# Adaptive Continuous Multi-Objective Optimization using Cooperative Agents

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

# Adaptive Continuous Multi-Objective Optimization using Cooperative Agents

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
{name.surname}@irit.fr

**Abstract.** Real-world optimization of complex products (*e.g.*, planes, jet engines) is a hard problem because of huge multi-objective and multi-constrained search-spaces, in which many variables are to be adjusted while each adjustment essentially impacts the whole system. Since the components of such systems are manufactured and output values are obtained with sensors, these systems are subject to imperfections and noise. Perfect *digital twins* are therefore impossible. Furthermore simulating with sufficient details is costly in resources, and the relevance of Population-based optimization approaches, where each individual is a whole solution to be evaluated, is severely put in question. We propose to tackle the problem with a Multi-Agent System (MAS) modeling and optimization approach that has two major strengths : 1) a natural representation where each agent is a variable of the problem and is perceiving and interacting through the real-world topology of the problem, 2) a cooperative solving process where the agents continuously adapt to feedback, that can be interacted with, can be observed, where the problem can be modified on-the-fly, that is able to directly control these variables on a real-world product while taking into account the specifics of the components. We illustrate and validate this approach in the *Photonics* domain, where a light beam has to follow a path through several optical components so as to be transformed, modulated, amplified, etc., at the end of which sensors give feedback on several metrics that are to be optimized. Robotic arms have to adjust the 6-axis positioning of the components and are controlled by the Adaptive MAS we developed.

**Keywords:** Continuous Optimization · Multi-Objective Optimization · Adaptive Multi-Agent Systems · Robotics Control · Photonics.

## 1 Problem Statement and Positioning

This study mainly concerns optimization problems from real-world applications, especially robotics control command based on sensor feedback. These applications go from system configuration based on test bench feedback to real-time

feedback control of an automated system. In the first case we want to optimize the response to an input. In the second case we mainly want to minimize the distance between sensor feedback and objectives by positioning, orienting, aligning one or more components. We define a robot, an actuator or any automated system as a composition of one or more axes, which are associated with the degrees of freedom of the system. Such problems have a wide variety of external constraints like limited resolution time or limited number of moves, predetermined components positions to respect, etc. We intend to develop an optimization system able to adapt to multiple robotics applications. The application domain we focus on is the photonics domain as illustrated in Fig. 1.

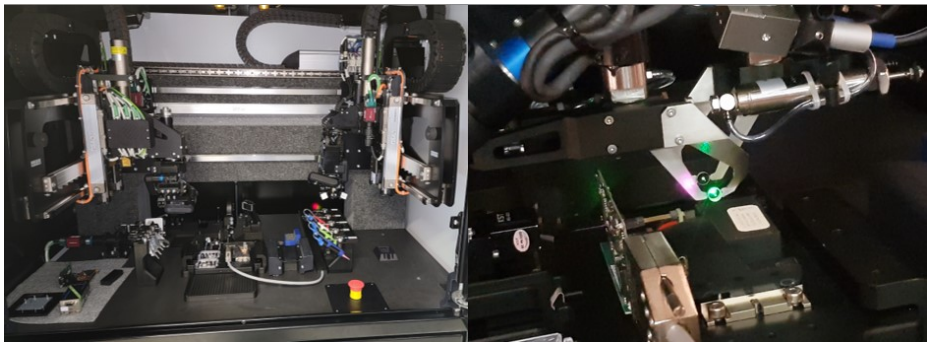

**Fig. 1.** Example of a real photonics application we are working on (Credits ISP System)

### 1.1 Search Space Topology

Optimization problems are divided into domains depending on their search space topology. Search space dimensions are defined by decision variables, and potentially limited by hard constraints. For example the number of robots and the number of axes per robot increase the dimensions. On the other hand, limits on a component assembly reduce the possibilities on one or more axes, so one or more dimensions get constrained in the corresponding search space. These constraints, alongside one or more expert-given objectives, most of the time antinomic, make appear a Pareto front in the search space. As constraints can generally be transposed into objectives dealing with the distance to a threshold (the further away from the threshold, the better, or defining a cost regarding an acceptable violation), these problems can be defined as Multi-Objective Optimization (MOO) problems, also called Pareto optimization problems.

Another factor of dimensioning of the search space is the decision variable domain. Depending on whether the domain is discrete or continuous, optimization problems are combinatorial or continuous. Since the precision of robotic systems is continuously increasing, robots positioning problems can be considered as continuous optimization problems.

Indeed, the axis resolution, meaning the minimal step with accuracy guarantee, is often very small compared to the range of values it can achieve. Furthermore, axes resolutions, ranges and other robots features can change from an application to another, changing therefore the decision variable domains.

A continuous problem has a potentially infinite number of solutions when a combinatorial problem has only all possible combinations of its discrete variables. This gap can have a significant influence in terms of calculation cost. However, those properties do not mean that combinatorial problems are trivial and continuous problems are not. It means that search space topology is quite different from a category to another so the employed method might not have the same results.

In this study, we focus on Continuous Multi-Objective Optimization problems and we consider that reliable problems with only one objective are a particular subdivision of these problems and can be processed in the same way.

### 1.2 Multi-Objective Optimization Approaches

This section introduces the main approaches used in the optimization domain before focusing on the most suited for our concern.

Analytical approaches are the most accurate but they are time-consuming and may not also be suitable for large-scale optimization problems. Arithmetic programming approaches on the contrary have fast computation performances since they are based on simplifications and sequential linearizations. However, they are very weak in handling multi-objective nonlinear problems and may converge to local optima, non-necessarily satisfying enough [16].

Meta-heuristic optimization algorithms are extensively used in solving Multi-Objective Optimization problems since 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 which provide, if not optimal, at least satisfying solutions by stochastically exploring and exploiting search spaces iteratively [17]. The following sections focus on these algorithms.

**Population-Based Heuristics** are a large part of the state of the art in Meta-heuristic Optimization Algorithms. The main principle is to simultaneously handle a population of solutions spread randomly or not in the search space. The population can evolve and select the best solutions iteratively, or converge to an optimum following a set of influence rules. These algorithms can also be used in hybrid solutions alongside more classical algorithms like simulated annealing [1] to balance their weaknesses.

It is difficult to be exhaustive about all works in this domain. For instance, Genetic Algorithms [9] and Particle Swarm Optimization Algorithms [12] have together more than 3000 publications per year [18].

A major limitation of Population-Based approaches is their potential computational cost. Such algorithms need to evaluate a relatively large number of

candidates in order to create a good population of solutions. Computing all the candidate solutions can be prohibitive for computationally expensive problems.

The type of applications we are studying here requires an adaptation each time we get a sensor feedback. As a result, an evolution process would require a huge amount of resources. As each new candidate in the population needs to be evaluated, the robot needs to reconfigure the whole experimental setting for each proposed configuration. An adaptive algorithm modifying and proposing for testing a unique configuration at each feedback is a more suitable strategy (i.e. a resolution process forming a single trajectory in the search space).

Moreover, when scaling up the number of objectives, the Pareto-dominance relation essentially loses the ability to distinguish desirable solutions, since nearly all population members are non-dominated at an early stage of the search. In fact, a large majority of the usual Population-Based methods (evolutionary algorithms, swarm intelligence) have been shown to degrade when the number of objectives grows beyond three, and moreover beyond eight [10]. This particularity explains why a part of the literature about these approaches focuses on Many-Objective Optimization, that is Multi-Objective Optimization problems with more than three objectives [13]. The need for this category of problems to have more relevant indicators than Pareto-dominance makes the Population-Based Heuristics to specify additional calculation for solution comparison.

The types of problems we are interested in require an optimization system able to converge without maintaining and computing a large number of solutions, especially when solution comparison becomes non-trivial. Furthermore, we need a decentralized real-time control of robots arms to actually optimize the real world system being processed, taking into account errors and noise.

**Multi-Agent Systems** (MAS) are a decentralized approach, based on self-organisation mechanisms [22], where the calculation task is distributed over *agents* which are virtual or physical autonomous entities[19].

Each agent has only a local point of view of the problem it is solving, corresponding to a local objective function. The global objective function of a problem is then the sum of all these local functions. This particularity enables to easily distribute calculation tasks in the resolution process and consequently reduces computational costs. That is why multi-agent approaches are preferred where centralized approaches have limited flexibility and scalability.

Multi-Agent Systems are used in a wide variety of theoretical [15] and real-world application domains of distributed optimization [21]: classification optimization algorithms [3], power systems [7] , complex networks and IoT [5] , smart manufacturing [2] or multi-robot systems [20].

**Distributed Constraint Optimization Problems** (DCOPs) are a well-known class of combinatorial optimization problems prevalent in Multi-Agent Systems [4]. DCOP is a model originally developed under the assumption that each agent controls exactly one variable.

This model was designed for a specific type of problems where the difficulty resides in the combination of multiple constraints. These problems are supposed to be easily decomposable into several cost functions, where the cost values associated with the variables states are supposed to be known. This major assumption does not stand for complex continuous optimization problems, where the complexity of the models and their interdependencies cause this information to be unavailable in most cases. It is important to remark that some works tried to extend DCOP model to continuous optimization problems but the state of art about those works remains scattered for now [8].

## 2 AMAS Theory for Optimization

As seen before, a MAS is a problem-independent solution making it possible to have a natural representation where each agent is a variable of an MOO problem. These agents, which perceive their environment and interact, are also a means to continuously adapt to real-world feedback and provide a "solution" anytime especially when the problem can be modified on-the-fly.

We propose to adopt the Adaptive Multi-Agent Systems (AMAS) theory where cooperation [6] is the engine that drives the adaptation of an agent and the emergence of a global functionality. This cooperation relies on three mechanisms : an agent may adjust its internal state to modify its behavior (tuning), may modify the way it interacts with its neighborhood (reorganization) or may create other agents or self-suppress when there is no other agent to produce a functionality or when a functionality is useless (evolution).

### 2.1 Natural Domain Modeling

As we previously stated, when solving complex continuous problems existing techniques usually require a transformation of the initial formulation, in order to satisfy some requirements for the technique to be applied. Beside the fact that correctly applying these changes can be a demanding task for the designers, imposing such modifications changes the problem beyond its original, 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 which is the most natural and easiest for the expert to manipulate. We call this modeling *Natural Domain Modeling for Optimization* (NDMO) [11].

In order to represents the elements of a generic continuous optimization model, we identified five classes of interacting entities: *models*, *design variables*, *outputs*, *constraints* and *objectives*. Briefly: given the values of the design variables, certain models will calculated output values, which will enable other models to calculate other outputs and so on, until constraints and objectives can be calculated, in a sort of calculus network. In general, three elements need to be *agentified*: the design variables (because they need to be optimised and that constitutes the solving process), the constraints and objectives. The last two

are there to model the requirements or statements of the problem, i.e. what the solving process has to achieve.

To this end, we use a mechanism based on a specific measure called *criticality*. This measure represents the state of dissatisfaction of the agent regarding its local goal. Each agent is in charge of estimating its own criticality and providing it to the other agents. The role of this measure is to aggregate into a single comparable value all the relevant indicators regarding the state of the agent. Having a single indicator of the state of the agent is interesting as it simplifies the reasoning of the agents. However the system designer has the difficult task to provide the agents which adequate means to calculate their criticality.

### 2.2 Agent Internal State and behavior

Our system is a work-in-progress implementation based on the AMAK Framework [14]. A main system representing the AMAS handles an environment representation and a collection of agents interacting with the environment. Each agent controls a parameter of the system, consequently a decision variable of the problem. The environment and the set of agents execute an iteration to update their state one at a time. The agents iteration order is randomly updated at the beginning of each cycle. All the agents have the same three-phase algorithm:

- Perception phase: the agent gets an observation of the environment, that is a set of *criticalities* calculated from the distance to the objective.
- Decision phase: the agent processes its new data and follows a decision tree to adjust its internal state.
- Action phase: the agent acts following its decision by changing its parameter.

The agent decision phase aims at increasing or decreasing the value of the decision variable it is responsible for (a predefined variation step depending on its characteristics), and is thus at the core of the process. Except in one case that we will explain below, a decision is always repeated a stochastically chosen number of iterations, from one to ten. This *momentum* mechanism allows to desynchronize the agents decisions to prevent them from being trapped in what we call *non cooperative synchronisations* (basically when two agents try to "help" at the same time, thus hindering each other).

When starting the resolution, each agent has only a set of criticalities given by the last environment update. Since it does not have any idea of which action is the best, its first decision is to act randomly. If the agent observes the criticalities decreased beyond a configurable threshold, it will repeat its decision. It is generally useful to converge fast when the current region of the search space is relatively regular. On the contrary, if the criticalities increased beyond the same threshold, the agent will make only one step in the opposite way. These two rules make a first decision process we can qualify as reactive. When it is not possible to reactively spot an adequate decision, the agent will process data it registered in the last perception phases. This more cognitive process consists in interpolating the variation of the criticalities according to its own value. The

goal is to identify a region (plus or minus) where the integrative is the lower. It appears that the momentum mechanism is also useful to this decision process since it made the agent do what we might call a stochastic scanning of its local area. In the rare case the second decision process fails to give a decision, the agent acts randomly.

## 3 Photonics Problem Modeling and Implementation

The environment of the AMAS has to update the system state, apply the changes from the AMAS and calculate the input variables used by the agents to take their decisions. The simulator we developed is shown in Fig. 2.

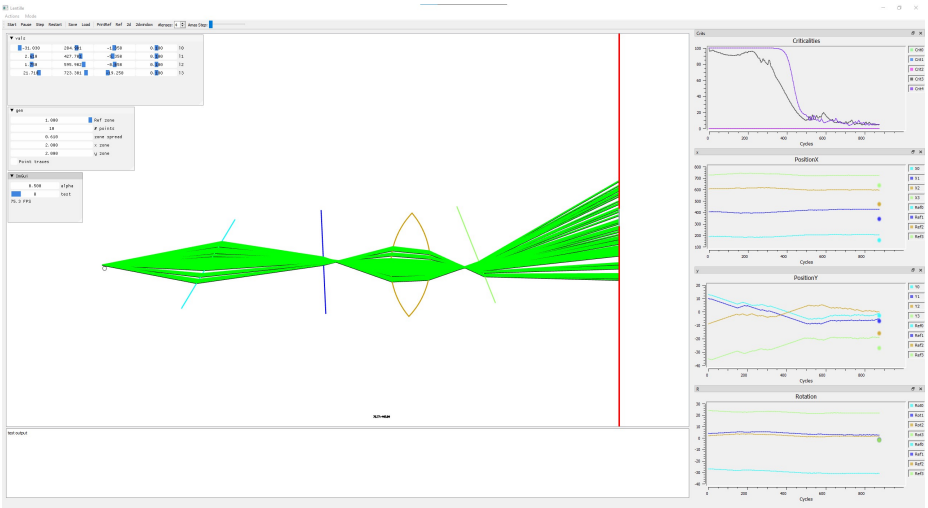


**Fig. 2.** Screenshot of the simulator at runtime

The system 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 $[1, M]$) in a conic shape. If a ray intersects with a lens it is refracted using Snell's law of refraction *i.e.* $n1.sin(\theta_1) = n2.sin(\theta_2)$ (with each $\theta$ as the angle measured from the normal of the boundary, and $n$ as the refractive index of the respective medium). So assuming a ray goes through all lenses in the system (which is the desired state) we have a mathematical suite 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 $[1, m]$.

Test cases for the AMAS are generated so that all rays pass through all lenses as follows: a set of lenses with various characteristics (thin or cylindrical, refraction index, focal length) are randomly placed on the axis between the source and the screen ($X$). A set of rays is generated parallel to the $X$ axis

so that each ray goes through all lenses and reaches the screen. Repeatedly, the lenses are shifted and rotated randomly as well as the direction of the rays, while keeping all rays going through all lenses. After a number of cycles, the state of the system is set as the reference to reach for the AMAS. Then the lenses are randomly placed and rotated. This new state is used as a starting point for the AMAS to work with.

A lens $L_i$ is represented by its type (thin or cylindrical), its position $P_i(x, y)$ and rotation angle $T_i$, its refraction index $n_i$ and focal length $F_i$. For cylindrical lenses the radius of each face are also necessary $R1_i$ and $R2_i$. From these parameters only $P_i$ and $T_i$ can change during the run and are controlled by the AMAS.

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 intersection points with the various lenses it goes through $\{p_{j,k}(x, y)\}$ with $k$ the index of the intersection, and the direction of the ray at these points represented as an angle with $X$ $\{ang_{j,k}\}$.

Rays are not directly known by the AMAS. Only the last position and direction of each ray (when it reaches the screen) is used to derive information to send to the AMAS as a feedback to its actions.

The derived information can be the results of various computations. The most straightforward is to form a set of $M$ differences between current rays and reference rays: $\{|p_{j,last}(y) - p^{ref}_{j,last}(y)|, |ang_{j,last} - ang^{ref}_{j,last}|\}$. This gives the AMAS quite a lot of precise information which is not often readily available in real life systems.

The second type of derived information are root mean square deviations ($rmsd$) of the positions and angles: $R_{pos} = \sqrt{(\sum((p_{j,last}(y) - p^{ref}_{j,last}(y))^2)/M)}$ and $R_{ang} = \sqrt{(\sum((ang_{j,last} - ang^{ref}_{j,last})^2)/M)}$

These two values are more representative of what can be perceived on a real system like the global intensity on the screen.

For each of the experiments presented hereafter a set of parameters are given which represents the setup of the run: first, the sequence of lenses, from source to screen, present in the system with $C$ for a cylindrical lens and $T$ for a thin lens. Then the number of rays, the percentage of maximum step for lenses moves and the type of information sent to the AMAS (full or $rmsd$). So for an experiment with 3 lenses, 10 rays, 50% of maximum step and using the $rmsd$, this set would be $\{CTC, 10, 50\%, rmsd\}$.

## 4 Experiments

First we checked the ability to manage different types of lenses, as for instance, a cylindrical lens has more complex interactions with rays than a thin lens.

We generated two experiments $\{T, 100, 10\%, rmsd\}$ and $\{C, 100, 10\%, rmsd\}$ (Fig. 3 and Fig. 4) in which all the other parameters were equal. As visible on the graphs, the evolution of criticality is more chaotic with a cylindrical lens.

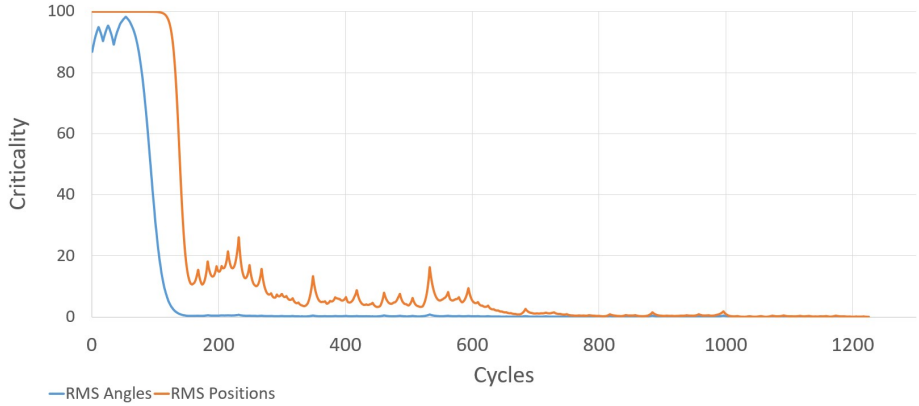


**Fig. 3.** Resolution with one thin lens $\{T, 100, 10\%, rmsd\}$

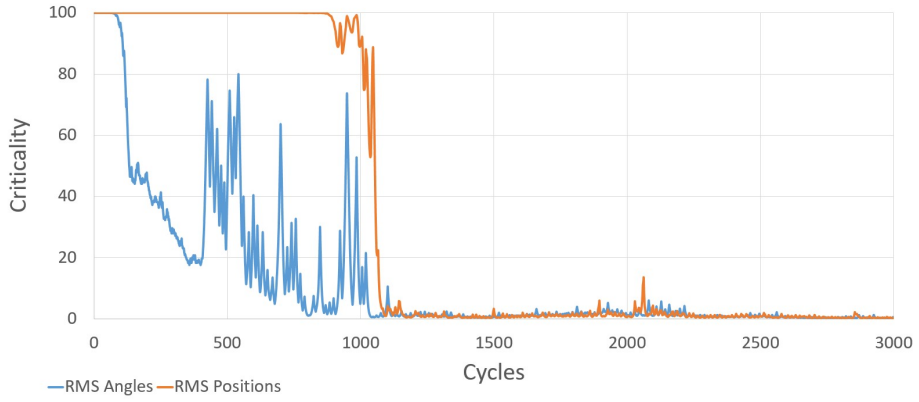


**Fig. 4.** Resolution with one cylindrical lens $\{C, 100, 10\%, rmsd\}$

We remark in the one lens experiments that curves make some peaks repetitively. These are the consequence of the momentum mechanism seen in section 2.2 and do not impact the convergence that remains globally continuous.

The other experiments (Figs. 5 and 6) show that the system seems to be scalable in terms of number of decision variables. In these cases, the peaks mentioned earlier disappeared. The results of the moves of each axis agent are more softened as the number of interactions between rays and optical surfaces grows.

This proof of concept shows promising results: the resolution process succeeds and no divergence from a satisfying area of the search space has been observed. However, some adjustments that have to be explored yet could greatly improve the resolution process. The difficulties encountered are mainly due to the problem itself: almost all positioning values impact all criticalities, and in a non-linear way. The parameters consequently are strongly interconnected: the current posi-

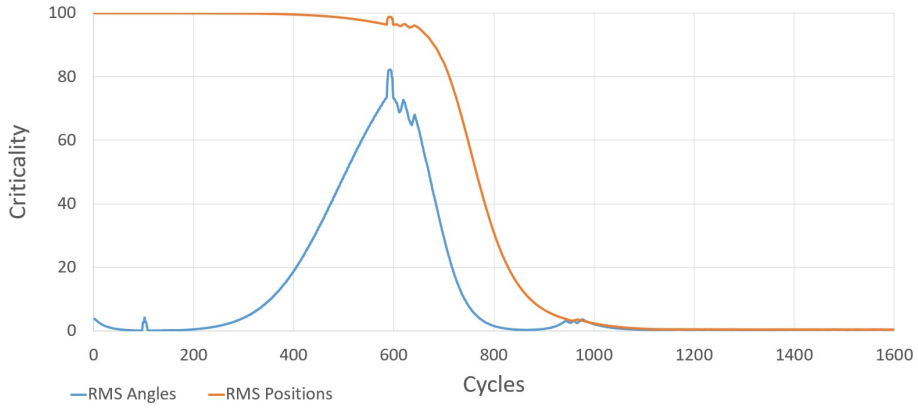


**Fig. 5.** Resolution with two lenses $\{CT, 100, 1\%, rmsd\}$

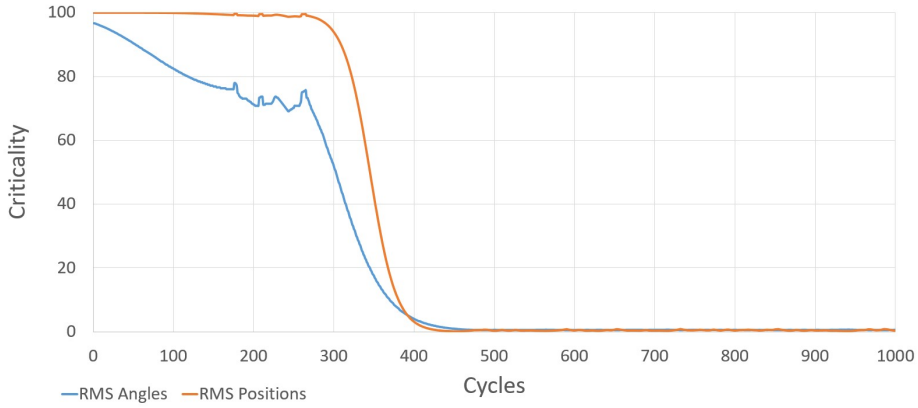


**Fig. 6.** Resolution with three lenses $\{TTT, 100, 1\%, rmsd\}$

tion of one axis agent moves the target position of one or more others. Therefore, the system is chaotic and the agents can collectively hinder themselves. At this point the optimization problem becomes a cooperation problem.

It has to be noted that the examples presented here are voluntary more difficult for the AMAS than a real case, where the starting positions are nearer from the optimum. Starting further away lets us test how the system behaves while crossing the vast search space of the problem. In this way we can observe that it does not suffer divergence from any achievement it has already made. Further work will be done to optimise the number of cycles needed to bring the criticalities down near zero.

## 5 Conclusion and Perspectives

State of the art in multi-objective optimization is dominated by Population-Based and DCOP approaches. However, it can be difficult for those methods to be used for real-world applications, especially when the context brings new constraints like narrowed computation time or resources. This is even worse in real world robotics control where only one "current solution" can be manipulated by the robots, and no *digital twin* is possible.

Moreover, we identified a limitation of current continuous optimization methods regarding the handling of complex problems with a multi-dimensional continuous search space. Problems of this category are usually too complex to be solved by classical optimization methods due to multiple factors: the inter-dependencies of their objectives, their heavy computational cost, their non-linearities, etc.

This limitation has been the motivation to propose a new decentralized approach. We designed a solver fitted for a large panel of real-world applications with miscellaneous search space topologies. This approach also permits to easily scale up problem complexity, in terms of number of parameters as well as number of objectives. Its aim is to naturally model a real-world optimization problem as a cooperative resolution problem and to satisfy as much as possible expert given objectives at a reasonable computation cost. The first results obtained with a proof of concept are promising. Enhancing the optimization process will now rely on enriching the cooperation capabilities of the agents.

**Acknowledgements** This work has been financially supported by the *Région Occitanie* (`www.laregion.fr`) as part of the READYNOV 2019-2020 research program. Quentin Pouvreau has been co-funded by the *Association nationale de la recherche et de la technologie (ANRT)* (`www.anrt.asso.fr`) and by *ISP System*(`www.isp-system.fr`), who also provided the test cases and expert knowledge on photonics.

## References

1. Assad, A., Deep, K.: A hybrid harmony search and simulated annealing algorithm for continuous optimization. Information Sciences **450**, 246–266 (2018)
2. Bendul, J.C., Blunck, H.: The design space of production planning and control for industry 4.0. Computers in Industry **105**, 260–272 (2019)
3. Couellan, N., Jan, S., Jorquera, T., Georgé, J.P.: Self-adaptive support vector machine: A multi-agent optimization perspective. Expert systems with Applications **42**(9), 4284–4298 (2015)
4. Fioretto, F., Pontelli, E., Yeoh, W.: Distributed constraint optimization problems and applications: A survey. Journal of Artificial Intelligence Research **61**, 623–698 (2018)
5. Fortino, G., Russo, W., Savaglio, C., Shen, W., Zhou, M.: Agent-oriented cooperative smart objects: From iot system design to implementation. IEEE Transactions on Systems, Man, and Cybernetics: Systems **48**(11), 1939–1956 (2017)

6. Georgé, J.P., Gleizes, M.P., Camps, V.: Cooperation. In: Serugendo, G.D.M., Gleizes, M.P., Karageorgos, A. (eds.) Self-organising Software, pp. 193–226. Natural Computing Series book series (NCS), Springer (2011)
7. González-Briones, A., De La Prieta, F., Mohamad, M.S., Omatu, S., Corchado, J.M.: Multi-agent systems applications in energy optimization problems: A state-of-the-art review. Energies **11**(8), 1928 (2018)
8. 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)
9. Holland, J.H.: Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT press (1992)
10. Ishibuchi, H., Tsukamoto, N., Nojima, Y.: Evolutionary many-objective optimization: A short review. In: 2008 IEEE congress on evolutionary computation (IEEE world congress on computational intelligence). pp. 2419–2426. IEEE (2008)
11. 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. pp. 146–154. Atlanta, United States (2013)
12. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of ICNN'95-international conference on neural networks. vol. 4. IEEE (1995)
13. Maltese, J., Ombuki-Berman, B.M., Engelbrecht, A.P.: A scalability study of many-objective optimization algorithms. IEEE Transactions on Evolutionary Computation **22**(1), 79–96 (2016)
14. Perles, A., Crasnier, F., Georgé, J.P.: Amak-a framework for developing robust and open adaptive multi-agent systems. In: International Conference on Practical Applications of Agents and Multi-Agent Systems. pp. 468–479. Springer (2018)
15. Sghir, I., Hao, J.K., Jaafar, I.B., Ghédira, K.: A multi-agent based optimization method applied to the quadratic assignment problem. Expert Systems with Applications **42**(23), 9252–9262 (2015)
16. Shaheen, A.M., Spea, S.R., Farrag, S.M., Abido, M.A.: A review of meta-heuristic algorithms for reactive power planning problem. Ain Shams Engineering Journal **9**(2), 215–231 (2018)
17. Sharma, M., Kaur, P.: A comprehensive analysis of nature-inspired meta-heuristic techniques for feature selection problem. Archives of Computational Methods in Engineering **28**(3) (2021)
18. Wang, Z., Qin, C., Wan, B., Song, W.W.: A comparative study of common nature-inspired algorithms for continuous function optimization. Entropy **23**(7) (2021)
19. Weiß, G.: Multiagent Systems, A modern Approach to Distributed Artificial Systems. MIT Press (1999)
20. Yan, Z., Jouandeau, N., Cherif, A.A.: A survey and analysis of multi-robot coordination. International Journal of Advanced Robotic Systems **10**(12), 399 (2013)
21. Yang, T., Yi, X., Wu, J., Yuan, Y., Wu, D., Meng, Z., Hong, Y., Wang, H., Lin, Z., Johansson, K.H.: A survey of distributed optimization. Annual Reviews in Control **47**, 278–305 (2019)
22. Ye, D., Zhang, M., Vasilakos, A.V.: A survey of self-organization mechanisms in multiagent systems. IEEE Transactions on Systems, Man, and Cybernetics: Systems **47**(3), 441–461 (2016)