Optimization of Complex Systems in Photonics by Multi-agent Robotic Control

Quentin Pouvreau, Jean-Pierre Georgé, Carole Bernon, Sébastien Maignan

Optimization of Complex Systems in Photonics by Multi-Agent Robotic Control

Quentin Pouvreau[1,2][0000−0003−0700−5149], Jean-Pierre Georgé[1][0000−0002−4255−236X], Carole Bernon[1][0000−0002−7602−141X], and Sébastien Maignan[1]

[1] IRIT, Université de Toulouse, CNRS, Toulouse INP, UT3, Toulouse, France
www.irit.fr
[2] ISP System, Vic-en-Bigorre, France
www.isp-system.fr
Quentin.Pouvreau@irit.fr, Jean-Pierre.George@irit.fr, Carole.Bernon@irit.fr, Sebastien.Maignan@irit.fr

Abstract. The optimization of complex industrial systems represents a class of difficult problems, due to their embodiment in the physical world, and whose search spaces are disrupted, non-linear and potentially vast. Their parametrization relies on the combination of many variables, each change generally impacting the whole system. Mathematical approaches are limited by the fact that the models are too coarse or non-existent, and by the imprecision of the measurements and the machining of components of such systems. The action cost of the system calls into question population-based heuristics and swarm intelligence where each individual must be tested. We propose a multi-agent approach allowing a global / black-box modeling of the system in terms of input variables and objectives, as well as an agnostic and continuous adaptive optimization, based on sensor feedbacks from the running system.

Keywords: Collective Problem Solving · Self-Organization · Continuous Multi-Objective Optimization · Robotic Control · Photonics

1 Problem description

This study deals with continuous optimization problems arising from complex robotic industrial applications. Such problems have a wide variety of external constraints like limited resolution time or number of moves, predetermined components positions, etc. Regardless of the specifics of the applications, we choose to consider these problems as black-box systems to be optimized, given sensor feedback.

We propose a self-adaptive multi-agent system (MAS) for optimization able to tackle multiple robotic applications, with two main advantages:

1. A natural representation of the domain, easily observable and explainable, where each agent represents a variable of the problem and can perceive and interact with the real topology of the problem.

2. A collective resolution process during which the agents continuously adapt to feedback, whether from a sensor or an expert interacting with the system, and can therefore integrate perturbations, imperfections, etc.

The application domain we focus on is the Photonics domain as illustrated in Figure 1. The main goal is to correctly align optical components to get certain beam properties. Input parameters of the problem are translation and rotation axes values of the optical components. We aim at developing an algorithm able to find a set of satisfactory solutions in a minimum number of iterations, and able to deal with measurement and machining imprecisions.

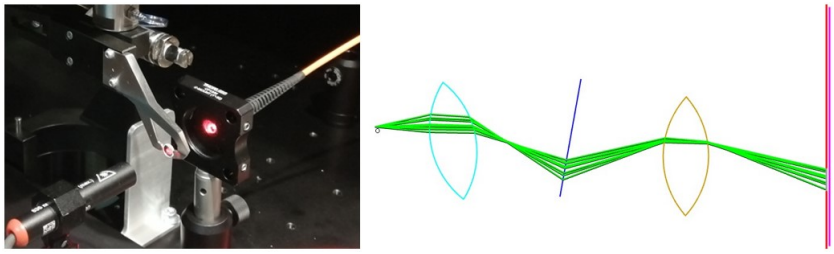

Fig. 1. Example of a real photonic application on which ISP System is working with the robot (left) and a schematic of a set of lenses that need to be positioned (right)

In this paper, we consider that optimizing a Photonics system is a continuous multi-objective optimization (MOO) problem. The following section presents the main approaches used in this field before focusing on the most suitable ones. The next section then describes the MAS we propose, followed by experiments. Finally we discuss the results and conclude.

2 Positioning

Exact methods are able to find optimal solutions but at the expense of high computation time and are thus not suited for large-scale optimization problems [5]. Moreover, it is often not possible to precisely model real systems taking into account all their particularities. The field of geometrical optics is subject to several types of aberrations between theoretical calculations and practical observation. Defects of optical components have also to be taken into account. This limits the relevance of learning approaches, which need repeatability.

Metaheuristic optimization algorithms are widely used to solve MOO problems, as they can find multiple optimal solutions in a single run, and improve the ratio between accuracy and computational cost. They are problem-independent optimization techniques that provide, if not an optimal solution, a set of satisfactory solutions by iteratively exploring and exploiting the search spaces stochastically [10]. The following sections focus on these algorithms.

2.1 Population-Based Heuristics

These approaches constitute a large part of the state of the art of optimization metaheuristics. The general principle is to simultaneously process a population of solutions distributed (randomly or not) in the search space. The population can evolve and select the best solutions iteratively according to the Darwinian principle, or converge towards an optimum by following a set of influence rules.

These algorithms can also be used in hybrid solutions alongside more classical algorithms or exact methods [7] to balance their weaknesses.

Being exhaustive on the state of the art in this area (e.g., Genetic Algorithms [4] and Swarm Intelligence algorithms [8] together have more than 3000 publications per year [12]) is difficult [11], however, such algorithms must evaluate a relatively large number of candidates in order to create a good population of solutions. To do so, the robot / bench must configure the entire experimental framework for each proposed new candidate solution in the population. The cost of activating a real robotic system is heavy compared to the algorithmic time. The computation of all candidate solutions is therefore prohibitive. A more suitable strategy would be an adaptive algorithm modifying and proposing for testing a unique configuration for each feedback (i.e. an optimization process forming a unique trajectory in the search space).

2.2 Multi-Agent Problem Solving

The MAS paradigm relies on a decentralized approach, based on self-organization mechanisms [14], where the computational task is distributed over the agents, which are virtual or physical autonomous entities. Each agent has only a local view of the problem it solves, corresponding to a local function. The global function of a problem then results from the *composition* of all these local functions. This feature allows to easily distribute the computational tasks in the solving process and thus to reduce the computational costs. This is why MAS approaches are preferred when centralized approaches have limited flexibility.

MAS are used in a wide variety of application areas of distributed optimization including power systems, sensor networks, smart buildings, smart manufacturing [13], etc. Many of these algorithms are based on a combinatorial logic such as the well-known Distributed Constraint Optimization Problem (DCOP) framework [1].

DCOP was originally developed under the assumption that each agent controls exactly one variable. This model was designed for problems where the difficulty lies in the combination of multiple constraints. Only a few works have tried to extend the DCOP model to continuous optimization problems [3], [2]. Fundamentally DCOP requires a problem to be easily decomposable into several cost functions (to be able to evaluate partial assignments) and the relations between the states of the variables are supposed to be known. These major assumptions do not hold for complex continuous optimization problems, where the complexity of the models and their interdependencies mean that this detailed knowledge is not available in most real-life cases.

3 A MAS for Optimizing Complex Systems

When solving complex continuous problems, existing techniques usually require a transformation of the initial formulation, in order to satisfy certain requirements of the technique to be applied. Besides the fact that the correct application of these changes can be a demanding task for the designers, imposing such modifications changes the problem beyond its original and natural meaning. What we propose here is an agent-based modeling where the original structure/meaning of the problem is preserved. Indeed, it represents the formulation that is the most natural and the easiest to handle for the expert. We call it *Natural Domain Modeling for Optimization* (NDMO) [6].

3.1 Natural Domain Modeling

In order to represent the elements of a generic continuous optimization model, we have identified five classes of interacting entities: *design variables*, *models*, *outputs*, *constraints*, and *objectives*. In short: given the values of the design variables, some models will compute output values, and following models will compute further outputs and so on until the constraints and objectives can be computed, in a kind of computational network. Three types of elements are *agentified*: design variables (which must be optimized and thus constitute the solving process), constraints and objectives (i.e. the requirements or problem statements).

To take into account these requirements, the solving process relies on a specific measure called *criticality*. This measure represents the state of dissatisfaction of the agent with respect to its local goal. The role of this measure is to aggregate in a single comparable value all relevant indicators concerning the agent's state. Having this single indicator simplifies the reasoning of the agents. Each agent is responsible for estimating its own criticality and providing it to the other agents. However, the system designer has the task of providing the agents with adequate means to compute their criticality, as for example with a "barrier" or "logistics" function (Eq. 1 and Fig. 2).

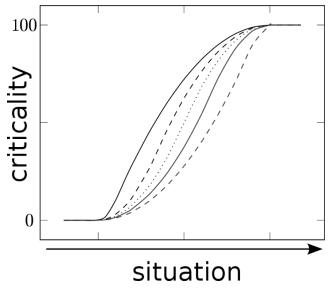

Fig. 2. Typical criticality shapes

$$crit(x) = \frac{C}{1 + a * e^{-rx}} \qquad (1)$$

Where C is the maximum critical value, a is a translation factor of the function determining the y-intercept and r enables to adjust the acceleration of the criticality curve and in a second hand determines the maximum tolerated distance to the objective.

3.2 Internal State and Behavior of the Agents

The approach we propose is a single-solution (also called trajectory) algorithm. It iteratively modifies the solution's parameters and consequently selects a trajectory in the search space. Our implementation is based on the AMAK framework [9]. Regarding a complex system to optimize, some inputs affecting its behavior are considered design variables of the problem and outputs of interest can have an objective value or some constraints or both. As explained above we agentify design variables, objectives and constraints. Design variables are handled by solver agents. In the same way that we want to be as close as possible to the objectives, we must move away as much as possible from the constraints, and then objectives and constraints are handled by objective agents. Before going into details, the main principle of the resolution process is to make the agents cooperate to reduce the highest criticality at each iteration. From their combined actions will emerge the trajectory to more satisfactory solutions.

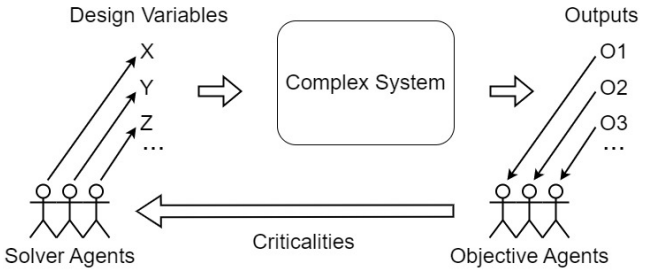

Fig. 3. Agentification process of a complex system to optimize

Objective agents Each objective agent is responsible for one objective / constraint and normalizes the associated feedback (the gap to expert-given ideal value) into criticality (see Eq. 1 and Fig. 2). It then propagates this criticality to all the solver agents who will act to lower it as much as possible.

Solver agents A solver agent has no criticality but has a value, corresponding to the design variable it is assigned to, and a memory. This memory is composed of a predefined number of previous observations cycles. Each entry of this memory contains the agent's value at the corresponding cycle, the set of objective agents criticalities, the entropy of the system, i.e. the number of solver agents having acted. The goal of a solver agent is to respond to the criticalities of the objective agents by adapting its value. Solver agents generally choose to help the most critical of the objectives so a cooperation may emerge. However, this mechanism is moderated by the local variations of the problem. To this end, a solver agent has a Perception-Decision-Action lifecycle:

- **Perception** : the agent records an observation of the last cycle in its memory and updates data needed during the decision phase.
- **Decision** : the agent may decide in 4 possible ways
  - If the agent has an empty memory or has never acted, then it acts in a *stochastic* way.
  - Otherwise the agent enters a *reactive* decision phase detailed below.
  - If the agent did not decide, it enters a *cognitive* decision phase detailed below.
  - Potentially, the agent can operate a *stochastic stop* to look whether its impact is real and not just a result of other agents impacts.
- **Action** : The agent changes or not its value according to its decision.

The resolution of the problem results from the capacity of the agents to make the right decision, therefore this phase is at the heart of the process. The difficulties encountered are mainly due to the problem itself: almost all the design variables have an impact on all the objectives, namely all the criticalities, and this in a non-linear way. The parameters are therefore strongly interconnected: the current value of one agent displaces the target value of one or several others. The system is therefore complex and the agents can collectively interfere with each other. At this point, the optimization problem becomes a cooperation problem.

The decisions of an agent are subject to a stochastic momentum mechanism, i.e. an agent will repeat its action for a random number of cycles equal to the momentum. The maximum value of the momentum is configurable. This mechanism of momentum allows to desynchronize the agents in order to avoid them being trapped in what we call non-cooperative synchronizations (essentially, when agents try to "help" at the same time, thus hindering each other). More importantly, this desynchronization individualizes the average perception of each agent over several cycles, allowing them to isolate their average impact on criticalities. Thus, if a solver agent sees that its impact on the most critical objective is negligible, it may decide to help another objective, potentially releasing a constraint on the improvement of the most critical.

Reactive decision If an agent observes that its target criticality decreases beyond a configurable threshold, it will increase its momentum. This rule is generally useful to converge quickly when the search space is locally regular or even linear. On the contrary, if the highest criticality increases beyond the same threshold, the agent will reduce its momentum so the next cognitive decision will come sooner, with updated data including these "error" cycles. The goal of this reactive method is to avoid more expensive processes when the current region of the search space is not ambiguous.

Cognitive decision When the momentum of its last decision drops to zero, the agent will process the data it has recorded during the successive perception phases. This more cognitive process consists in interpolating the variation of the current target criticality according to its own value. The goal is to identify

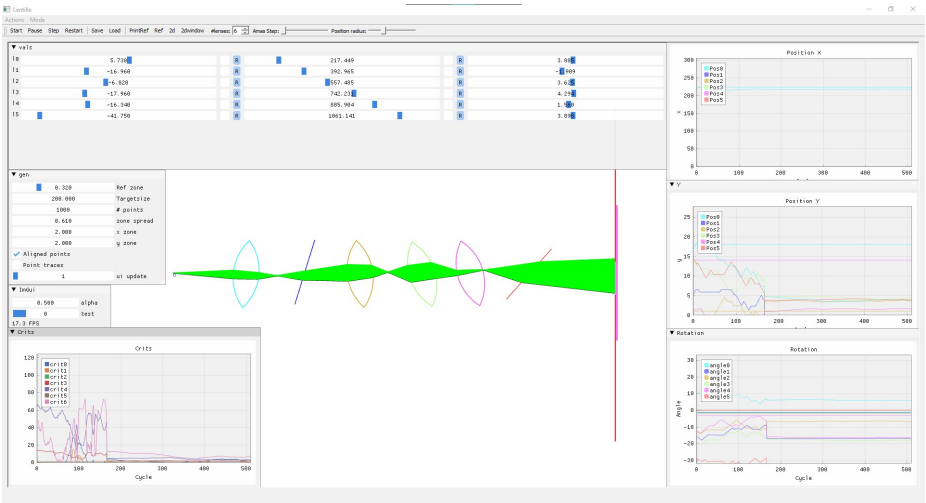

Fig. 4. Screenshot of simulator running

an objective to help according to its criticality and the impact that the agent believes it has on it. The momentum mechanism has another advantage with this decision process since the agent operates what we could call a stochastic pseudo-scan of its local search area. In cases where this second decision process does not lead to a decision, the agent acts randomly.

4 Simulation and Experiments

The MAS is connected to an optical simulator and has to update the state of this system, apply the changes decided by the agents and compute the feedback. The simulator we have developed is presented in Figure 4. It is a 2D-world composed of a light source, several lenses (L_i with i in $[1, N]$) and a screen. The light source emits a number of rays (R_j with j in $[1, M]$) of conical shape. If a ray crosses a lens, it is refracted according to the Snell-Descartes laws: $n1.sin(\theta_1) = n2.sin(\theta_2)$ (where $theta$ is the angle measured from the normal to the surface hit by the ray, and n_i is the refractive index of the respective medium). Thus, assuming that a ray passes through all the lenses in the system, we have a mathematical sequence of operations applied to its position and orientation: $R_j(pos_{j,i}, \theta_{j,i}) = L_i(R_j(pos_{j,i-1}, \theta_{j,i-1}))$ with i in $[1, N]$ and j in $[1, M]$.

The test cases are generated so that all rays pass through all lenses as follows: a set of lenses with various characteristics (thin or cylindrical shape, refractive index, focal length) are randomly placed on the axis between the light source and the screen. A set of rays is generated parallel to the axis X so that each ray passes through all the lenses and reaches the screen. The lenses are then iteratively and randomly moved and rotated, changing the direction of the rays. After a certain number of cycles, the state of the system is defined as the reference to be reached

by the agents. The following iterations are used to artificially deteriorate the state of the system to what will be the starting point of the experiment.

A lens L_i is represented by its type (thin or cylindrical), its position $P_i(x, y)$ and its rotation angle T_i, its refractive index n_i and its focal length F_i. For cylindrical lenses, the radius of each face is also defined: $R1_i$ and $R2_i$. Among these parameters, only P_i and T_i can change during the execution and are controlled by the agents. So we have 3 solver agents per lens[3]: X and Y for the position on 2 axes and Rt for the rotation in the plane, as represented in Figure 5.

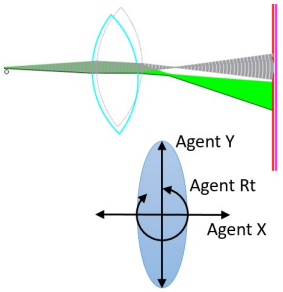

Fig. 5. Lens control diagram (theoretical objective beam in hashed gray)

A ray is a more complex structure since it is represented by a path. A path is an ordered list of positions and directions describing the points of intersection with the various lenses it passes through $\{p_{j,k}(x, y)\}$ and the direction of the ray at these points represented by an angle to the X axis $\{ang_{j,k}\}$, with k the index of the intersection.

The rays are not directly known by the agents. Only the last position and direction of each ray (when it reaches the screen) are used to compute sensor measures to send to the MAS. For example two measures can be computed from the mean square deviation of the positions and angles of the rays:

$$R_{pos} = \sqrt{(\sum((p_{j,last}(y) - p^{ref}_{j,last}(y))^2)/M)} \tag{2}$$

$$R_{ang} = \sqrt{(\sum((ang_{j,last} - ang^{ref}_{j,last})^2)/M)} \tag{3}$$

As seen in section 3.2, an objective agent is associated with each sensor measure and updates its criticality depending on the gap with its objective value.

Other measures such as the width of the beam, or the power (percentage of rays arriving on the screen) can be calculated from this set of intersections of the rays with the screen. Our goal is to construct values that are as representative as possible of what can be perceived on a real system with one or more specific sensors.

5 Results

For each experiment presented in this section, for observability issues necessary to validate the tested hypotheses, each parameter evolves at a fixed step, and a curve of the most critical objective is represented at each iteration.

Each of the first two graphs (Fig. 6) represent an experiment run one hundred times, allowing to observe the variability of the resolutions on a single lens

[3] Note that in reality, in 3 dimensions, there will be 6 agents for the 6 degrees of freedom.

(cylindrical and thin). We notice that when the system is far enough from the solution (in the first cycles where the criticalities oscillate under 100), the convergence is slower. We observed the agents decisions are mostly cognitive and stochastic at this point. The main reason is that agents try to know what is the right direction. The momentum mechanism is crucial here. Then when the acquired information becomes sufficient to perceive a direction, the criticality decreases more and more quickly. The agents then make a series of reactive decisions that maintain this so-called acceleration phase. When the criticalities reach a level where their competition is strong, thus stopping the convergence, the agents reach an equilibrium in cognitive and reactive decisions which will help to continue converging towards the solution. However, this final phase includes the most non-cooperative synchronizations.

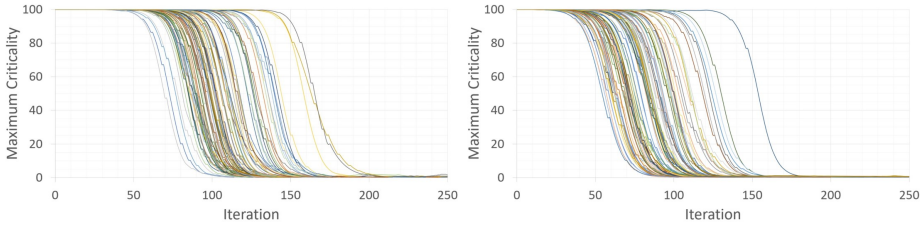

Fig. 6. Highest criticality over 100 repetitions of 2 problems with 1 lens (cylindrical on the left and thin on the right)

5.1 Problem Specific Knowledge

We made the hypothesis that the convergence of the system towards satisfactory states tends to accelerate when we provide agents with additional information that they take into account seamlessly without having to change their algorithms.

This hypothesis was tested by introducing a criticality on the deviation from the desired position of the quartile rays (in the order of the screen intersections). For example if Q_1 is more critical than Q_3 it means that the beam is too high compared to the target. The added information is that depending on which quartile is more critical, the collective will tend to redirect the beam faster than with the overall criticality of the positions.

The collective tries to help the most critical agent first, so it will lower the beam position before continuing the optimization of other objectives. This observation, validated empirically, led to better results when the quartile ray position criticalities are added (Fig. 7 and 8).

Indeed, we notice that the convergence of the system tends to be accelerated in the first phase of the resolution. The addition of criticality for purely informative purposes for the solver agents improves the worst resolution cases. However, we must be careful not to disturb their cooperation, because these additional criticalities are new criteria between which to arbitrate.

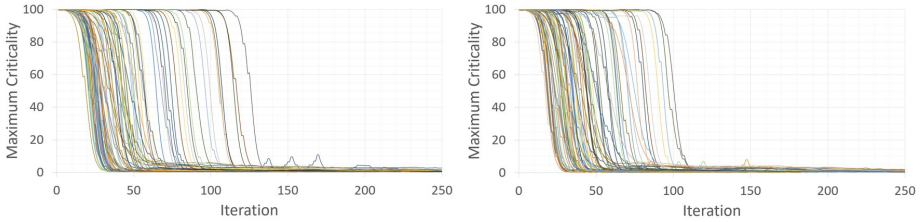

Fig. 7. Highest criticality over 10 repetitions of 10 1-lens problems without (left) and with criticality on quartiles (right)

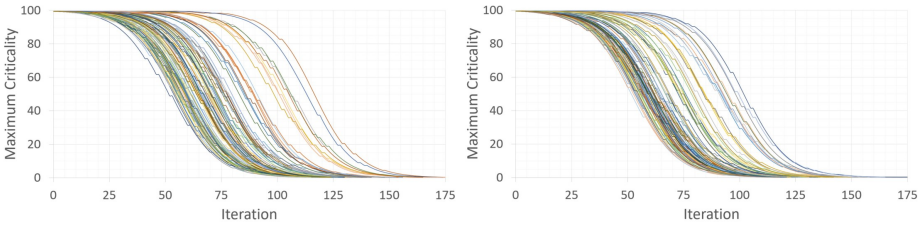

Fig. 8. Highest criticality over 100 repetitions of a 1-lens problem without (left) and with criticality on quartiles (right)

5.2 Increasing Complexity by the Number of Lenses

For an optical system, we consider that the increase of the complexity of a problem goes hand in hand with the increase of the number of lenses that compose it. Indeed, each lens has 3 axes of alignment, and therefore 3 design variables to the problem, which means 3 additional agents. Thus, for 10 lenses, we have a total of 30 agents who must cooperate in a 30 dimensional search space where each step deforms this space.

The following experiments have been performed with strongly reduced fixed steps because the instability of the system is greatly increased, due to the discontinuities of the beam in the chaining of the lenses. This causes peaks in criticality curves that can be remarked for 3-lens problems (Fig. 9) and even more for 7-lens

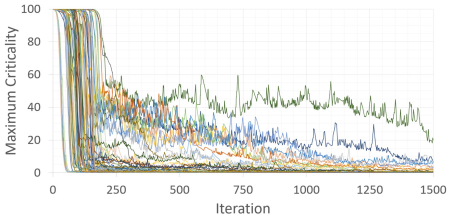

Fig. 9. Highest criticality over 10 repetitions of 10 3-lens problems

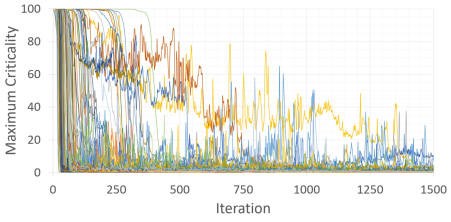

Fig. 10. Highest criticality over 10 repetitions of 10 7-lens problems

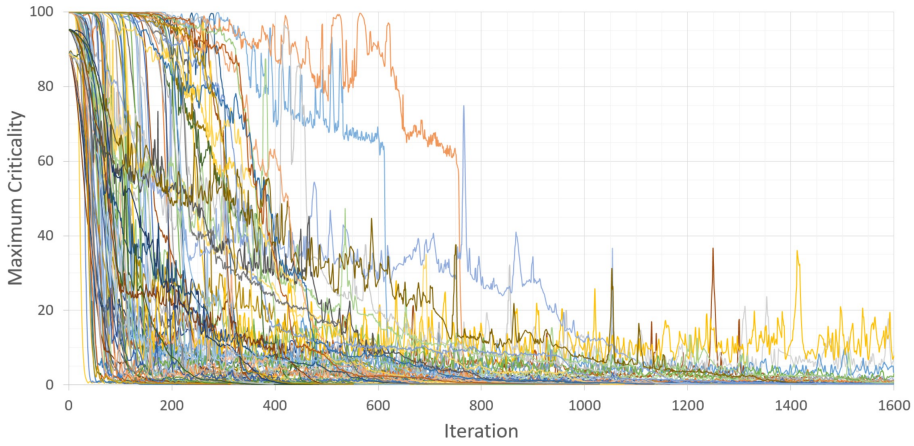

Fig. 11. Highest criticality for 10 repetitions of 10 problems with 10 lenses

problems (Fig. 10). However, a strong homeostasis capacity allows the agents to successively degrade and improve the solution to globally converge to more satisfying configurations. This is the result of the trade-off between exploration and exploitation in the agents' behavior. We can see that the system also manages to converge with 10 lenses (Fig. 11). The increase in complexity is not the heart of the problem. The difficulty lies in the cooperation of an increasing number of interacting elements in an environment with many interconnections. The optimization problem has become at this stage a problem of systemic cooperation. The more efficient it is, the better the MAS will be in its resolution.

6 Conclusion and Perspectives

This article proposed a multi-agent approach to naturally model a real-world complex optimization problem as a cooperative solving problem and to satisfy as much as possible objectives given by experts at a reasonable computational cost. This is made possible by the agentification of the constraints and objectives of the problem and by making these agents cooperate in order to adapt the parameters of the system.

Experiments performed and results obtained show that this approach is generic enough to be applied to a wide range of complex systems, notably in Photonics with various search space topologies. It also shows that the agents manage to differentiate themselves by observing the behavior of the complex system in response to their changes. Results are yet to be confirmed on real data, and a deployment of this solution is in progress on physical robotic industrial systems, both in Photonics and control loop tuning domains.

The behaviour of solver agents, notably their decision phase, has been kept as simple as possible for now. Some improvements are planned inside the agents

by implementing further learning algorithms able to work on sparse data, or by adding criticalities to have a constraint/stagnancy repulsive behavior.

Acknowledgements This work is financially supported by the Occitanie Region (`www.laregion.fr`) as part of the READYNOV 2019-2020 research program. Quentin Pouvreau is co-funded by the French National Association for Research and Technology (ANRT) (`www.anrt.asso.fr`) and by *ISP System*.

References

1. Fioretto, F., Pontelli, E., Yeoh, W.: Distributed constraint optimization problems and applications: A survey. Journal of Artificial Intelligence Research (2018)
2. Hoang, K.D., Yeoh, W.: Dynamic continuous distributed constraint optimization problems. In: PRIMA 2022: Principles and Practice of Multi-Agent Systems: 24th International Conference, Valencia, Spain, November 16–18, 2022, Proceedings. pp. 475–491. Springer (2022)
3. Hoang, K.D., Yeoh, W., Yokoo, M., Rabinovich, Z.: New algorithms for continuous distributed constraint optimization problems. In: Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems (2020)
4. Holland, J.H.: Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and AI. MIT press (1992)
5. Hussain, K., Mohd Salleh, M.N., Cheng, S., Shi, Y.: Metaheuristic research: a comprehensive survey. Artificial intelligence review **52**, 2191–2233 (2019)
6. Jorquera, T., Georgé, J.P., Gleizes, M.P., Régis, C.: A Natural Formalism and a MultiAgent Algorithm for Integrative Multidisciplinary Design Optimization. In: IEEE/WIC/ACM International Conference on Intelligent Agent Technology - IAT. Atlanta, USA (2013)
7. Jourdan, L., Basseur, M., Talbi, E.G.: Hybridizing exact methods and metaheuristics: A taxonomy. European Journal of Operational Research **199**(3) (2009)
8. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of ICNN'95-international conference on neural networks. IEEE (1995)
9. Perles, A., Crasnier, F., Georgé, J.P.: AMAK - A Framework for Developing Robust and Open Adaptive Multi-agent Systems. In: 16th International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS). Communications in Computer and Information Science book series (CCIS), Spain (2018)
10. Sharma, M., Kaur, P.: A comprehensive analysis of nature-inspired meta-heuristic techniques for feature selection problem. Archives of Computational Methods in Engineering (2021)
11. Sörensen, K.: Metaheuristics—the metaphor exposed. International Transactions in Operational Research **22**(1), 3–18 (2015)
12. Wang, Z., Qin, C., Wan, B., Song, W.W.: A comparative study of common nature-inspired algorithms for continuous function optimization. Entropy (2021)
13. Yang, T., Yi, X., Wu, J., Yuan, Y., Wu, D., Meng, Z., Hong, Y., Wang, H., et al.: A survey of distributed optimization. Annual Reviews in Control (2019)
14. Ye, D., Zhang, M., Vasilakos, A.V.: A survey of self-organization mechanisms in multiagent systems. IEEE Transactions on Systems, Man, and Cybernetics: Systems (2016)