct connection between two agents, by which both can exchange information related to their internal state. For example, when an aircraft need to takeoff, he knows its delay from the departure airport. So, in the lifecycle presented in Figure 2: Multi-agent system lifecycle, when the agent aircraft is activating, it exchanges delay messages with its airport. For example, aircraft agent's set of links contains a link between airport departure and the aircraft.

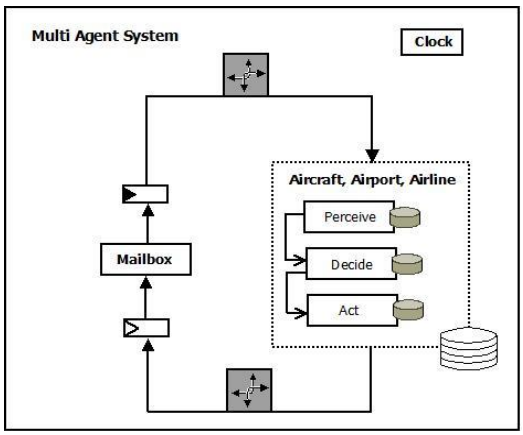

Figure 2: Multi-agent system lifecycle

4.3.1. Agents lifecycle

In our system and based on the agent behavior definitions given by **AMAS4Opt**, the propagation of the delay is provided by message passing between its agents. The life cycle of each agent is decomposed into three steps: "Perceive - Decide - Act". During phases of perception and action, agents receive and send messages. The phase of decision is the key step. Indeed, according to its perception and its state, the agent chooses which action it has to perform. The cycle "Perceive - Decide - Act" is repeated until the system provides a solution. As the agents only have partial perceptions, they do not know if a global solution is reached. In order to detect that a good and coherent solution is obtained, and stop the agents, we introduce an Observer: as its name suggests, its function is only to observe the evolution of the system. Whenever this one is stable: agents do not change their state, the Observer asks all the agents to stop and exposes their states. It is very important to note that the Observer has no interaction with agents during the solving, it only observes and stops them: it does not belong to the solving process.

5. STATE ESTIMATION AND ADAPTIVE DECISION RULES

5.1. Agent attributes and operations

In a simulation framework, a perfect knowledge of the situation is generally assumed, so that the state of each entity in the simulated world is known in advance. When robustness and adaptability are key ingredients of the modeled system, such an approach is not satisfying as it will not reproduce the way real actors will behave. In order to gain an extra level of realism in the framework introduced in this work, the knowledge of the internal states of the agents involved in the simulation will be partial and has to be estimated for the remote entities. The following assumptions are made to make the system amenable to mathematical modeling:

- The agents $A_i, i \in 1...n$ are vertices in a connectivity graph with one edge e_{ij} for each communication channel.
- Communications are perfectly reliable and do not suffer transmission delays.
- Each agent has a state, that is made of categorical attributes (e.g; "runway saturated", "normal operation") and continuous attributes (e.g; number of inbound flights in the next 30mn, number of taxiing aircraft).
- The knowledge of its own state is perfect for an agent, but only an indirect observation of it is made available to the others.
- Decision rules applied to mitigate delays are encoded as a decision tree that is learned during the simulation process. The internal nodes are based on literals that can be of the

5.2. Evolving the agent system

Following the general principles of multi agent modeling, that is collective behavior through communication, the delay simulator will implement the two classical phases of message passing and decision. However, since the environment is only known in a probabilistic fashion, an extra level of adaptation is needed. Furthermore, the successive phases in the lifecycle of an agent are assumed to be synchronous over all the system: an agent cannot enter a stage before all the others are ready to perform the same operation. It will not be a problem when the simulator is operated on a single compute node, even in a multithreading environment, but some care must be

taken when spreading over a cluster. In such a case, a clock message must be issued by a master node, so that all agents involved in the process know the current simulation stage and acknowledge in turn. Currently, such a case was not covered and the implementation is deferred to future work.
The process implemented to complete one iteration is given below:

- In the communication phase, each agent broadcast a set of performances indicators that partially reflects its own state.
- The state of neighboring agents is estimated, using a hidden markov model [14].
- The decision stage is then performed, using decision trees [15].
- A backward update step is performed, starting from the terminal nodes involved in the decision, and propagating errors.

5.3. State estimation for neighbors

To make the model amenable to statistical thinking, it is assumed that the state of neighbors is a categorical random variable. Since this assumption does not complies with the communication process described above, that makes use of performance indicators that are real variables, a preprocessing phase is applied to convert them to a single or a vector of categorical states. The rules applied to perform this operation are based on experts experience and are fixed during all the simulation. When trying to estimate the state of an agent, given only partial observations of it, it makes sense to assume that the state evolution is governed by a Markov chain with unknown transition matrix. It can be further generalized to k −markovian processes by gathered successive states into k −uples, so that most of usual situations can be handled that way. The classical Baum-Welch algorithm is used to learn the unknown parameters. Please note that it is a local optimization procedure (in fact an adaptation of the EM-algorithm to hidden Markov chains), so the initialization is critical in finding the correct parameters. Expert knowledge about true operational practices are thus used as a starting point for the transition matrix. For the sake of completeness, the Baum-Welch algorithm [16] is given below. The estimated state for agent *j* at iteration *i* will be denoted by X_i^j, while the estimated transition matrix will be M_i^j, with elements $M_i^j(lk) = P(X_{i+1} = k\,|X_i = l)$. Finally, the observations and the state are related by an estimated conditional probability matrix O_i^j with elements$O_i^j(lk) = P(Y_i = l|X_i = k)$.
The first step in the Baum-Welch algorithm is to estimate for all states *k* the probability of being in state *k* at iteration *i*. First, the probability of the chain of observations $Y_1, \dots, Y_i$ knowing that the current state is k can be obtained by forward propagation. Namely:
$P(Y_1 = l, X_1 = k) = P(Y_1 = l|X_1 = k)P(X_1 = k) = O_1^j(kl)P(X_1 = k)$
and recursively:
$P\big(Y_1 = l_1, \dots, Y_p = l_p, X_p = k\big) = P\big(Y_p = l_p\big|X_p = k\big)P\big(X_p = k, Y_1 = l_1, \dots, Y_{p-1} = l_{p-1}\big) =$
$O_p^j\,(kl_p)\sum_{i=1}^n M_p^j(ik)P(Y_1 = l_1, Y_{p-1} = l_{p-1}, X_{p-1} = i)$
The previous probability is generally denoted as $\alpha_p(k)$.
Learning matrix M_{i+1}^j and emission probabilities O_{i+1}^j is made using maximum likelihood estimation, that is in this case a simple counting estimator. Let $(l_1, \dots, l_Q)$ be a sequence of observations over a size Q time window (please note that due to the Markov property, it can be of arbitrary origin). The probability:
$\beta_k(i) = P\big(Y_{k+1} = l_{k+1}, \dots, Y_Q = l_Q\big|X_k = i\big)$
Can be estimated using a backward algorithm:

1. Initialization $\beta_Q(i) = 1$
2. Back-propagation $\beta_k\,(i) = \sum_{p=1}^n \beta_{k+1}\,(p)O(l_{k+1}p)M^j(ip)$

The probability of being in state *i* at time *k* knowing the observation sequence can be obtained as:
$\gamma_k(i) = \frac{\alpha_k(i)\beta\_k(i)}{\sum_{i=1}^n \alpha_k(i)\beta_k(i)}$
And finally, the joint probability of state *p* and state *l* at time *k* is computed by the formula:
$\xi_k(pl) = \alpha_k(p)M(pl)O(ll_{k+1})\beta_{k+1}(l)\lambda$ with λ a normalizing constant that ensure that all the ξ_k sum to 1.
The maximum likelihood estimator of the transition matrix is then:
$M(pl) = \frac{\sum_{k=1}^Q \xi_k(pl)}{\sum_{k=1}^Q \gamma_k(p)}$
Similarly, the observation probabilities are estimated by:

$O(pl) = \frac{\sum_{k=1}^{Q} 1_{l_k=l} \gamma_k(p)}{\sum_{k=1}^{Q} \gamma_k(p)}$

And the state probability vector as γ_1. The Baum-Welch algorithm is easily implemented in pure Java, with a minimal impact on the performance compared to numerical computing oriented languages like Fortran. Each agent maintains an hidden markov model for all its neighbors, the update being performed during the communication phase. Since the state estimations of the agents are independent, it is advisable to split the set of agents into equally sized batches, and to distribute the computation over all the available cores of the processor. In the case of clusters, the same process can be applied to speed up the computation. However, communication cost may become dominant if the size of agent pools present on each compute node is too low, so that care must be taken to use this kind of parallelism only on large simulation instances. In the current implementation, no cluster computation is available, but is planned for future releases.

5.4. Agent rules

The agents will compute their states synchronously during the update phase. The current state of the agent and the estimated ones of its neighbors are considered as attributes of a decision tree whose leaves are the updates states. While many algorithms exist for learning such structures, the special nature of the agent system requires some features:

- The tree learning must be adapted to data streams, since it must be performed during the simulation run.
- Some attributes are random variables, namely the states of the neighbors that are only estimated. The learning algorithm must thus be able to work with uncertain data.

To cope with these requirements, a novel learning procedure, the Dirichlet Decision Tree (DDT) is proposed. First, for a given agent, decision attributes can be split into categorical values, that are elements of a finite set, and continuous values that belong to an interval of the real numbers. To simplify the problem, only categorical values will be considered as estimated states of neighboring agents, while the current agent has access to continuous values that may be understood as key performance indicators (KPI). With this assumption, inner nodes can be equality testing of the form $at_i = v_j$ for attribute number *i* and value v_j when at_i is categorical or real inequality $at_i \leq v_j$ for continuous attributes. In this last case, at_i is not considered to be a random variable, but will be such in the former if it is estimated. The decision tree is initialized either from an expert advice or from an already build database of examples. Since there is no need to special treatment in this first stage, a very classical algorithm based on entropy is selected. The samples of the initial databases are denoted as $(x_1^i, \dots, x_N^i, s^i), i = 1 \dots P$ where for the sample number *i*, s^i is the state obtained as a response to the vector of attributes $x_1^i, \dots, x_N^i$. Please note that in this phase, no attribute is a random variable. The overall process is summarized below.

1. The entropy of the complete sample is computed as: $H = \sum_{j=1}^{n} p_j \log_2 p_j$, where p_j is the estimated probability of occurrence of state *j*, given by $p_j = N^{-1} \sum_{i=1}^{N} 1_{s^i=j}$
2. For each attribute *k*, the entropy of the sample relative to the attribute is defined to be $\sum_{l=1}^{Q_k} q_l \sum_{j=1}^{n} p_j^l \log_2 p_j^l$ where Q_k is the number of possible values for attribute *k*, $q_l = N^{-1} \sum_{i=1}^{N} 1_{x_k^i=l}$ and $p_j^l = \sum_{i=1}^{N} \frac{1_{s^i=j} 1_{x_k^i=l}}{N q_l}$
3. The attribute that maximizes the difference between the original entropy and the relative entropy will be selected at the root node.
4. The process is iterated down to leaves using at each stage the remaining part of the original sample that remains after application of the rules borne by the parent nodes.

It worth to note that estimating probabilities with counting ratio is a maximum likelihood procedure when the underlying probability distribution is multinomial. When using the output of hidden markov models , the state is no longer deterministic, but follows a probability distribution. In such a case, the procedure makes use of a Bayesian estimator, with a Dirichlet distribution as the conjugate prior. At each node, the probabilities p_j^l are assumed to be drawn from a Dirichlet probability law with unknown parameters $\alpha_1, \dots, \alpha_n$:

$P(p_1, \dots, p_n) = \frac{\Gamma(\sum_i \alpha_i)}{\prod_i \Gamma(\alpha_i)} \prod_i p_i^{a_i - 1}$

A sufficient statistic for the estimation is given by the sum $\sum_i \log p_i$, which can be update online. While the maximum likelihood estimator is obtained by solving a system of nonlinear equations, only a few fixed-point iterations is enough to get the solution up to machine precision. Once the vector of parameters $(\alpha_1, \dots, \alpha_n)$ has been obtained, the Bayes estimator for the probabilities $(p_1, \dots, p_n)$ assuming a Dirichlet prior distribution can be

obtained and replaces the counting estimator. As the prior distribution estimation requires some probabilities samples, a sliding window procedure is applied. The resulting algorithm can correctly estimate the decision attributes, even when the states of the linked agents is only known probabilistically.

6. CONCLUSION AND FUTURE WORK

The work presented here is in early stage of development, and complete testing is yet to be done. However, the estimation and learning procedures are operational on test situations and make the agents able to mimic real world behaviors. A new principle for stochastic decisions was also introduced, thanks to the use of a priori Dirichlet distribution, that gives a mean to cope with uncertain or noise contaminated observations. In a future work, an in-depth analysis of the algorithm will be done, and an extension to continuous valued attributes will be sought after. All the code will be made publicly available, and will hopefully be adopted in the air traffic management community as a mean of testing new concepts applicable to delay mitigation.

REFERENCES

[1] PRC - EUROCONTROL Performance Review Commission, «Performance Review Report Executive Summary – PRR 2016,» 2017.

[2] S. Belkoura, J. Peña and M. Zanin, "Beyond Linear Delay Multipliers in Air Transport," *Journal of Advanced Transportation Volume 2017, ,,* vol. Volume 2017, no. Article ID 8139215, p. 11 pages, 2017.

[3] CODA - EUROCONTROL Central Office for Delay Analysis, «A Matter of Time: Air Traffic Delay in Europe,» 2007.

[4] N. Pyrgiotis, K. Malone and A. Odoni, "Modeling delay propagation within an airport network," *Transportation Research,* vol. Part C, no. 27, pp. pages: 60-75, 2013.

[5] F. Moracamino, L. Weigang, O. Diaz Olariaga and G. Mykoniatis, "Network Delay Multipliers and Air Traffic Management," in *Roadef*, Metz, 2017.

[6] M. ALDERIGHI, A. CENTO, P. NIJKAMP and P. RIETVELD, "Assessment of New Hub-and-Spoke and Point-to-Point Airline Network Configurations," *Transport Reviews,,* vol. Vol. 27, no. 5, p. 529–549, September 2007.

[7] F. Balbo and S. Pinson, "An agent oriented approach to transportation regulation support systems," in *in Proceedings of the 5th Workshop in Agent in Traffic and Transport pp. 225–242*, Estoril, Portugal, 2008.

[8] D. Capera, M. Gleizes and P. Glize, "Self-organizing agents for mechanical design," *Engineering Self-Organising Systems,* pp. 169-185, 2004.

[9] M. Dorigo, M. Birattari and T. Stutzle, "Ant colony Optimization," *Computational Intelligence Magazine, IEEE,* vol. vol. 1, no. 4, pp. pp. 28–39,, November 2006.

[10] J. Ferber, Multi-agent systems: an introduction to distributed artificial intelligence, vol. 1, Harlow: Addison Wesley Longman - ISBN 0-201-36048-9, 1999.

[11] C. Rieger, K. Moore et T. Baldwin, «Resilient control systems: A multi-agent dynamic systems perspective,» chez *Electro/Information Technology (EIT), 2013 IEEE International Conference*, Rapid City, SD, Region 04 - Central USA, 2013.

[12] M. Gleizes, "Self-adaptive Complex Systems," in *European Workshop on Multi-Agent Systems*, Maastricht, The Netherlands, 2012.

[13] E. Kaddoum, "Optimization under constraints of distributed complex problems using cooperative self-organization," *Doctoral dissertation, Université de Toulouse,* 2011.

[14] L. E. Baum et Petrie, «Statistical Inference for Probabilistic Functions of Finite State Markov Chains,» *The Annals of Mathematical Statistics,* p. 1554–1563, 1966.

[15] J. R. Quinlan, "Induction of Decision Trees," *Machine Learning,* vol. 1, pp. 81-106, 1986.

[16] C. Sammut and G. Webb, "Baum-Welch Algorithm," in *Encyclopedia of Machine Learning*, 2010.

[17] P. Wang, L. Schaefer and L. Wojcik, "Flight Connections and their impacts on Delay propagation," in *Digital Avionics Systems Conference, 2003. DASC '03. The 22nd*, Indianapolis, IN, USA, 2003.