Agent-Based Natural Domain Modeling for Cooperative Continuous Optimization

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

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

Abstract. While multi-agent systems have been successfully applied to combinatorial optimization, very few works concern their applicability to continuous optimization problems. In this article we propose a framework for modeling a continuous optimization problems as multi-agent system, which we call NDMO, by representing the problem as an agent graph, and complemented with optimization solving behaviors. Some of the results we obtained with our implementation on several continuous optimization problems are presented.

Keywords: Multi-Agent System, Continuous Optimization, Problem Modeling

1 Introduction

One of the major areas in which Multi-Agent Systems (MAS) have been successfully applied is the domain of combinatorial optimization. Using DCOP (Distributed Constraint Optimization Problem), numerous agent-based algorithms have been proposed in the field. However, the applicability of MAS to another optimization domain, continuous optimization, is still mostly unexplored, despite the existence of whole categories of complex optimization problems. Continuous optimization presents interesting challenges concerning the scalability of optimization methods. The various topologies of optimization problems couple has lead to highly specialized methods targeting specific problems types. However, the industrial development created a need for solving highly complex optimization problems involving heterogeneous models and complex interdependencies. Some methods adapted to handle such types of problems have been proposed, but are often unwieldy and lack flexibility. Arguably, the scalability and adaptability properties of MAS indicate their strong potential for the field.

We present in this article a way of modeling a continuous optimization problem as an agent graph, akin to how DCOP is used to model combinatorial optimization problems, named Natural Domain Modeling for Optimization (NDMO). We propose to complement this graph representation with specific agent behaviors tailored for continuous optimization.

In the next part (section 2), we begin by introducing complex continuous optimization methods and the existing works on MAS for optimization. We

then present in section 3 a new generic agent-based modeling for continuous optimization problems. To complement this modeling, we present in section 4 specific agent behaviors for continuous optimization as well as some results, and finish by perspectives about future improvements based on the current work.

2 Context

2.1 Complex Continuous Optimization Methods

Continuous optimization problems are optimization problems where the shape of the search space is defined by one (or several) continuous function. Contrary to combinatorial optimization, where the difficulty lies in the combinatorial explosion of possible states, continuous optimization is more concerned with the efficient exploration of search spaces of heterogeneous (and sometimes exotic) topology. However, a common point between these two fields is the fact that the problems encountered can vary from very simple ones to problems of tremendous complexity.

A good example of complex continuous optimization problems are multi-disciplinary optimization (MDO) problems. Sobieszczansky-Sobieski and Haftka proposed to define MDO *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.

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. For example Multi-Disciplinary Feasible Design, considered to be one of the simplest methods [2], 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 [3]. Other approaches, such as Collaborative Optimization [4] or Bi-Level Integrated System Synthesis [5], are said bi-level. They introduce different levels of optimization [6], 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 [7],and can have large computation time [3].

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 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.
It should be clear from these shortcomings that MDO methods are not universal optimization methods but specialized tools to be applied in specific contexts. However the nature of the problems they aim to solve is in this regard not different from the one of "smaller" continuous optimization problems. Their usefulness lies in the fact that classical optimization techniques are not able to handle the increased complexity incurred by the combinatorial explosion of the interactions between the disciplines and their subproblems. This statement brings to light the fact that one of the fundamental difficulties of continuous optimization problems concerns the scalability of the approaches, which in itself is a strong indication of the potential of MAS techniques for this domain.

2.2 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 framework [8].

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 [9], it 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, a property which is not true for continuous optimization problem in general. Moreover 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 in most cases.

DCOP is a very successful framework since it provides a common ground for researchers to propose and to compare new agent-based algorithms for combinatorial optimization. No equivalent has been proposed for continuous optimization using MAS, an obvious impediment to the development of the field.

3 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).

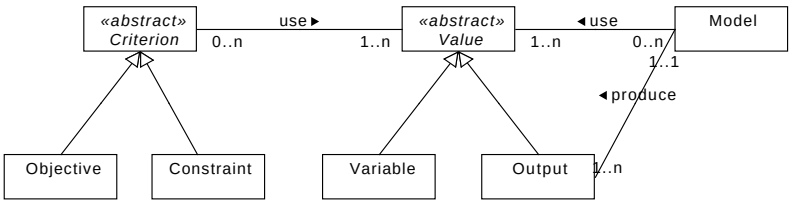


Fig. 1: Class diagram of continuous optimization problems.

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*. These entities and their relations are represented by the diagram in Fig. 1.

To illustrate how an optimization problem is modeled, we use a simplified Turbofan optimization problem, provided by one of our industrial partners. In Fig. 2, the analytic expression of this optimization problem is given,with the corresponding relations graph. 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.

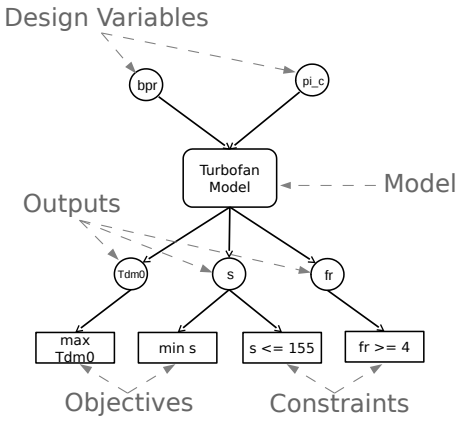
Let's now see in more details the roles of each of these fives entities: *model*, *variable*, *output*, *constraint* and *objective*.

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.

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

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

(a) mathematical formulation.



(b) corresponding entities graph.

Fig. 2: Turbofan problem.

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 models, constraints as well as 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.

4 Agent Behavior and Experiments

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.

A summary of the basic principles of each agent type is given in Algorithm 1.

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.

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

Algorithm 1 Agents Behaviors

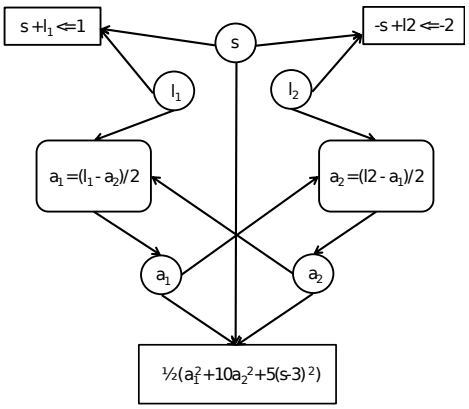
```
procedure Model Agent Behavior
    loop
        analyze received messages
        if received new information messages then
            recalculate outputs
            inform depending agents
        end if
        if received new requests then
            use optimizer to find adequate inputs
            propagate requests to input agents
        end if
    end loop
end procedure

procedure Variable Agent Behavior
    loop
        analyze received messages
        if received new requests then
            select most important
            adjust value
            inform depending agents
        end if
    end loop
end procedure

procedure Output Agent Behavior
    loop
        analyze received messages
        if received new information messages then
            update its value
            inform depending agents
        end if
        if received new requests then
            select most important
            transmit selected request to model agent
        end if
    end loop
end procedure

procedure Constraint/ Objective Agent Behavior
    loop
        analyze received messages
        if received new information messages then
            update its value
            use optimizer to find adequate inputs
            send new requests to variable/output agents
        end if
    end loop
end procedure
```

is of little interest here and will not be detailed. The interested reader can refers to [10] for more detailed explanations.

In order to validate our prototype, we experimented on several test cases. We present here synthetic results on three of them: Alexandrov, Turbofan and Viennet1 test cases.

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

(a) mathematical formulation.



(b) corresponding agent graph.

Fig. 3: Alexandrov problem

The Alexandrov test case is inspired from an academic example taken in literature by Alexandrov and al [6]. 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 Fig. 3. Interestingly, the NDMO representation is quite similar to the one adopted by the original authors of the problem.
The Viennet1 test case is part of a series of problems proposed in [11] to evaluate multi-criteria optimization techniques. This problem involves three objectives.

In each test case, the MAS is executed 100 times with random starting points for each *design variable* and consistently converges towards the best (or one of the best) solution. As the performances of the system are not central to the topic at hand, we will only present some summarized results illustrating the convergence of the system. Once more, the interested reader can refer to [10] for a more detailed analysis of the performances.

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).

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

5 Conclusion

We have presented a model of numerical optimization problem and an agent-based optimization algorithm. While classical methods often have difficulties to handle complex continuous 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 a continuous 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 way to represent continuous optimization problems as agents graphs ,which we name NDMO. This representation can reconstruct a great variety of problems while mirroring their original formulation. We applied NDMO by proposing a MAS capable of solving continuous optimization problem.

In the same way DCOP is a framework used by several MAS techniques to translate combinatorial optimization problems, this agent graph representation is not restricted to the specific MAS we presented but can be used as a base for multiple different techniques. With this work we achieved a proof-of-concept for the mostly unexplored field of continuous optimization using MAS.

We continue to work on the validation of our approach, as well as of the performances of our MAS, and already obtained interesting preliminary results concerning the extension of this modeling for uncertainty propagation, as well as optimization of scaled-up industrial test cases.

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

References

1. Sobieszczanski-Sobieski, J., Haftka, R.T.: Multidisciplinary aerospace design optimization: Survey of recent developments. Structural Optimization **14** (1996) 1–23
2. Cramer, E., Dennis Jr, J., Frank, P., Lewis, R., Shubin, G.: Problem formulation for multidisciplinary optimization. SIAM Journal on Optimization **4**(4) (1994) 754–776
3. Yi, S., Shin, J., Park, G.: Comparison of mdo methods with mathematical examples. Structural and Multidisciplinary Optimization **35**(5) (2008) 391–402
4. Kroo, I.M., Altus, S., Braun, R.D., Gage, P.J., Sobieski, I.P.: Multidisciplinary optimization methods for aircraft preliminary design. AIAA 5th Symposium on Multidisciplinary Analysis and Optimization (September 1994) AIAA 1994-4325.
5. Sobieszczanski-Sobieski, J., Agte, J., Sandusky, R.: Bi-Level Integrated System Synthesis. NASA Langley Technical Report Server (1998)
6. Alexandrov, N., Lewis, R.: Analytical and computational aspects of collaborative optimization for multidisciplinary design. AIAA journal **40**(2) (2002) 301–309
7. Perez, R., Liu, H., Behdinan, K.: Evaluation of multidisciplinary optimization approaches for aircraft conceptual design. In: AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Albany, NY. (2004)
8. Modi, P.J., Shen, W.M., Tambe, M., Yokoo, M.: An asynchronous complete method for distributed constraint optimization. In: International Conference on Autonomous Agents: Proceedings of the second international joint conference on Autonomous agents and multiagent systems. Volume 14. (2003) 161–168
9. Stranders, R., Farinelli, A., Rogers, A., Jennings, N.: 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, International Foundation for Autonomous Agents and Multiagent Systems (2009) 601–608
10. Jorquera, T., Georgé, J.P., Gleizes, M.P., Couellan, N., Noel, V., Régis, C.: 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)
11. Viennet, R., Fonteix, C., Marc, I.: Multicriteria optimization using a genetic algorithm for determining a pareto set. International Journal of Systems Science **27**(2) (1996) 255–260