Self-Regulating Multi-Agent System for Multi-Disciplinary Optimisation Process

JEAN-BAPTISTE WELCOMME[1,2], MARIE-PIERRE GLEIZES[1], ROMARIC REDON[2] and THIERRY DRUOT[3]
[1] IRIT – Paul Sabatier University, Toulouse III
[2] EADS CRC - Corporate Research Center
[3] Airbus France

Abstract

This article presents a multi-agent method to tackle multidisciplinary optimisation, based on the notions of cooperation and self-regulation. It is focused on the preliminary aircraft design, which is a complex compromise. In our approach several cooperative agents collectively act to achieve a common goal, i.e. optimising a multi-objective function, even if the environment of the system (the user's requirements) changes during the solving process. In MASCODE, one agent encapsulates one discipline and is designed individually without considering the dependencies with the others. So the computation is conceptually distributed without central control. Experimental results including efficiency comparison with the classical FSQP method are presented, and show that the adaptive behaviour of MASCODE provides new capabilities to understand and manage the complexity of the preliminary aircraft design.

1 Introduction

1.1 Preliminary Aircraft Design

Preliminary aircraft design involves a lot of disciplines like weight, range, aerodynamic and operating cost estimations [8]. In addition to the multidisciplinary aspect, manufacturers and airlines have different objectives on the product. Most of the time, manufacturers search to design product families, whereas airlines are looking for aircraft satisfying their needs at best (number of passenger, range, depreciation...). So during this design process a lot of compromises are carried out in order to specify the high-level design achieving expected aircraft performances (number of seats, cruise range, takeoff distance, etc.). These compromises are difficult to achieve, because constraints are numerous and dependent.
Preliminary aircraft design is organised in two steps. First, a simulation function is built. It is obtained from the complex assembling of disciplinary models that represent a physic as a mathematical function with a set of inputs and outputs. Then, when the technical requirements (product performances) are known, the simulation is used to calculate the performances (Max Take Off Weight, Range, Operating Weight Empty); this is the *design direction.* Unfortunately, a mathematical inverse problem must be solved iteratively, because computational models are only known in the *analysis direction* ; computing product performances from design parameters. In the particular case of aircraft design, a lot of parameters are shared between disciplines. So the parameters and performances are highly interdependent, and constrained by their mutual tradeoffs [5].

1.2 Multi-Objective Optimisation

As presented, preliminary aircraft is an optimisation problem. Mathematical tools using response surfaces allow dealing with it. Especially the *Feasible Sequential Quadratic Programming* (FSQP), a gold standard method, enables to define objectives on performances and on design parameters, and then to find solutions to the design problem [12]. However the number of degrees of freedom and the parameter interdependencies imply that the solution space is discontinuous. So, these traditional mathematical methods are not really adapted to the preliminary aircraft design, because the discontinuity makes difficult to find design points that satisfy all the constraints and then to optimise them. Genetic algorithms (GA) offer very interesting robustness to tackle this problem and to find admissible point, because they are independent of the discontinuity. However GAs optimise the design as a global problem, and provide a limited view on compromise solution, since it is obvious that the aircraft is never a mathematical optimum but an alchemic compromise [2]. Pareto front are computed to compensate for this lack by pointing the region of the best compromise rather than providing only one optimal design. However if they improve the solution quality by providing more information, they do not really offer a better understanding since they are difficult to compute and to visualise especially when the targeted solution is really multi-objectives.

1.3 Multi-Disciplinary Optimisation

Kroo defines MDO (Multi-Disciplinary Optimisation) as "a methodology for the design of complex engineering systems and subsystems that coherently exploits the synergism of mutually interacting phenomena" [8]. During the last three decades, various types of computational or computer-aided design systems have been developed in MDO domain. A lot of issues were addressed like interoperability, problem decomposition, design robustness analysis and uncertainty propagation.
Several strategies were proposed for the global optimisation and the subsystems linkage, exploiting the synergy of interactions through *Fixed Point Iteration* (FPI) algorithms [1]. Many relations between mathematic analyser and optimiser were studied, in which an analyser defines an execution order for computing the different models, whereas an optimiser compares their results and adapts the design parameters to converge on target criteria, like in *All at Once* (AAO), *Multi Disciplinary Feasible* (MDF) and *Individual Disciplinary Feasible* (IDF). However these strategies are finally first decomposed in subsystems and then centralised within an optimiser. So the decomposition of the system becomes a key point and influences the resolution. More complete approaches such as *Collaborative Optimisation* (CO), *Concurrent Sub-Space Optimisation* (CSSO) offer multi-level architectures, where each disciplinary has its individual optimisation strategy [8]. *Analytical Target Cascading* (ATC) is another alternative, in which each component is itself an optimiser [1]. As a consequence, the system is hierarchical and each component tries to minimise its individual objectives and those of its neighbours. The MASCODE[1] approach presented in this paper have some similitude with it, but it processes are adaptive and dynamic.

1.4 Self-Organising Multi-Agent Approach

Distributed Constraint Optimisation Problems (DCOP) are an important research area for multi-agent systems. Its objective is to propose an optimal assignment to a set of variables spread over a number of agents. A number of powerful distributed algorithms such as SynchBB [7], ADOPT [10], OptAPO [9] have been developed, and provided solutions either optimally, or close to optimality. However as these approaches are inspired from non-distributed combinatorial optimisation, they present lacks to be well-used to solve a dynamics problem with continuous parameters.
Self-organising multi-agent approach works on the apparition of a functional structure spontaneously maintained in a dynamic equilibrium by all the participating components [6]. As described in [4], self-organising MAS (Multi-Agent Systems) offer opportunities to simulate real complex systems, because of agents have ideally autonomous behaviours; adapt constantly their state relative to each other; learn from experience; and create dynamically group and organization. As described previously, the preliminary aircraft design is a complex process, because of multi-disciplinary aspects and of multi-objectives criteria. In addition, the interdependencies between the parameters imposed to make a lot of compromises that dynamically change the problem formulation. All these characteristics make self-organising multi-agent approach a promising solution to support the preliminary aircraft design.
In this paper, we present a cooperative multi-agent approach based on the AMAS theory (Adaptive Multi-Agent Systems) to solve the preliminary aircraft problem [3]. According to the "organisaction" principles [11], a self-organising system is described as to be able to self-regulate, self-relate and self-product. The paper is focused on the description of the self-regulation process, which aims

[1]MASCODE : Multi-disciplinary Aircraft Simulation for COnceptual DEsign

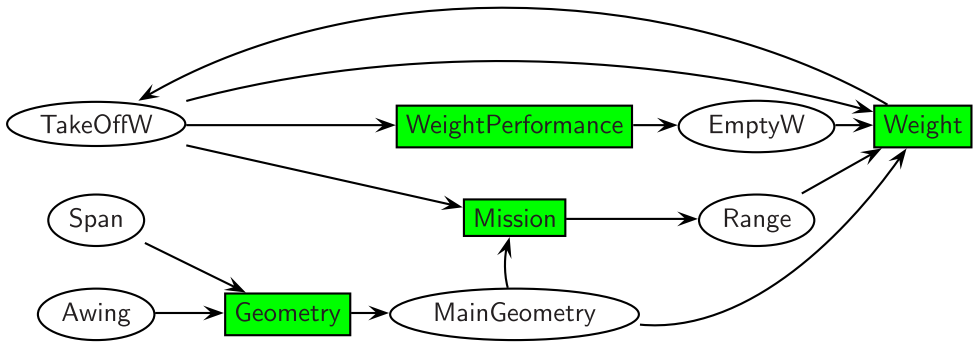

Figure 1: A simplified example of relation between models

at finding the optimised values of several parameters in a given system. The system properties make that some parameters are shared between several disciplines and/or strongly interdependent. The paper is structured as follows. First, the principle of using a MAS for enabling preliminary aircraft design through cooperative reasoning are detailed; then, MASCODE results are described and compared with FSQP results; finally we highlight the main long term opportunities associated to our approach compared to latest MDO research works.

2 System Design

2.1 Introduction

MASCODE uses the specific resolution strategy of AMAS theory. This is a cooperative strategy, which is focused on the identification of a set of local Non Cooperative Situations (NCS) for the agents. By now, a cooperative agent is assigned to a discipline and the aim of an agent is to cooperate with its neighbours to find the values of different parameters in a given system, as shown in figure 1. Agents are drawn in square and shared parameters in oval.
For the system, some parameters are inputs (TakeOffW, Span, Awing), outputs (TakeOffW) and intermediates (Range, EmptyW...). Any one of these parameters can be a user objective. However regarding the characteristics of the problem, the only freedom degrees are input parameters. Due to interdependencies between parameters, decomposition of the global problem by disciplinary and subtask, it seems possible to gain advantages to use a distributed resolution process that will take into account the shared constraints between entities as in ADOPT or DPOP algorithms, or in multi-agent approaches in general.
In MASCODE, one agent controls one discipline. Therefore, these agents are called ***Disciplinary Agents*** (DA). The multi-agent system is a network of DA corresponding to the model hierarchy commonly found in preliminary aircraft design. Each DA owns representation knowledge of the model and learned knowledge from experiences, which are used through a set of behaviours to communicate and to take decision according to environment perception.

2.2 DA's Knowledge

This knowledge is static or dynamic, and is twofold: knowledge on its relations (connection with neighbours) and knowledge on its model.

2.2.1 Knowledge on Relations

To interact, each DA knows its provider and user agents. For example, in figure 1, for the agent Weight the users are WeightPerformance and Mission, and the providers are Geometry, Mission and WeightPerformance. A user agent uses the computed value of one of its output parameter, and a provider sends the value of one of its input parameters. In addition to this static knowledge, DA learns experiences during the execution, and builds memories. Memory is a key element in the

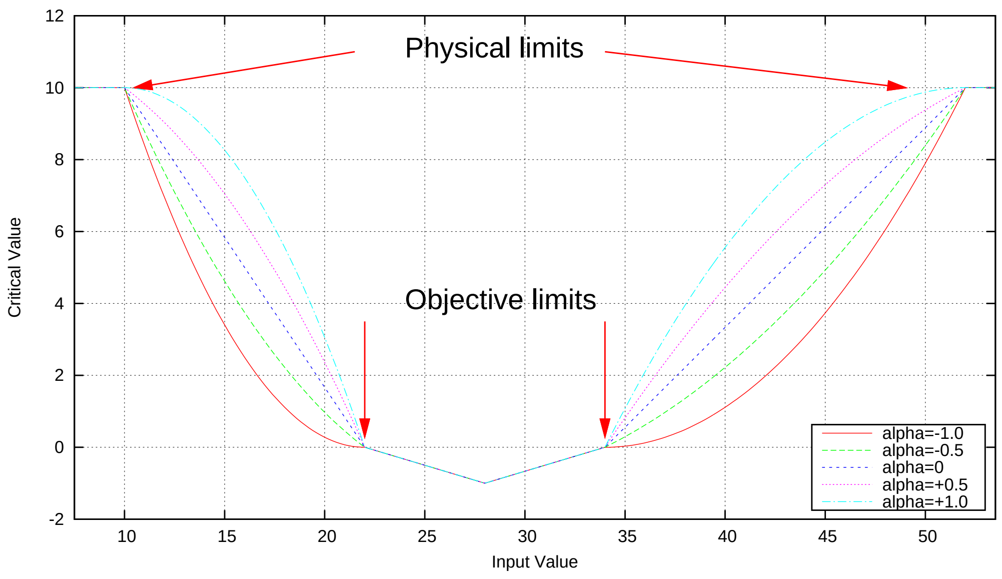

Figure 2: Evaluation functions of critical values (interval validity functions).

AMAS approach, because an agent adapts its behaviour, and takes decisions in function of its past experiences (see section 2.3.2).

2.2.2 Knowledge on Model

Each model possesses physical properties. They are explainable on each input of models and defined with validity domains:

- Lower and upper bounds of the design variables provide a *physical validity interval* (*physical limits*) for each input, in which the domain is computable.
- An *objective validity interval* (*objective limits*) describes a preferred interval. All the values inside this range fit the user constraints.

With these intervals, we defined a parametric[2] piecewise continuous mathematical function, shown in figure 2, that enables for the agent to compute a satisfaction criteria. It indicates whether the agent respects its physical limits and its objective limits:

- when the input value is inside the objective validity interval the critical value is negative,
- when the input value is inside the physical validity interval but outside the objective validity interval, the critical value is positive and inferior to a critical threshold, predefined by the designer, which is the maximal critical value in the system,
- when the input value is outside the physical limits, its critical value is equal to the critical threshold.

Finally, the non-satisfaction degree of the agent is defined as the *maximum* of its input critical values from its input parameters.

2.3 DA's Behaviour

Each agent is able to receive and send messages. In a first phase, agents compute their modules and transmit, via a *forward message*, the value of their outputs to their user agents. This phase is completed once an agent received all its forward messages from its providers. Consequently to the reception of forward messages, agents may send *backward messages* to inform providers when

[2]The parameter *alpha* is used to indicate whether a constraint is hard or not (higher *alpha* is, harder the constraint is).

```
begin
  1. while not all requests for all outputs received do
     (a) while not all requests for output $j$ received do
           i. reception of backward message on output $j$ for user $k$ ;
          ii. update knowledge on output neighbour $k$ ;
         done
     (b) select the most critical request for output $j$ ;
     (c) for each input dependencies do
           i. build the corresponding modification for input $i$
         done
     done
  2. for each input do
     (a) select the most critical input objective ;
     (b) if input $i$ no predecessor then
           i. adapt the input value according to the most critical input objective
     (c) else
           i. send a new backward message according to the most critical input objective
         fi
     done
end
```

Figure 3: Backward message phase procedure for a DA.

the provided value is not relevant. This second phase is completed once the agent received all its backward messages from its users. Thus, according to the received information in backward messages and to its individual state, it sends a modification request to its predecessors.

2.3.1 Cooperative Reasoning

The Cooperative Reasoning is designed across Non Cooperative Situations (NCS) [3], composed of a description (conditions, triggers) and a set of actions. The description can be viewed as a rule containing all the conditions necessary to recognise the NCS. The sets of actions describe how the agents can improve the cooperation of their neighbourhoods. When all NCS are identified, the main objectives and the high-level decision model of agents are known.
In our cooperative approach and due to a set of NCS, we define the aim of each agent, which consists to do the action that decreases the most critical situation in the system. By measuring a non-satisfaction degree (the maximum of all the critical input values) in function of its objectives and of its physical limits, each agent can compare its critical value with the critical values of its neighbours (received requests). Then, it takes local cooperative decisions according to the following main principles:

- When the agent is the most critical, it builds a modification request for itself.
- When the agent is less critical than a modification request, it acts for the modification request. For that it computes its Jacobian matrix[3] and finds the local dependencies between the concerned output and its inputs. Thus with the modification request and with its local dependencies, the agent is able to send a new modification request to its neighbours, that would help the received one.

[3]The Jacobian is equivalent to a derivative of a multivariate function

2.3.2 Learned Experiences and Adaptive Input Variation Steps

The reasoning can be cooperative only if the decision model takes into account the past experiences of the agent. Without any reasoning on the past experiences the system is open to oscillations and chaotic phenomenons. However in MASCODE, the memory is quite simple. While moving to a solution, if the modification direction of an input is successively the same, the agent considers it as a positive feedback and increases an input variation step. Conversely, if the modification direction is changing, agent considers it as a negative feedback, and decreases the variation step. The initial variation step is a percentage of the total interval of the objective limits given in the figure 2. This behaviour allows a dynamic equilibrium when the system converges to a global solution as shown in section 3.

2.3.3 Algorithm

To sum up the DA's behaviours, the *backward message phase*, is presented in figure 3. The modifications are propagated across the system. DA agent uses its cooperative reasoning to select the modification requests it wants to create or transmit on each input. For each output, an agent possesses several users, because one parameter is often shared between several disciplines. So in a first time, it receives the modification for each output parameter (step 1a). Then, it selects for each output the most critical request and uses its knowledge on intput/output dependencies to build the corresponding request on its inputs (step 1b). Then, it selects the modification to transmit to its provider. When all critical situations have disappeared, all agents are in a satisfied state and the system has converged.

3 Experiments and Results

MASCODE uses the framework JADE (Java Agent DEvelopment Framework). To validate the approach, some experiments have been done for a sample preliminary aircraft design case study with 10 models and 60 parameters (20 inputs, 17 outputs, 23 intermediates), in which 14 parameters are objectives (7 inputs design freedom degrees, 7 outputs performances).

3.1 Comparison with FSQP

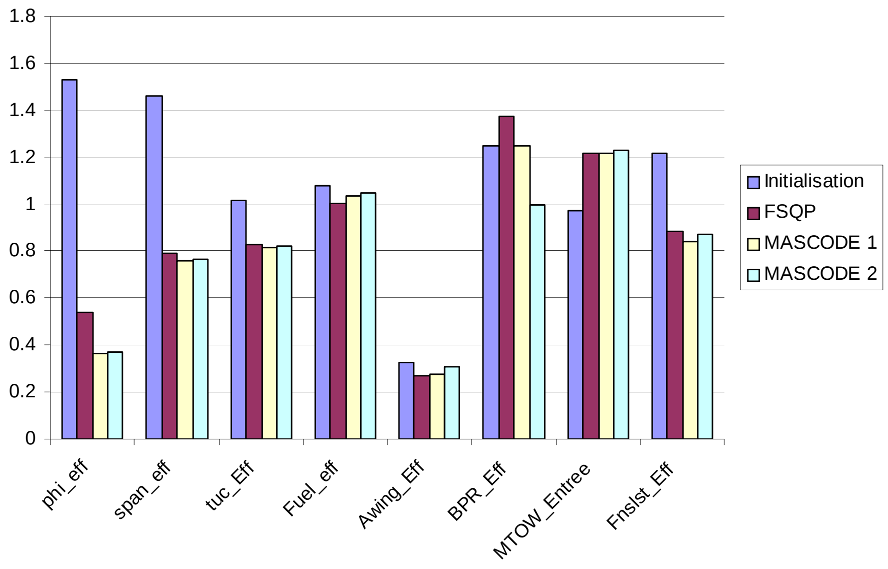

Figure 4: Comparison of input objectives obtained with FSQP and Mascode.

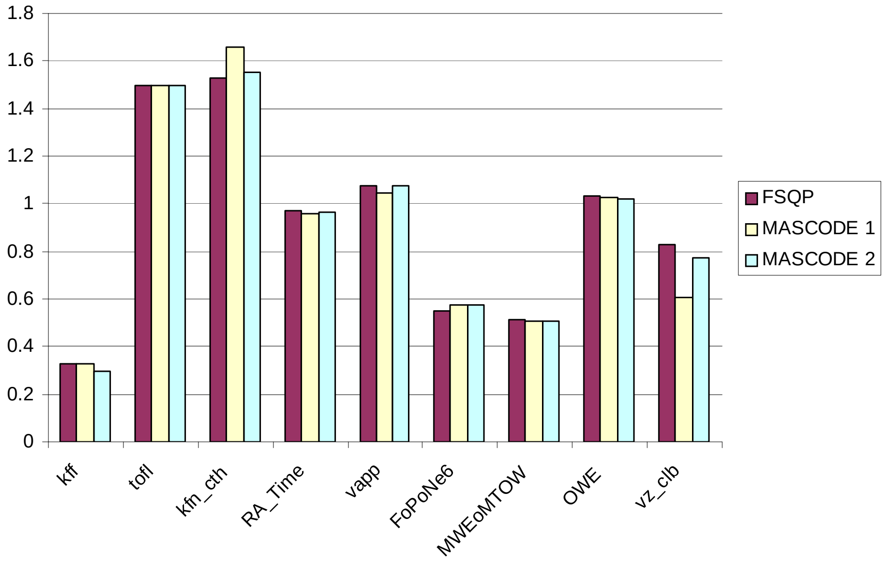

Figure 5: Comparison of output objectives obtained with FSQP and Mascode.

MASCODE solutions have been compared with FSQP constraints satisfaction mode. For MASCODE and FSQP, the same intervals are provided on the objective parameters. Then the system adapts its parameters until constraints would be satisfied. Experimental solutions are plotted on the figures 4 and 5. The input objectives of the problem are illustrated with figure 4 and output objectives with 5. Results show that the found solutions are similar. For the same problem, two different solutions called "MASCODE 1" and "MASCODE 2", are presented on the histograms. Solutions can be different at each resolution, but are equivalent because they respect the problem constraints. By contrast, FSQP provides always the same solution, because its optimisation process is deterministic.

3.2 A MASCODE Execution

Figure 6 shows the evolution of the objectives parameters[4] during the solving process. X-axis represents the time and Y-axis the normalised parameter values. Thus, all parameters can be plotted on a same graph. Figure 7 shows the same evolution for the critical value of each objective parameter. X-axis represents the time and Y-axis the parameter critical values. The system finds a solution, when the critical values are all null. As shown in figure 7, the system finds four solutions during the computation. In fact at each solution, the user introduces some new constraints in the system, figure 8. These new constraints breaks the equilibrium by introducing new critical situations (new disorders). Then a new self-adaptive process is engaged, because the problem formulation has changed. An entire scenario is explained in the next section. Figure 7 shows that the critical values globally decrease. However the decrease is sometimes discontinuous, because parameters are more or less sensible to the modifications. So an agent can decide a modification without knowing it would not be really a cooperative choice. However each agent learns progressively this kind of non-cooperative situations and the system converges.

3.3 Adaptive Behaviour of MASCODE

MASCODE provides user interfaces that help to understand/manage the system. For example, it provides a view of the system with the repartition of the critical values, individual interfaces for each DA and various graphics to pay attention on the parameter evolutions. The scope of this article is not to present all these properties. However, we provide a general survey of MASCODE capabilities

[4]These values are normalised for visual representation.

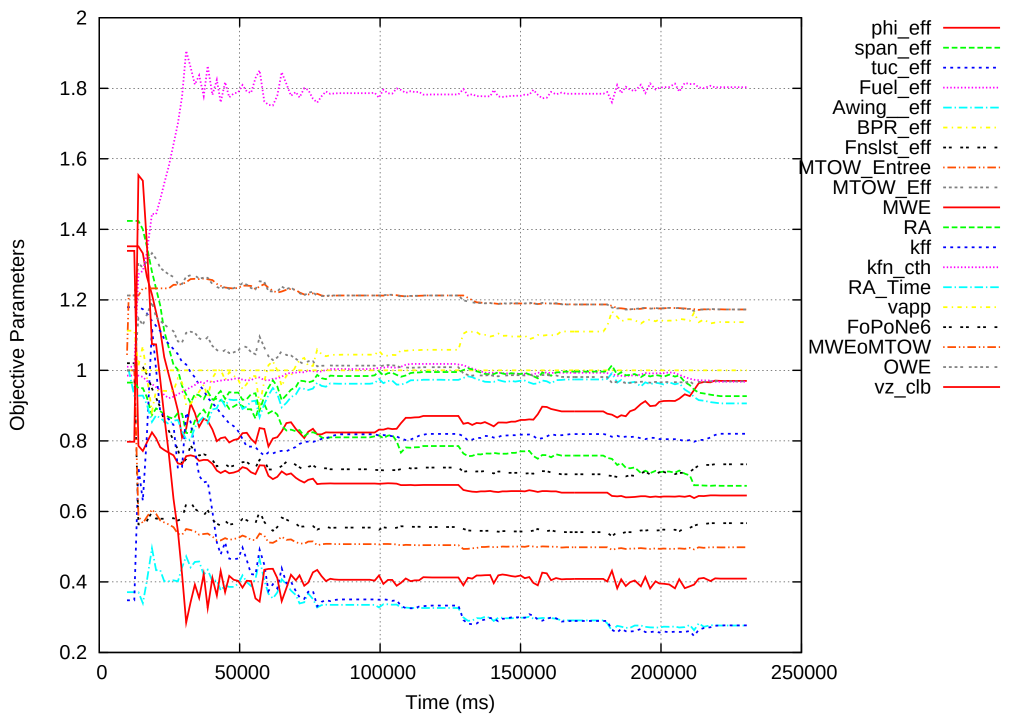

Figure 6: Evolution of objective parameters during a Mascode execution

to provide a really dynamic and adaptive system. Figures 8 and 9 detail some parameters during a resolution, described in 3.2. These figures illustrate the dependencies between the parameters RA (RAnge of the mission), $MTOW$ (Max Take Off Weight) and MWE (Manufacturing Weight Empty). Some of the relations of these parameters are illustrated on the figure 1. To provide an example and to simplify, they could be expressed as follows:

- The range RA impacts the fuel weight and so $MTOW$.
- MTOW impacts the manufacturing weight empty. When MTOW increases, the aircraft structural constraints change.
- If the aircraft structure changes, geometry could evolve.
- If the geometry of the aircraft evolves, aerodynamic forces and the range could also be modified.

During the presented process, the constraints was changed by the user as follow (see figures 8 and 9):

1. At time $t = 84s$, the objective on RA (range) was increased of 1%. It immediately introduces a new critical value for RA. But this modification does not impact $MTOW$ and MWE.
2. At $t = 100s$, user asks for a diminution of the $MTOW$. First the critical value of $MTOW$ increases and then the new constraint is shared between RA, $MTOW$ and MWE. Then system is unable to converge, because it is over-constrained.
3. At $t = 146s$, a modification of the MWE objective provides new freedom degrees. This modification is not important (see figure 8) but enough to decrease $MTOW$ without changing the mission RA.
4. At $t = 150s$, the $MTOW$ constraint is reinforced (a lowering of 2%).
5. At $t = 190s$, the mission performance RA is degraded and enables the system to converge, because of links between RA and $fuel$, and between $fuel$ and $MTOW$.

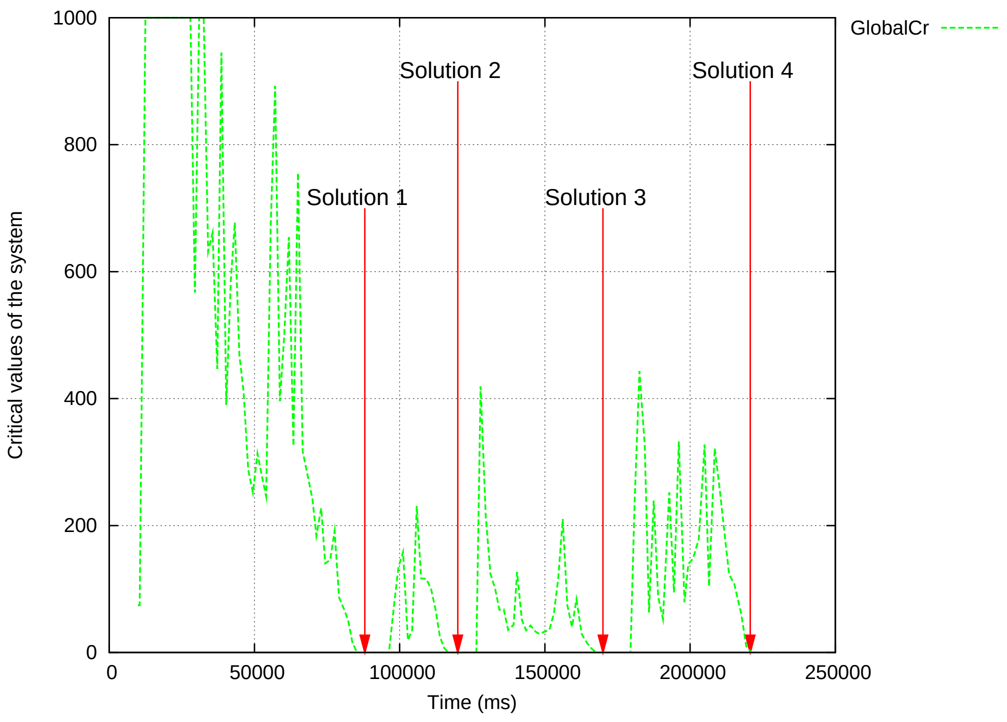

Figure 7: Evolution of critical values in the system during a Mascode execution

About the results and this scenario, it is quite clear that MASCODE helps the designer to understand and to manage the constraints in the system. Each time the problem formulation changes, the agents adapt their behaviour and search a new equilibrium. When a new stable state is not reachable [5], agents self-regulate the critical values in the system and help the user to identify conflicting parts. Thanks to this information, he can alter the strongest constraints and let the system converge toward another relevant solution.

4 Discussion

In addition to the comparison to the FSQP method for the case study in the previous section, MASCODE can be compared to other MDO approaches. This comparison is done at a relative high-level, since existing methods are not agent-based, and do not tackle dynamic and changeable problems.

Solution quality The quality of MASCODE solutions is equivalent to the solution found by FSQP. FSQP method is based on a gradient descent, recognised as finding good solutions in nonlinear optimisation problems. However we used FSQP only in its constrained solving mode. So we need to go further in a multi-objective approach for a full comparison.

Convergence speed The time of convergence is the same as FSQP, but none systematic measure of convergence speed has been realised for large problems, because it is not our first intention. Nevertheless as described in [1], the convergence speed depends on the problem decomposition. Our problem decomposition is close to AAO, which is considered as the faster in comparison with IDF and MDF.

Robustness and disciplinary knowledge integration In MASCODE, validity intervals are local knowledge about physical models. Introducing this knowledge in the resolution process is a first key point for improving the result consistency. Thus, it will be possible to add other knowledge in the reasoning, and to include it in the agent decision model. As other

[5]The system is over-constrained

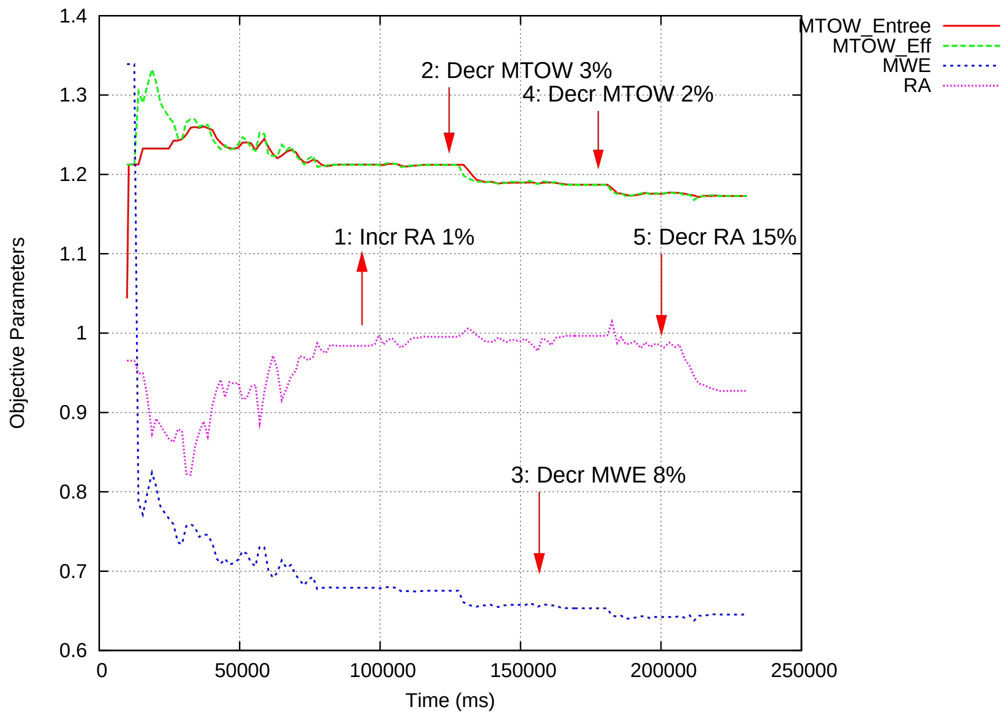

Figure 8: Values of parameters RA, $MTOW$ and MWE

methods imply a mathematical formulation of the constraints, adding new knowledge implies new constraint formulations, which is not evident in the general case since it requires multi-expertise.

Disciplinary openness MASCODE requires no global information and decision process. Consequently, adding or deleting physical models consists only in updating the MAS. In all other non-agent MDO approaches, the openness capability is never invoked. However, this incremental functionality could be very useful for designing complex systems, because as we saw the context is dynamic and the designers often change their requirements.

Parameter adjustment In MASCODE, users can adjust values and associated validity domains of parameters in real time, because agents will dynamically change their behaviours according to this new knowledge. There is no information about parameter modification in runtime for the other approaches. By now, this is the most relevant property of MASCODE, because it permits to understand the relation between disciplines by the negotiations and to manage/adapt dynamically the constraints of the problem. As described, to integrate this particularity, the system needs to be robust to change and have to be itself dynamic and self-adaptive.

5 Conclusion

This article has presented a multi-agent method to tackle multidisciplinary optimisation, based on the notions of cooperation and self-adaptation. In MASCODE, the physical models are encapsulated in cooperative agents which negotiate and cooperate to find an optimal (or near optimal) solution. This approach is efficient and provides relevant results, in comparison to the classical FSQP method. The method is non complete, but the main objective, in the preliminary aircraft design context, is to quickly provide approximated solutions, with some user guidances and interferences during the solving process. From learned lessons, DA approach for multidisciplinary design optimisation can be considered without doubt as relevant for many reasons:

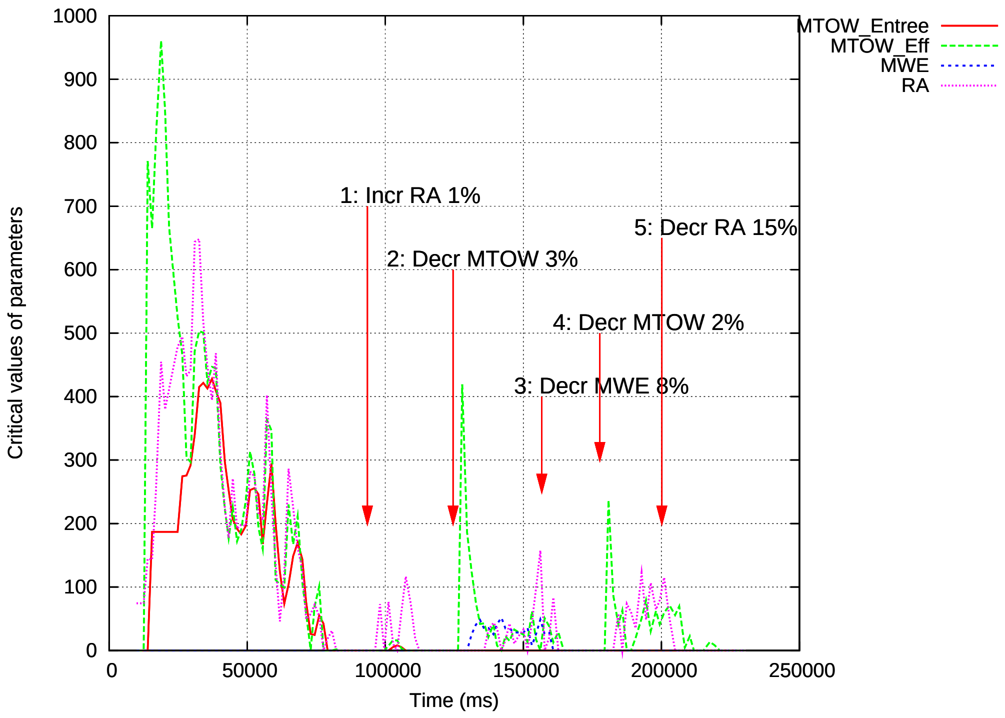

Figure 9: Critical values of parameters RA, $MTOW$ and MWE

- Each disciplinary model can be design individually without considering the dependencies with its neighbours. This ability reduces greatly the complexity of the MDO framework.
- An agent can encapsulate the disciplinary model, but also all the associated knowledge such as critical values, execution time, precision and granularity. So the quality of the solution is potentially better.
- The MASCODE computation is conceptually distributed without central control. Thus, the running can be entirely concurrent leading to a time reduction.

The design process of an aircraft is multi-disciplinary, multi-objective and also multi-level. All these aspects increase the complexity of an aircraft design. By now, we consider some of the multi-disciplinary and multi-objective aspects. However to provide a self-organized system, we need to add other knowledge on the disciplinary models (granularity, precision, computation time, semantic...) and by consequence to consider new cooperative situations for fully re-organize the system. For example, a re-organization process could be to change a model inside a discipline for a concurrent one or to change the granularity of a model. This is the main focus of our research and developments by now.

References

[1] J. T. Allison. Complex System Optimization: A Review of Analytical Target Cascading, Collaborative Optimization, and Other Formulations. Master's thesis, Department of Mechanical Engineering, University of Michigan, 2004.

[2] C. Badufle, C. Blondel, T. Druot, and M. Duffau. Automatic satisfaction of constraints set in aircraft sizing studies. In *6th World Congresses of Structural and Multidisciplinary Optimization (WCSMO'05)*, 2005.

[3] C. Bernon, V. Camps, M.-P. Gleizes, and G. Picard. *Engineering Self-Adaptive Multi-Agent Systems : the ADELFE Methodology*, chapter 7, pages 172–202. Idea Group Publishing, 2005.

[4] G. Di Marzo Serugendo, M.-P. Gleizes, and A. Karageorgos. Self-Organisation and Emergence in Multi-Agent Systems: An Overview . *Informatica, An International Journal of Computing and Informatics*, 30(1):45–54, 2006.

[5] K. Fujita and H. Yoshida. Product Variety Optimization Simultaneously Designing Module Combination and Module Attributes. *Concurrent Engineering, Research and Applications*, 12(2):105–118, 2004.

[6] Francis Heylighen and Carlos Gershenson. The meaning of self-organization in computing. *IEEE Intelligent Systems*, pages 72–75, July/August 2003.

[7] K. Hirayama and M. Yokoo. Distributed Partial Constraint Satisfaction Problem. In *Principles and Practice of Constraint Programming*, pages 222–236, 1997.

[8] I. Kroo. Distributed Multidisciplinary Design and Collaborative Optimization. White paper, VKI lecture series on Optimization Methods and Tools for Multicriteria/Multidisciplinary Design, 2004.

[9] Roger Mailler and Victor Lesser. Solving Distributed Constraint Optimization Problems Using Cooperative Mediation. In *Proceedings of Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, pages 438–445. IEEE Computer Society, 2004.

[10] P. J. Modi, W. Shen, M. Tambe, and M. Yokoo. ADOPT: Asynchronous Distributed Constraint Optimization with Quality Guarantees. *Artificial Intelligence*, 161(2):149–180, 2005.

[11] E. Morin. *La mthode : La vie de la vie.* 1980.

[12] J. L. Zhou, A. L. Tits, and Lawrence C. T. *User's Guide for FFSQP Version 3.7: A FORTRAN Code for Solving Constrained Nonlinear (Minimax) Optimization Problems, Generating Iterates Satisfying All Inequality and Linear Constraints*, 1997.