# A novel approach to MDO using an adaptive multi-agent system

Tom Jorquera, Jean-Pierre Georgé, Marie-Pierre Gleizes, Christine Régis, Pierre Glize

# A Novel Approach to MDO using an Adaptive Multi-Agent System

Tom Jorquera, Jean-Pierre Georgé, Marie-Pierre Gleizes, Christine Régis, Pierre Glize
*IRIT (Institut de Recherche en Informatique de Toulouse), Paul Sabatier University, Toulouse, France*
{*jorquera,george,gleizes,regis, glize*}*@irit.fr*

Keywords: Self-Adaptation, Multi-Agent System, Multidisciplinary Optimization, Integrative Design

Abstract: MultiDisciplinary Optimization (MDO) problems represent one of the hardest and broadest domains of continuous optimization, often too complex to be tackled by classical optimization methods.
We propose an original approach for taking into account this complexity using a self-adaptive multi-agent system where each elements of the problem become an agent in charge of a small part of the problem.

## 1 INTRODUCTION

Multidisciplinary optimization (MDO) problems are a specific class of optimization problem where the number of variables and disciplines involved is to important to directly apply classical optimization methods. Most of the existing approaches concentrate on separating the problem in distinct subproblems and using standard optimization methods on these subproblems while trying to maintain consistency among the variables shared by the subproblems. Basically these methods try to help the user to find an optimization process which reduces the complexity of the problem. However, a shortcoming of these MDO methods is that they require a strong expert knowledge of both the problem to be solved and the method which is applied, in order to obtain interesting results.

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

Figure 1: Mathematical formulation of an optimization problem.

We propose a radically different approach where one only requires to express the problem, and the system tries to automatically solve the problem without trying to determine in advance a specific process. On figure 1, we can see the classical, analytic formulation of an optimization problem. However, it is also possible to express such a problem as a relationship graph, as represented on figure 2. Our approach is to create for each entity of this graph an autonomous intelligent agent, whose behavior is defined by local cooperatives rules and goals. The global optimization process can be said to *emerge* from these local interactions between agents. The user can let the system try to optimize the problem on its own, or he can interact with the system during the solving in order to guide it toward a new search point or to change the problem to be solved. The system is able to integrate and adapt at runtime to these changes. We call this vision of MDO *Integrative and Interactive Design*.

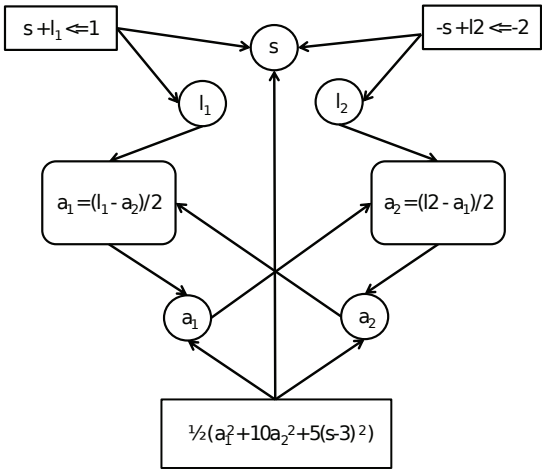


Figure 2: Equivalent graph representation.

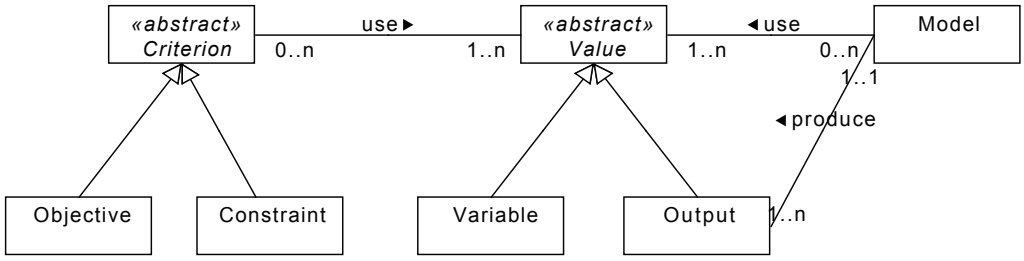


Figure 3: Class diagram of MDO problems

## 2 PROBLEM MODELING

The first step is to obtain an explicit and coherent representation of what are the different elements present in a MDO problem. To this end we worked with engineers to provide a consensual MDO model and to extract the important entities which always arise in these types of problem. As an example, a MDO problem usually occurs in a generic domain (aeronautics, engine, ...) which can be represented through several models. So the entity *model* will be used to import and to use different models of a domain. These entities and the relations between them are represented by the class diagram in figure 3. We will now examine them briefly.

**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 the context of an aircraft design, an example of model could be the one calculating the total weight of the aircraft from the weight of the different parts.

**Design Variables** These are the inputs of the problem and can be adjusted freely (within their defining boundaries). The engineer's 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 variable can be shared among several models, objectives and constraints.

Some very simple examples of design variables could be, for the design of an aircraft, the length of the wings, the mass of the cockpit, the fuel tank capacity.

**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.

Using the example above, the total weight of the aircraft is an output variable of our model. It is worth to notice that output variables of a model, as well as design variables, can be taken as input of other models.

**Constraints** These are strict restrictions on 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. For example, the total weight of an aircraft must be lower than the total lifting capacity of the engines and greater than zero.

**Objectives** The goals to be optimized. In the general case, different objectives are often contradictory.

Constraints and objectives are usually regrouped under the more general term of optimization criteria.

## Existing Approaches

A major part of the research in the field focuses on providing strategies for reformulating the problem, in a way that makes possible to optimize different subparts while trying to keep them consistent among each other. To this end, a variety of techniques has been developed, such as Multi-Disciplinary Feasible Design [Dennis Jr et al., 1993], Collaborative optimization [Kroo et al., 1994], Bi-Level Integrated System

Synthesis [Sobieszczanski-Sobieski et al., 1998] for example.

One of the major shortcomings of these 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 involved models as well as of the chosen method. This is required to be able to correctly reformulate the models accordingly 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 involve 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.

This is to answer these shortcomings that we propose a completely generic approach, which requires only from the user to express the problem in the way which is the most natural to him. Instead of relying on a predefined strategy to reduce the complexity of the problem, our novel approach uses a multi-agent system where the agents collectively solve it.

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) framework, where the definition domains of the variables are discrete and finite. MDO problems, by contrast, are inherently continuous problems.

Moreover, the existing agent-based optimization techniques for DCOP present similar shortcomings to MDO methods, in the sense that they require a strong expertise to be efficiently applied [Kaddoum, 2011].

# 3 A MULTI-AGENT SYSTEM FOR MDO

Each variable, each objective, each constraint, each model is associated with its own representative agent and thus the multi-agent system is the representation of the problem to be solved with the links and communication between agents reflecting the natural structure of the problem. It is worth to underline the fact that this transformation is completely automatic. When the user has expressed the problem in the terms described before there is no extra work on his part to build the multi-agent system, which is fully derived from the expression of the problem.

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 we 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.

**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 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 should translate and transmit the request to its inputs.

To find the input values corresponding to a specific output value, the model agent can use an external optimizer. This optimizer can be provided by the engineer based on expert knowledge regarding the structure of the model. If it is not the case, the model agent still can make some estimations by observing the output variations when the inputs change. Doing so, the agent is able to estimate a local correlation coefficient between each input and output. Using these coefficients, the agent can then make a rough estimation of some input values corresponding to a specific output value. While this method is less precise and efficient than calling a specialized optimizer it is always available, even in the case where the engineer knows no adequate specific optimization technique to apply to the model.

**Variable Agent** This agent represents a design variable of the problem and is responsible for giving a value to its variable. Its individual goal is to find a value which will be the best equilibrium among the requests of the models and criteria which are using it as an input. The agents to which it is linked can request it to change its value to another one. When its value changes (because it received a request or by a manual change of the engineer), this agent informs all the agents which are linked to it (its neighbors) of its new value.

One limit of this simple mechanism is that, if the search space of the problem is large, the system can take a long time to converge towards the solution. To mitigate this we use an exploration strategy based on *Adaptive Value Trackers* (AVT). The AVT is a tool introduced in the work of [Lemouzy et al., 2011] and

can be seen as an adaptation of dichotomous search for dynamic values. The main idea is to change value according not to the requested new value but 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.

By adjusting the variation by a factor of 2, we can find a correct value with comparable efficiency to the one of dichotomous search.

While changing value based not on the value requested but on the direction can seem paradoxical, it must be recalled that, since no agent has a global view of the system, the requests made by the agents will often be approximate, so the agents need to iterate many times. If the search space in large, the system could take time to converge towards the solution. By using a near-dichotomous strategy, we greatly accelerate this convergence.

This capability to take into account a dynamic value is of utmost importance here since during solving, agents can aim for a value which, while apparently good, will be revealed as inadequate later when another part of the system will have converged to a different solution. On another part, 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. This role is similar to the one of a variable agent, except it cannot directly change its value, but must instead send a request to the model agent handling the model from which its output depends. In this regard, the output agent will act as a filter for the model agent it depends on, selecting among the different requests the ones it will transmit.

**Constraint Agent** The constraint agent has the responsibility for handling a constraint of the problem. When receiving a message from an agent handling one of its inputs, the agent recalculates its constraint and checks its satisfaction. If the constraint is not satisfied anymore, the agent sends requests to its inputs asking them to take values with whom the constrain will be satisfied again.

It should be noted that, to estimate the input values required to satisfy the constraint, this agent employs the same technique than the model agent, using an external optimizer if available, or basing itself on the estimated correlations of the inputs to the value of the constraint.

**Objective Agent** The objective agent is in charge of the improvement of one of the objectives of the problem. This agent sends requests to its inputs aiming to improve its objective, and recalculates the objective when receiving messages from its inputs informing it of some changes.

As the model and constraint agent, to estimate input values which would improve the objective, this agent uses an external optimizer or the observed correlations of the inputs to the objective.

# 4 APPLICATIONS

This Multi-Agent System has be developed in the context of the ID4CS[1] project, involving 9 academic and industrial partners, including among others Airbus and Snecma (Safran Group), and is integrated into a proof-of-concept user interface based on the Eclipse Modeling Framework. While the system and interface are still in development. Some demos of the tool are availables at `www.irit.fr/id4cs/index.php?page=demonstration`.

# ACKNOWLEDGEMENTS

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

# REFERENCES

Dennis Jr, E., Frank, P., Lewis, R., and Shubin, G. (1993). Problem formulation for multidisciplinary optimization.

Kaddoum, E. (2011). *Optimization under Constraints of Distributed Complex Problems using Cooperative Self-Organization*. PhD thesis.

Kroo, I., Altus, S., Braun, R., Gage, P., and Sobieski, I. (1994). Multidisciplinary optimization methods for aircraft preliminary design. *AIAA paper*, (94-4325).

Lemouzy, S., Camps, V., and Glize, P. (2011). 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.

Sobieszczanski-Sobieski, J., Agte, J., Sandusky, R., and Center, L. R. (1998). *Bi-level integrated system synthesis (BLISS)*. Citeseer.

[1] Integrated Design for Complex Systems`http://www.irit.fr/id4cs`