10th World Congress on Structural and Multidisciplinary Optimization
May 19 - 24, 2013, Orlando, Florida, USA

Experimenting on a Novel Approach to MDO using an Adaptive Multi-Agent System

Tom Jorquera, Jean-Pierre Georgé, Marie-Pierre Gleizes, Christine Régis, Victor Noël

IRIT (Institut de Recherche en Informatique de Toulouse)
Paul Sabatier University, Toulouse, France
{jorquera,george,gleizes,regis,vnoel}@irit.fr

1. Abstract
MultiDisciplinary Optimization (MDO) problems represent one of the hardest and broadest domains of continuous optimization. By involving both the models and criteria of different disciplines, MDO problems are often too complex to be tackled by classical optimization methods. We propose an approach for taking into account this complexity using a new formalism (NDMO - Natural Domain Modeling for Optimization) and a self-adaptive multi-agent algorithm. Our method agentifies the different elements of the problem (such as the variables, the models, the objectives). Each agent is in charge of a small part of the problem and cooperates with its neighbors to find equilibrium on conflicting values. Despite the fact that no agent of the system has a complete view of the entire problem, the mechanisms we provide make the emergence of a coherent solution possible. Evaluations on several academic and industrial test cases are provided.
2. Keywords: Multi-Agent System, Multidisciplinary Optimization, Integrative Design

3. Introduction

In their review about multidisciplinary optimization (MDO), Sobieszczansky-Sobieski and Haftka propose to define it *as methodology for the design of systems in which strong interaction between disciplines motivates designers to simultaneously manipulate variables in several disciplines* [1]. Designers have to simultaneously consider different disciplines (such as, for example, aerodynamics, geometrics and acoustics for an aircraft engine) which are often not only complex by themselves but also strongly interdependent, causing the classical optimization approaches to struggle handling them.

Formally, MDO problems are continuous optimization problems where the goal is to find the values of a set of inputs that maximize (or minimize) several objectives while satisfying several constraints (both often regrouped under the term *optimization criteria*). These problems tend to be complex to solve as they can involve calculus-heavy, interdependent models and contradictory criteria.

Currently, MDO problems require specific strategies to be solved, and a major part of the research in the field has been focusing on providing these strategies. These approaches often involve reformulating the problem, requiring techniques to maintain coherency among variables shared by different disciplines and specific ordering of local optimizations on sub-parts of the problem. Thus an important part of the burden is still on the shoulders of the engineers.

In this paper, we propose an original approach using a Multi-Agent System (MAS) [2] for solving this kind of optimization problem in the most generic way while keeping the need to reformulate the problem at a minimum. This system is composed of autonomous agents which allow to model each discipline independently. They interact and cooperate with each other in order to solve discipline interdependencies. Inside the MAS, each discipline may be easily distributed and may evolve without impacting the global system.

As an MDO problem implies different disciplines, several engineers (one per discipline for instance) may have to intervene in the global optimization process of the problem. We propose that each engineer may directly interact with the system *during the solving process* in order to change, to test, to adapt or to add elements to the parts of the problem inherent to its discipline. This implies offering the engineers an easy way to modify their own constraints of the problem, to set specific values to some variables or change their definition domains and to automatically take these changes into account. We call this vision of MDO *Integrative and Interactive Design* as stated by the ID4CS project*.

*Integrated Design for Complex Systems, national french project regrouping 9 academic and industrial partners, including

Our main focus is to design the self-adaptation capabilities of the proposed system as a potentially infinite feedback loop between the system and its environment, which is typical of self-adaptive and self-organizing complex systems. As explained in [3], by using the emergence phenomena in artificial systems, our aim is to obtain a system able to cut through the search space of any problem far more efficiently than by simply dividing the problem and distributing the calculus.

In the next part (section 4), we begin by reviewing existing optimization methods, both from MDO and MAS sides, and argue that they are not adapted to solve the issues we propose to tackle. We then present in section 5 a new generic agent-based modeling for continuous optimization problems, called Natural Domain Modeling for Optimization (NDMO). Using NDMO we describe in section 6 an adaptive multi-agent algorithm for solving continuous optimization problems. We present in section 7 the results of our algorithm on different test cases, and finish by perspectives about future improvements based on the current work.

4. Existing methods

4.1. MDO methods

Classical MDO methods delegate the optimization in itself to standard optimization techniques, which must be chosen and applied by the engineer, according to his knowledge of the problem and his skills. The functioning of these methods can vary greatly. For example Multi-Disciplinary Feasible Design, considered to be one of the simplest methods [4], consists only in a central optimizer taking charge of all the variables and constraints *sequentially*, but gives poor results when the complexity of the problem increases [5]. Other approaches, such as Collaborative Optimization [6] or Bi-Level Integrated System Synthesis [7], are said bi-level. They introduce different levels of optimization [8], usually a local level where each discipline is optimized separately and a global level where the optimizer tries to reduce discrepancies among the disciplines. However these methods can be difficult to apply since they often require to heavily reformulate the problem [9],and can have large computation time [5].

One of the major shortcomings of these classical methods is that they require a lot of work and expertise from the engineer to be put in practice. To actually perform the optimization process, one must have a deep understanding of the models involved as well as of the chosen method itself. This is mandatory to be able to correctly reformulate the models according to the formalism the method requires, as well as to work out what is the most efficient way to organize the models in regard to the method. Since by definition MDO involves disciplines of different natures, it is often impossible for one person to possess all the required knowledge, needing the involvement of a whole team in the process. Moreover, answering all these requirements implies a lot of work *before* even starting the optimization process.

4.2. Multi-Agent Systems for Optimization

While multi-agent systems have already been used to solve optimization problems, the existing works concern their application to *Combinatorial* Optimization, mainly in the context of the DCOP (Distributed Constraint Optimization Problem) formalism, which usually applies to constraint optimization problems where the definition domains of the design variables are discrete and finite.

In DCOP, the agents try to minimize a global cost function (or alternatively, maximize a global satisfaction) which depends on the states of a set of design variables. Each design variable of the optimization problem is associated to an agent. The agent controls the value which is assigned to the variable. The global cost function is divided into a set of local cost functions, representing the cost associated with the conjoint state of two specific variables. An agent is only aware of the cost functions which involve the variable it is responsible for.

While some works successfully used DCOP in the context of continuous optimization [10], this formalism is not adequate to handle the type of problems we propose to solve here. DCOP problems are supposed to be easily decomposable into several cost functions, where the cost values associated to the variables states are supposed to be known. This major assumption does not stand for MDO problem, where the complexity of the models and their interdependencies cause this information to be unavailable

Airbus and Snecma (Safran Group)
`http://www.irit.fr/id4cs`

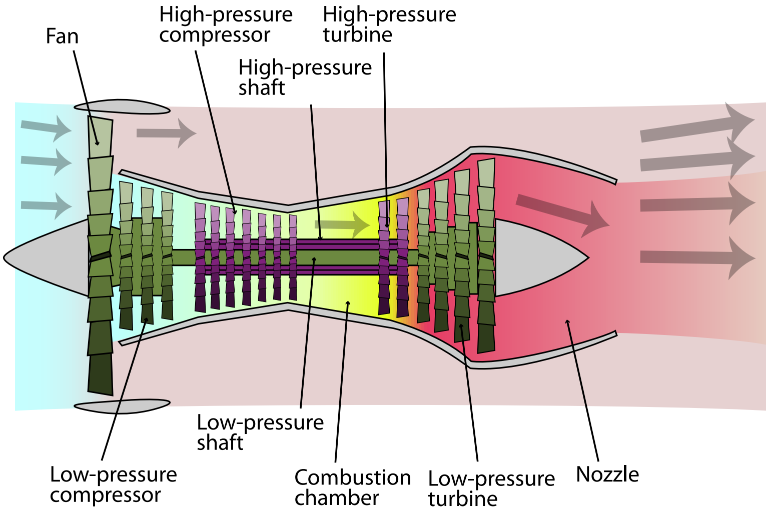


Figure 1: Illustration of a Turbofan engine (CC SA-BY K. Aainsqatsi)

in most cases. Trying to model such MDO problems with DCOP would result in a system where most agents are related to every other agent, with unknown cost functions.

Moreover, the existing agent-based optimization techniques for DCOP often present similar shortcomings to MDO methods, in the sense that they require a strong expertise to be efficiently applied [11].

5. Problem Modeling with NDMO

In answer to the previous shortcomings, we propose a generic approach called Natural Domain Modeling for Optimization (NDMO) that relies on a natural or intrinsic description of the problem (*i.e.* close to the reality being described).

To illustrate how an optimization problem is modeled, we use a simplified Turbofan optimization problem. On Figure 1, an illustration of the principle of the turbofan can be seen. In this figure, the bypass ratio is the ratio between the air drawn in by the fan not entering engine core (which is *bypassed*) and the air effectively used for the combustion process. The pressure ratio is the ratio between pressure produced by the compressors and the pressure it receives from the environment.

In order to identify the elements of a generic continuous optimization model, we worked with experts from several related fields: numerical optimization, mechanics as well as aeronautics and engine engineers. As a result, we identified five classes of interacting entities: *models*, *design variables*, *output variables*, *constraints* and *objectives*.

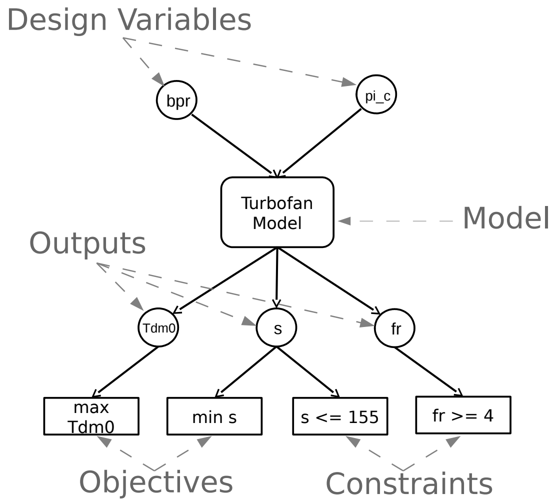
In Figure 2a, the analytic expression of this optimization problem is given, while in Figure 2b, the problem is presented as a graph of the different entities. The design variables of this problem are *pi_c* and *bpr*, which indicate respectively the compressor pressure ratio and the bypass ratio of the engine. The turbofan model produces three outputs: $Tdm0$, s and fr, representing respectively the thrust, fuel consumption and thrust ratio of the engine. In this problem we try to maximize the thrust and minimizing the fuel consumption while satisfying some feasibility constraints.

Let's now see in more details the roles of each of these fives entities: *model*, *variable*, *output*, *constraint* and *objective*.

Models. In the most general case, a *model* can be seen as a black box which takes input values (which can be *design variables* or *output variables*) and produces output values. A *model* represents a technical knowledge of the relations between different parts of a problem and can be as simple as a linear function or a much more complex algorithm requiring several hours of calculation. Often some properties are known (or can be deduced) about a model and specialized optimization techniques can exploit this information. In our Turbofan example, a *model* entity is the $Turbofan$ function which calculate the three outputs using the values of *bpr* and *pi_c*.

$$(Tdm0, s, fr) = Turbofan(pi\_c, bpr)$$
$$max\ Tdm0$$
$$min\ s$$
$$subject\ to$$
$$s \leq 155$$
$$fr \geq 4$$

(a) mathematical formulation.



(b) corresponding entities graph.

Figure 2: Turbofan problem.

Design Variables. These are the inputs of the problem and can be adjusted freely (within their defining boundaries). The goal is to find the set(s) of values for these variables that maximize the objectives while satisfying the constraints. *Design variables* are used by *models* to calculate their outputs and by constraints and objectives to calculate their current value. A *design variable* can be shared among several *models*, objectives and constraints. Keeping with our example, *bpr* and *pi_c* are the two *design variables* of our optimization problem.

Output Variables. These values are produced by a *model*, and consequently cannot be changed freely. As for the *design variables*, the *output variables* are used by *models* to calculate their outputs and by constraints and objectives to calculate their current value. In our example, $Tdm0$, s and fr are *output variables* produced by the $Turbofan$ model.

Constraints. These are strict restrictions on some parts of the problem, represented as functional constraints defined by equalities and/or inequalities. These can be the expression of a physical constraint, or a requirement concerning the problem. Regarding the Turbofan, the two *constraints* are $s <= 155$ and $fr >= 4$.

Objectives. The goals to be optimized. In the general case, different objectives are often contradictory. The two *objectives* of the Turbofan problems are to maximize $Tdm0$ and to minimize s.

An interesting and important point is that both models, constraints and objectives involve computation. Often the most heavyweight calculus is encapsulated inside a model and the calculi concerning criteria tend to be simple equations, but this is neither an absolute requirement nor a discriminating characteristic.

The NDMO modeling aims to provide the most complete and natural representation of the problem. This modeling preserves the relations between the domain entities and is completely independent of the solving process. Since we now have a way to model optimization problems as graphs of entities, we now present the multi-agent algorithm proposed to solve them.

6. A Multi-Agent System for MDO

In complement to this modeling of the problem, we propose for NDMO a multi-agent system and associated solving behaviors where each domain entity is associated with an agent. Thus, the multi-agent system is the representation of the problem to be solved with the links and communication between agents reflecting its natural structure. It is worth underlining the fact that this transformation (*i.e.* the agentification) can be completely automatic as it is fully derived from the analytical expression of the problem.

The solving process relies on two continuous simultaneous flow of information: downward (from design variables to criteria) with new values computed by models, and upward (from criteria to design variables) with change-value requests that drive the movements of the design variable in the search space. Intuitively, by emitting requests, criteria agents are "pulling" the different design variables, through the intermediary agents, in multiple direction in order to be satisfied. The system thus converges to an equilibrium between all these "forces", especially in the case of multiple contradicting criteria, which corresponds to the optimum to be found.

The functioning of the system can be divided into two main tasks: problem simulation and collective solving.

Problem simulation can be seen as the equivalent of the analysis of classical MDO method. The agents behavioral rules related to problem simulation concern the propagation of the values of design variables to the models and criteria based on the value. For this part, the agents will exchange *inform* messages which contains calculated values. The "message flow" is top-down: the initial inform messages will be emitted by the variable agents and will be propagated down to the criteria agents.

Collective solving concerns the optimization of the problem. The agent behavioral rules related to collective solving are about satisfying the constraints while improving the objectives. For this part, the agents will exchange *request* messages which contains desired variations of values. The "message flow" is bottom-up: the initial request messages will be emitted by the criteria agents and propagated up to variable agents.

Methodologically, by studying how the system handles specific problems with specific characteristics, we defined different cooperation mechanisms that enable the system to work for all problems with these characteristics. In its current state, the system described here can find the optimum solution only for some classes of problem using the realized mechanisms. Most of these mechanisms are presented in section 7.

We now detail the general behaviors of our five agent types: *model*, *variable*, *output*, *constraint* and *objective* agents.

Model Agent. A *model agent* takes charge of a model of the problem. It interacts with the agents handling its inputs (which can be *variable* or *output agents*) and the *output agents* handling its outputs. Its individual goal is to maintain the consistency between its inputs and its outputs. To this end, when it receives a message from one of its inputs informing it of a value change, a *model agent* recalculates the outputs values of its model and informs its*output agents* of their new value. On the other part, when a *model agent* receives a message from one of its *output agents* it translates and transmits the request to its inputs.

To find the input values corresponding to a specific desired output value, the *model agent* uses an external optimizer. This optimizer is provided by the engineer based on expert domain-dependent knowledge regarding the structure of the model itself. It is important to underline that the optimizer is used only to solve the local problem of the *model agent*, and is not used to solve the problem globally.

Variable Agent. This agent represents a *design variable* of the problem. Its individual goal is to find a value which is the best equilibrium among all the requests it can receive (from models and criteria for which it is an input). The agents using the variable as input can send to it request asking to change its value. When changing value, the agent informs all agents linked to it of its new value.

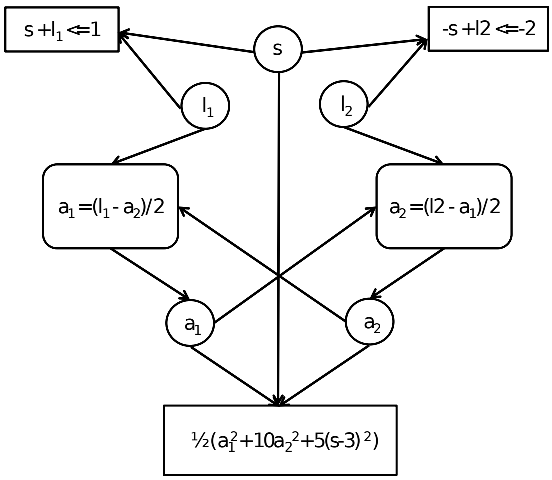
To find its new value, the *variable agent* uses an exploration strategy based on *Adaptive Value Trackers* (AVT) [12]. The AVT can be seen as an adaptation of dichotomous search for dynamic values. The main idea is to change value according to the direction which is requested and the direction of the past requests. While the value varies in the same direction, the variation delta is increased so the value varies more and more. As soon as the requested variation changes, it means that the variable went past the good value, so the variation delta is reduced.

This capability to take into account a changing solution allows the *variable agent* to continuously search for an unknown dynamic target value. This capability is also a requirement for the system to be able to adapt to changes made by the engineer during the solving process.

Output Agent. The *output agent* takes charge of an output of a model. *Output agent* and *variable agents* have similar roles, except *output agents* cannot directly change their value. Instead they send a

$$
\begin{gathered}
a_1 = (l_1 - a_2)/2 \\
a_2 = (l_2 - a_1)/2 \\
min\ \tfrac{1}{2}(a_1^2 + 10a_2^2 + 5(s-3)^2) \\
subject\ to \\
s + l_1 \leq 1 \\
-s + l_2 \leq -2
\end{gathered}
$$

(a) mathematical formulation.



(b) corresponding agent graph.

Figure 3: Alexandrov problem

request to the *model agent* they depend on. In this regard, the *output agent* act as a filter for the *model agent* it depends on, selecting among the different requests the ones it then transmits.

As we will see in the next section, the *output agent* is distinct from the *variable agent* in the way that it can be involved in cycles. A cycle is a situation of interdependent models (that is, models which depend of each other to calculate their outputs).

Constraint Agent. The *constraint agent* has the responsibility for handling a constraint of the problem. When receiving a message from one of its inputs, the agent recalculates its constraint and checks its satisfaction. If the constraint is not satisfied, the agent sends *change value* requests to its inputs.

It should be noted that, to estimate the input values required to satisfy the constraint on its computed value, this agent employs the same technique as the *model agent* (*i.e.* an external optimizer).

Objective Agent. The *objective agent* is in charge of an objective of the problem. This agent sends requests to its inputs aiming to improve its objective, and recalculates the objective when receiving *value changed* messages from its inputs.

This agent uses an external optimizer to estimate input values which would improve the objective, as the model and constraint agents.

The most important point is that each agent only has a local strategy. No agent is in charge of the optimization of the system as a whole, or even of a subset of the other agents. Contrary to the classical MDO methods presented earlier, the solving of the problem is not directed by a predefined methodology, but by the structure of the problem itself. The emerging global strategy is unique and adapted to the problem.

These basic mechanisms are in themselves not sufficient to handle some of the specificities of complex continuous optimization problems such as MDO. We introduced several specific mechanisms used in conjunction with the previously presented behaviors. The mechanisms have been designed to handle specific challenges related to complex continuous optimization, such as conflicting objectives, cycle handling, hidden dependencies *etc.* The exact working of these mechanisms is of little interest here and will not be detailed. The interested reader can refers to [13] for more detailed explanations.

7. Experiments

In this section we present three test cases, Alexandrov Problem, Turbofan Problem and Viennet1, on which our system has been applied, and the experimental results we obtained. In each test case, the MAS consistently converges towards the best (or one of the best) solution.

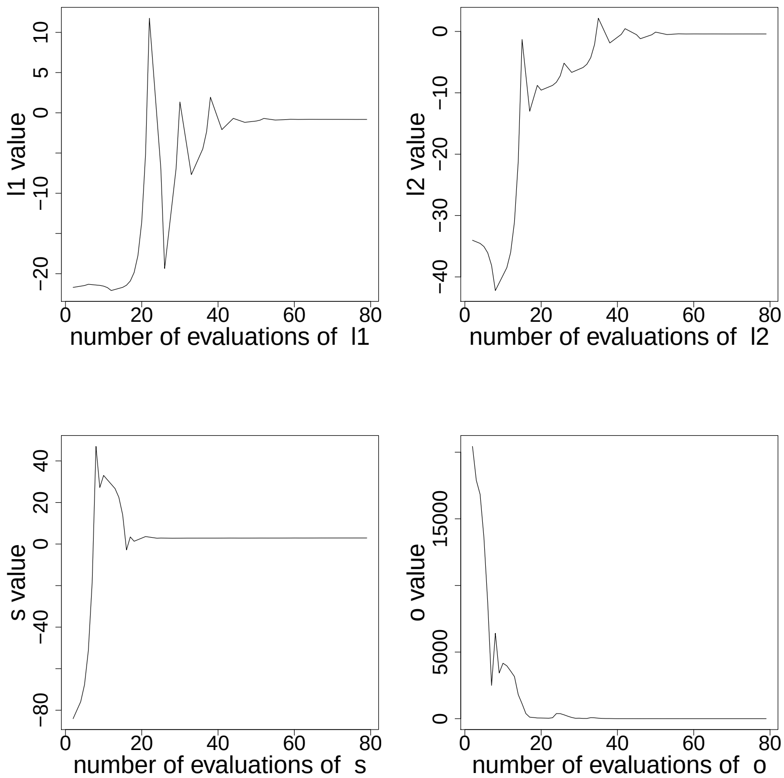


Figure 4: Alexandrov agents behavior

7.1. Alexandrov Problem

Our first test case is inspired from an academic example taken in literature by Alexandrov and al [8]. This simple example presents some of the commons characteristics of MDO problems, such as interdependent disciplines and multiple criteria. In the original article, the example was used to illustrate some properties of Collaborative Optimization, which we presented earlier, in terms of reformulation. While the paper only gave the structure of the problem, we adapted it with meaningful values and equations. The mathematical formulation of the problem and the corresponding agent graph can be seen in Figure 3. Interestingly, the NDMO representation is quite similar to the one adopted by the original authors of the problem.

On Figure 4, the behavior of the *design variables* agents l1, l2 and s, as well the evolution of the objective, can be observed on one instance of the problem with random starting points. On Figure 5, we show the evolution of the objective over 100 iterations with starting points for each *design variable* randomly drawn over the interval [-100; 100]. We can see how the system converges towards the same optimum despite the wildly different initial conditions.

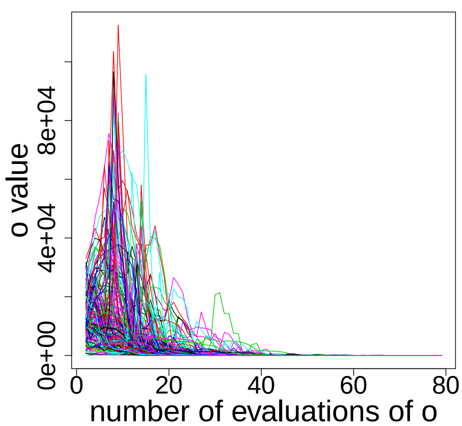


Figure 5: Convergence of the Alexandrov objective for 100 random starting points

7.2. Other Experiments

Table 1: Summary of experiments results for the tests cases

	nb. evaluations to best			average distance to best			
	10%	50%	90%	0% (start)	30%	60%	100% (end)
Alexandrov	29	52	79	13109.169	803.126	5.685	0.059
Turbofan_o1	16	38	50	67.654	14.971	0.743	0.313
Turbofan_o2	10	23	35	23.876	1.853	0.143	0.101
viennet_o1	4	17	31	8.514	0.300	0.025	0.021
viennet_o2	4	15	30	9.412	0.320	0.02	0.02
viennet_o3	5	14	27	10.622	0.063	4.40E-004	1.68E-004

We now briefly present results we obtained on two other test cases, the Turbofan problem and Viennet1. For each case, the system was executed 100 times with random starting points for each *design variable*.

Turbofan Problem. The turbofan problem we introduced in Figure 2 is a based on a real-world optimization problem, albeit simplified for demonstration purpose, concerning the conception of a turbofan engine.

As stated before, the problem concerns two *design variables* pi_c and bpr. pi_c is defined inside the interval [20-40] and bpr inside [2-10]. The model produces three variables $Tdm0$, s and fr. The problem has two objectives, maximizing $Tdm0$ and minimizing s, under the constraint $s \leq 155$ and $fr \geq 4$. The main interest and difficulty of this problem is the existence of two contradictory objectives. As we can see on Figure 6, the system consistently converges toward the same optimal solution.

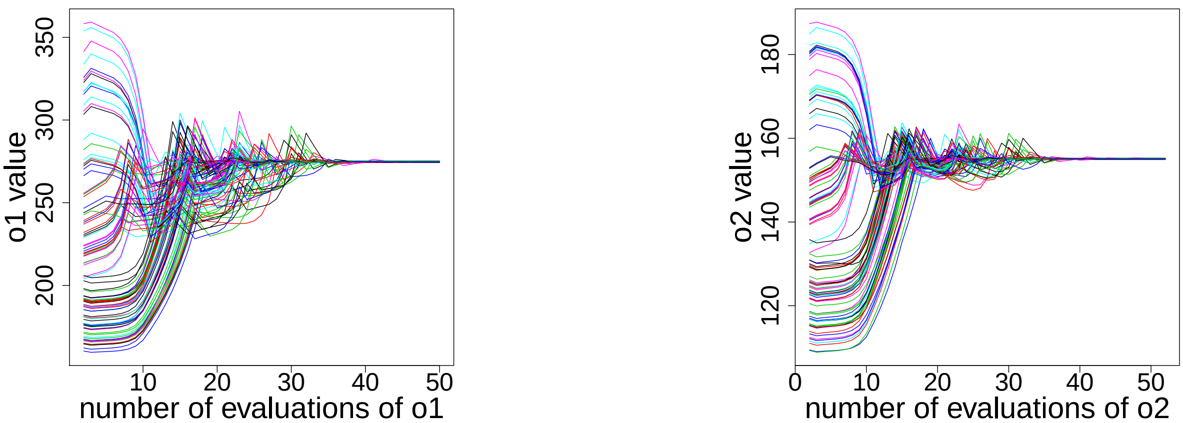


Figure 6: Convergence of the Turbofan objectives for 100 random starting points

Viennet1. The Viennet1 test case is part of a series of problems proposed in [14] to evaluate multicriteria optimization techniques. This problem involves three objectives. Its analytical formulation is:

$$\text{Minimize } o1 = x^2 + (y-1)^2,\ o2 = x^2 + (y+1)^2 \text{ and } o3 = (x-1)^2 + y^2 + 2$$

$$\text{where } x, y \in [-4; 4]$$

Figure 7 illustrates the convergence of the system towards a valid solution.

A summary of these results are presented on Table 1. The first group of values represents the number of evaluations which was needed for respectively 10%, 50% and 90% of the instances to find the best solution. The second group represent the average distance to the best solution (trucated at 10^{-3}) among all instances at different times (0% being the start 100% being the end of the solving in the worst case).

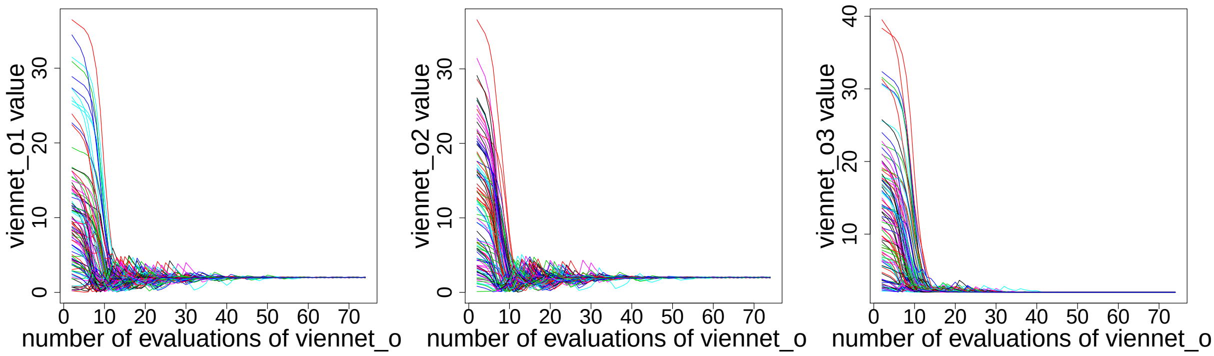


Figure 7: Convergence of Viennet1 objectives for 100 random starting points

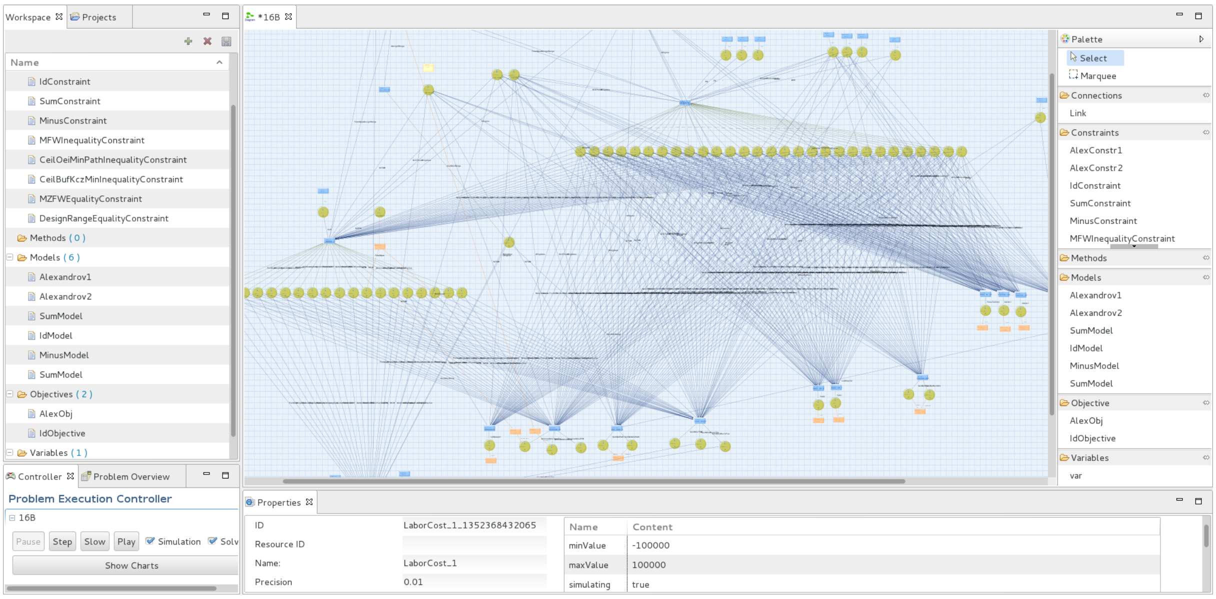


Figure 8: Preliminary aircraft design test case as represented in our prototype

8. Conclusion

We have presented a generic model of numerical optimization problem and an agent-based optimization algorithm. While classical methods often have difficulties to handle complex MDO problems and require the use of specific methodologies, we distribute the problem among the agents in order to keep a low local complexity.

One of our concerns has been to facilitate the work of the engineer and allow him to express his problem in a way which is the most natural to him, instead of restricting him to a specific formulation. By analyzing the different concepts involved in the expression of an MDO problem, we extracted several atomic roles upon which we based the relations between the entities of our system. With these low-level entities, we are able to propose a new formalism we name NDMO. This new formalism can reconstruct a great variety of problems while mirroring their original formulation. Using this formalism, we proposed an agent-based optimization algorithm integrating MDO-specific mechanisms.

We have exposed here the results of preliminary experiments using simple but representative problems in order to validate the soundness of our approach. Obviously these test cases are a first step to demonstrate the validity of the MAS we propose. We continue to work with our industrial partners in order to show the scalability of our approach on more complex real world-based problems. As an example of the problems we are currently studying, the figure 8 represents a preliminary aircraft design problem (as visualized by our prototype tool) which involves sixteen disciplines and a hundred variables.

Our goal is to make a system that grows not only with the complexity of the problem but also with the needs of the engineer. This is why our approach can, by design, easily be interfaced with any local optimization method. In the same idea, one of our next goals is to integrate into our system the capability to handle and propagate uncertainties among the different parts of the problem. Another line of research is about efficiently and interactively exploring the Pareto front of a problem.

Acknowledgements. This work has been supported by French National Research Agency (ANR) through COSINUS program with ANR-09-COSI-005 reference.

References

[1] Jaroslaw Sobieszczanski-Sobieski, Raphael T. Haftka, Jaroslaw Sobieszczanski-sobieski, and Raphael T. Haftka. Multidisciplinary aerospace design optimization: Survey of recent developments. *Structural Optimization*, 14:1–23, 1996.

[2] Gerhard Weiss, editor. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, 1999.

[3] G.D.M. Serugendo, M.P. Gleizes, and A. Karageorgos. Self-organising systems. *Self-organising Software: From Natural to Artificial Adaptation*, page 7, 2011.

[4] E.J. Cramer, JE Dennis Jr, P.D. Frank, R.M. Lewis, and G.R. Shubin. Problem formulation for multidisciplinary optimization. *SIAM Journal on Optimization*, 4(4):754–776, 1994.

[5] S.I. Yi, J.K. Shin, and GJ Park. Comparison of mdo methods with mathematical examples. *Structural and Multidisciplinary Optimization*, 35(5):391–402, 2008.

[6] Ilan M. Kroo, Steve Altus, Robert D. Braun, Peter J. Gage, and Ian P. Sobieski. Multidisciplinary optimization methods for aircraft preliminary design. *AIAA 5th Symposium on Multidisciplinary Analysis and Optimization*, September 1994. AIAA 1994-4325.

[7] J. Sobieszczanski-Sobieski, J.S. Agte, and R.R. Sandusky. *Bi-Level Integrated System Synthesis*. NASA Langley Technical Report Server, 1998.

[8] N.M. Alexandrov and R.M. Lewis. Analytical and computational aspects of collaborative optimization for multidisciplinary design. *AIAA journal*, 40(2):301–309, 2002.

[9] R.E. Perez, H.H.T. Liu, and K. Behdinan. Evaluation of multidisciplinary optimization approaches for aircraft conceptual design. In *AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Albany, NY*, 2004.

[10] R. Stranders, A. Farinelli, A. Rogers, and N.R. Jennings. Decentralised coordination of continuously valued control parameters using the max-sum algorithm. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 601–608. International Foundation for Autonomous Agents and Multiagent Systems, 2009.

[11] E. Kaddoum. *Optimization under Constraints of Distributed Complex Problems using Cooperative Self-Organization*. PhD thesis, Université de Toulouse, Toulouse, France, november 2011.

[12] S. Lemouzy, V. Camps, and P. Glize. Principles and properties of a mas learning algorithm: A comparison with standard learning algorithms applied to implicit feedback assessment. In *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2011 IEEE/WIC/ACM International Conference on*, volume 2, pages 228 –235, aug. 2011.

[13] Tom Jorquera, Jean-Pierre Georgé, Marie-Pierre Gleizes, Nicolas Couellan, Victor Noel, and Christine Régis. A natural formalism and a multi-agent algorithm for integrative multidisciplinary design optimization *(to be published). AAMAS 2013 workshop: International Workshop on Optimisation in Multi-Agent Systems*, may 2013.

[14] R. Viennet, C. Fonteix, and I. Marc. Multicriteria optimization using a genetic algorithm for determining a pareto set. *International Journal of Systems Science*, 27(2):255–260, 1996.