Multimethod Optimization for Reverse Engineering of Complex Biological Networks

Patricia González
Computer Architecture Group, University of A Coruña
Spain
patricia.gonzalez@udc.es

David R. Penas
MODESTYA research group, Institute of Mathematics (IMAT), University of Santiago de Compostela
Spain
david.rodriguez.penas@usc.es

Xoan C. Pardo
Computer Architecture Group, University of A Coruña
Spain
xoan.pardo@udc.es

Julio R. Banga
BioProcess Engineering Group, IIM-CSIC
Spain
julio@iim.csic.es

Ramón Doallo
Computer Architecture Group, University of A Coruña
Spain
ramon.doallo@udc.es

ABSTRACT

Optimization problems appears in different areas of science and engineering. This paper considers the general problem of reverse engineering in computational biology by means of mixed-integer nonlinear dynamic optimization (MIDO). Although this kind of problems are typically hard, solutions can be achieved for rather complex networks by applying global optimization metaheuristics. The main objective of this work is to handle them by means of multimethod optimization, in which different metaheuristics cooperate to outperform the results obtained by any of them isolated. For its preliminary evaluation we consider a synthetic signaling pathway case study and we assess the performance of the proposal on a public cloud. These results open up new possibilities for other MIDO-based large-scale applications in computational systems biology.

CCS CONCEPTS

• **Theory of computation → Parallel algorithms**; • **Applied computing → Bioinformatics**;

KEYWORDS

Reverse Engineering, Computational Systems Biology, MINLP Problems, Parallel Metaheuristics, Multimethod Optimization

ACM Reference Format:
Patricia González, David R. Penas, Xoan C. Pardo, Julio R. Banga, and Ramón Doallo. 2018. Multimethod Optimization for Reverse Engineering of Complex Biological Networks. In *PBio 2018: 6th International Workshop on Parallelism in Bioinformatics (PBio 2018), September 23, 2018, Barcelona, Spain.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3235830.3235832

PBio 2018, September 23, 2018, Barcelona, Spain

ACM ISBN 978-1-4503-6531-4/18/09…$15.00
https://doi.org/10.1145/3235830.3235832

1 INTRODUCTION

Optimization problems arises in many areas of life science, such as bioinformatics or computational systems biology [3]. The optimization process consists in locating the best solution or *optimum* inside of a topology or *search space* described by one or more mathematical functions. Optimization problems can be categorized in different models, such as combinatorial optimization models, constraint satisfaction models, non-analytic models, or mathematical programming models.

Models based on mathematical programming are the most popular ones, and can be further classified in linear and nonlinear models, depending if the objective function and the constraints were linear or not with respect to the decision variables. Mathematical programming problems can be classified according to the properties described by their functions, such as the existence of nonlinearities, the domain of the variables, or the presence of differential equations as constraints and time dependent decision variables. In models based on *nonlinear programming* (NLP), the complexity increases due to the nonlinearity of the objective function and constraints. When an optimization problem presents nonlinearity and the domain of the variables can be discrete or continuous, we talk about *mixed integer nonlinear programming* (MINLP) problems [6]. Besides, MINLP can be convex when all the functions are convex, or non-convex otherwise. Non-convex MINLPs are extremely hard to solve compared to NLP and convex MINLP problems.

Global optimization (GO) methods are robust alternatives to solve complex optimization problems. These methods can be roughly classified into *deterministic GO methods*, that explore the entire search space and find the global optimum, and *stochastic GO methods*, that do not guarantee convergence to the global optimum but provide near-global solutions in reasonable computation times. Recently, also *hybrid GO methods* have been proposed [3]. These methods arise from a combination of two or more methodologies: a couple formed by a global method and a local search, the union of global optimization solver with a deterministic method, or a set of GO methods combined among them.

Among the different GO methods, metaheuristics have become the *de-facto* choice for complex problems. A metaheuristic [14] is

an iterative generation process that guides a subordinate heuristic by combining different concepts for exploration (global search) and exploitation (local search) of the search spaces. It uses learning strategies to structure information in order to find efficiently near-optimal solutions. Metaheuristics allow handling NP-hard optimization problems by providing good enough solutions in a reasonable computation time, because they do not need to explore the entire address space. Moreover, modern metaheuristics often use hybrid approaches where the global search includes also local searches to obtain a compromise between the diversification provided by the global optimization method, and the intensification obtained by the inclusion of a local method. However, there is no guarantee to find global optimal solutions or even bounded solutions.

Although the use of metaheuristics allows a significant reduction of the computational complexity of the search process, it still remains time consuming for many problems in multiple domains of application. High performance computing (HPC) represents an effective strategy to speed up the time-to-solution. Metaheuristics as algorithms may have limited parallelism, however, as problem solving methods they offer opportunities for large-scale parallel computing. Since it is not easy to know in advance which of the numerous existing metaheuristics will be the most suitable for solving a given problem, a multimethod solution, where multiple different global search algorithms are executed concurrently and cooperate through information exchange, could perform a more effective exploration of the search space. In [10] we have further explored this idea in NLP problems, demonstrating that, if we can devote a significant amount of resources to the search, an adaptive multimethod would achieve a more effective exploration of the search space.

The aim of this paper is to explore the multimethod approach further considering extensions of the previous multimethod algorithm [10] so that it can handle general MINLP problems. The key differences and novel aspects with respect to the previous work are: (1) the addition of an efficient local solver for MINLP problems, (2) changes to the self-adaptation mechanism to avoid premature stagnation of the convergence in this kind of problems, and (3) the addition of new mechanisms to ensure diversity while keeping parallel cooperation.

As a case study to evaluate the performance of the proposal in challenging hard problems, we select the reverse engineering of cell signaling, which is of particular importance in systems biology [1]. Signaling pathways play, for instance, a key role in the development of novel therapies in complex diseases such as cancer. To describe all the possible regulatory structures for a given dynamic model of a pathway, a logic-based formalism with mixed-integer dynamic optimization (MIDO) can be used [12]. This framework aims to simultaneously identify the regulatory structure (represented by binary parameters) and the real-valued parameters that are consistent with the available experimental data, resulting in a logic-based differential equation model. Although this kind of problems are usually hard to solve, solutions can be achieved considering the use of mixed-integer nonlinear programming [12].

The organization of the paper is as follows. Section 2 presents the extensions carried out in the previous multimethod approach to handle MINLP problems. Section 3 describes, as a case study, the reverse engineering of cell signalling phenomena and the approach followed in this work for its solution. The performance evaluation is discussed in Section 4. Finally, conclusions and future work are detailed in Section 5.

2 SELF-ADAPTIVE COOPERATIVE MULTIMETHOD

Metaheuristics are popular methods to solve hard optimization problems. However, select and apply in an efficient and effective way a metaheuristic for a given problem is a difficult task, given that: a metaheuristic algorithm started from different initial solutions will explore different regions of the solution space and return different solutions; different configuration settings using the same metaheuristic algorithm will give different results; there will be large variations in performance over different instances of the same problem; and it is difficult to know in advance which metaheuristic will be the most suitable for solving the problem at hand. Thus, a multimethod global approach, in which multiple different global search algorithms are executed concurrently and cooperate among them, turns on a very appealing solution. Since we often do not know in advance the most suitable algorithm for a given optimization problem, an adaptive multimethod would be able to try a number of methods and select the appropriate one. Moreover, it would be able to use different algorithms in different search spaces and dynamically adapt the search according to the results obtained during the execution.

In a previous work [10] we have explored this direction proposing a self-adaptive cooperative multimethod, called saCMM. The main features of the saCMM implementation are:

- a loosely-coupled coarse-grained parallelization of the search diversification, following a master-slave approach, where the master is in charge of the control of the cooperation between the slaves (islands), that perform a different metaheuristic each
- a fine-grained parallelization of the cost function evaluations within each island
- an asynchronous communication protocol to avoid having idle islands waiting for information exchanged from others
- a self-adaptive procedure that changes dynamically the settings of those islands that do not progress with those of the most promising searches

The saCMM method has been extensively evaluated both in local clusters and in the Microsoft Azure cloud, demonstrating its potential for solving very challenging NLP problems. Therefore, we are interested in extending this method so it can be applied to large MINLP problems. As a result, we present here modifications and extensions of the original saCMM method.

First, an efficient local solver for MINLP problems has been added. A sequential quadratic programming solver, called *Mixed-Integer Sequential Quadratic Programming* (MISQP) [8, 9] has been included in saCMM. It assumes that the model functions are smooth: an increment of a binary or an integer variable can produce a small change of function values, though it does not require the mixed-integer function to be convex or relaxable, i.e. the cost function is evaluated only with discrete values in the integer or boolean parameters.

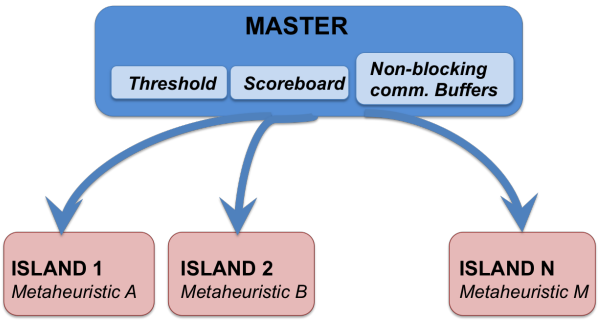

Figure 1: Schematic representation of saCMM method.

Second, changes to avoid a problem of premature convergence due to a quick lose of diversity in the islands have been performed. In mixed-integer problems a promising incoming solution in an island acted as an attractor for the members of the population, bringing them fast to the vicinity of this new value. Thus, two new strategies where introduced in the saCMM method to allow for a dynamic breakout from local optima, and to further preserve the diversity in the search for these problems, avoiding prematurely stagnation:

- On the one hand, cooperation between islands decreases too fast as the algorithm converges, since many of them stagnate. Thus, the criteria used in the original saCMM method to trigger the tuning of those islands that are not progressing in the search should be accommodated for MINLP problems, relaxing the adaptive conditions to allow for an earlier escape from the stagnated regions.
- On the other hand, in mixed-integer problems, when an island stagnates, most of the times is due to the lost of diversity in the population. Thus, diversity is promoted by a modified strategy: once an island requests a reconfiguration, most of the members of the population, except for two solutions (the best known solution and an arbitrary one), are randomly initialized again.

Figure 1 graphically illustrates the saCMM method. It follows a master-slave approach. Each slave executes a different metaheuristic. At present, two metaheuristics are implemented, the Differential Evolution (DE) [20], implemented with the enhancements described in [16], and the enhanced Scatter Search (eSS) [7], using the implementation outlined in [17]. However, each slave perform one of these metaheuristics also with different configuration parameters, which leads to different searches. The master process controls the long-term behavior of the parallel searches and their cooperation, and dynamically changes, in execution time, configuration parameters to improve the success of the parallel cooperative scheme.

Figure 2 summarizes the new saCMM method for MINLP problems. A local variable *BestKnownSol* is set to monitor the best solution shared in the cooperation among slaves. The master process also sets the initial communication threshold ϵ and initiates a scoreboard to monitor the progress of each slave. Then, a loop is carried out until a stopping criteria is reached, where the master waits for the messages coming from the slaves. In the cooperation stage the master manages the promising solutions received from slaves. Since an excess of cooperation may have a great impact in diversity in these kind of problems, a incoming candidate solution is broadcasted only when it significantly improves the current *BestKnownSol*. The master process is able to self-tuning the cooperation threshold based on the number of incoming solutions that are refused with the current threshold. When a new incoming solution promotes to a cooperative solution spread to the rest of the slaves, there is an increment on the score of the slave that achieved that solution.

The master process also manages the slaves adaptation requests. Each island identifies if it is not progressing in the search. An island will ask the master for a reconfiguration when promising cooperative solutions are arriving from the master but it cannot improve its local best known solution. Then, the master uses the information of the scoreboard to communicate the new configuration settings to that island. Finally, if the master receives a termination message from one of the slaves, it broadcast the termination request to the rest.

In the slaves, some steps are included to implement cooperation and self-tuning. First, a reception memory buffer keeps the messages arriving from the master that have not been processed yet, thus, the communications are all done in a non-blocking asynchronous way. The slave inspects its reception memory buffer looking for new best solutions from the master. When new solutions have arrived, the slave checks whether the new solutions improve the local *BestKnownSol* or not. If a new solution improves the local one, this new solution upgrades to *BestKnownSol* and it replaces a cooperation entry in the process population. The loop to check the reception of new solutions must be repeated until there are no more shared solutions to attend. This is because the execution time of one external iteration may be very different from one process to another. Thus, while a process has completed only one external iteration, their neighbors may have completed more, and several messages from the master may be waiting in the reception buffer. Then, the slave also checks the reception of new reconfiguration settings. Note that, the request for a reconfiguration is also a non-blocking operation. This means that the slave goes on with its execution until the message with the reconfiguration settings arrive. Besides, in the reception step, the slave also checks the arrival of termination messages from the master. If a termination message arrives, the slave finishes its execution.

After the reception step, the slave checks if its best local solution improves in, at least, an ϵ the *BestKnownSol*. If so, *BestKnownSol* is updated with the best local solution and the slave sends the promising result to the master.

To conclude the iteration, an adaptive step is accomplished. Each slave decides if it is progressing in the search based on:

- *Number of evaluations performed since its last cooperation*:

$$N_{eval} > N_{par} \times 500$$

  where N_{eval} is the number of evaluations performed by this process since its last cooperation with the master and N_{par} is the number of parameters of the problem.
- *Balance between the received and sent solutions*:

$$recvSolutions > (4 \times sendSolutions) + 10$$

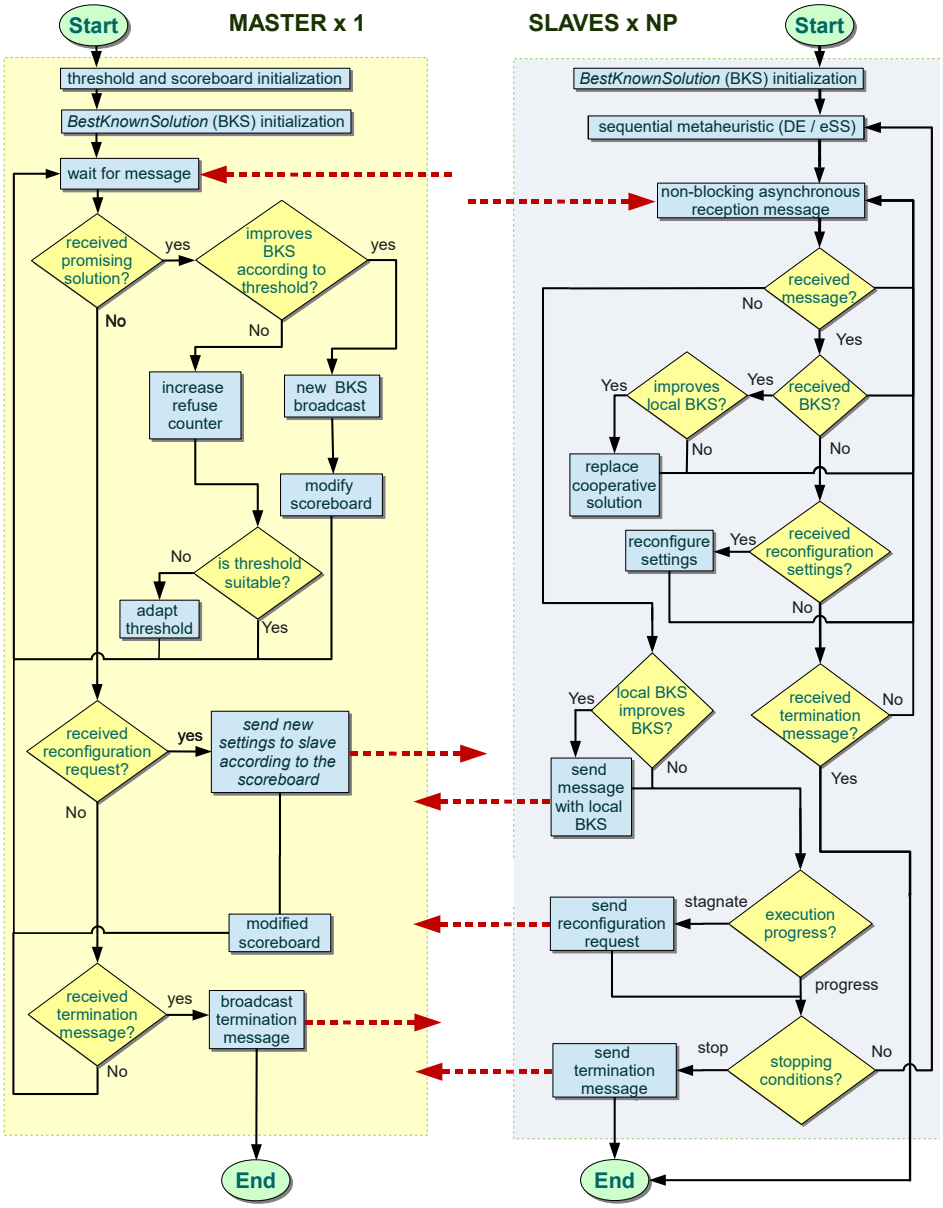

Figure 2: Summary of saCMM algorithm.

adaptation is requested when the number of received solutions is significantly greater than the number of solutions sent (with a minimum value of 10, to avoid requests at the beginning of the process), that is, if other slaves are cooperating much more than itself.

In summary, if a process recognizes that it has stagnated, it sends a request for reconfiguration to the master process. In response to these requests, the master sends to those slaves the most encouraging settings, i.e., those that are on the top of the scoreboard. In order to inject further diversity into those reconfigured islands, most of the members of their population (all of them except the two most promising solutions) are randomly re-initialized.

The saCMM algorithm repeats the external loop until the stopping criterion is met. Three different stopping criteria (or any combination of them) may be used: maximum number of evaluations, maximum execution time and a *value-to-reach (VTR)*.

To better illustrate how the self-adaptive mechanism works, Figure 3 shows, as an example, a Gantt diagram of a given execution of the saCMM method, using 10 islands. Five islands initiate DE with different configurations, from aggressive ones (that perform frequent local searches) to conservative ones (that do not perform local searches or perform them only sporadically). Another five

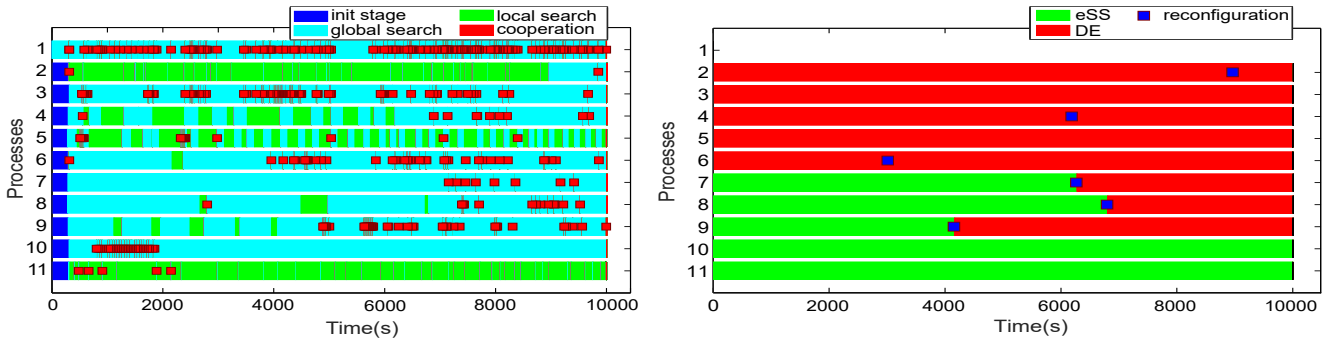

Figure 3: Gantt diagrams representing the processes and cooperation among them during the execution progress. Process 1 is the central master, and processes 2-11 the islands. At the left diagram, red dots represent asynchronous cooperation communications between master and islands, light blue areas represent global search steps and green areas represent local search steps. At the right diagram, red areas represent DE islands, green areas represent eSS islands and dark blue dots represent reconfiguration requests.

islands initiate eSS, also with different aggressive and conservative configurations. The red dots represent the cooperation between islands. It can be observed that DE outperforms eSS for this problem, and also that conservative configurations, with few local searches, outperform aggressive islands. Thus, when the unsuccessful islands request for a reconfiguration, the master sends them the configuration of the promising islands. In this example processes 2, 4 and 6, executing DE with absence of cooperations during a long time, request a reconfiguration and change from an aggressive to a conservative one, improving their results and begining to cooperate again. Also processes 7, 8 and 9, executing eSS without progress, request for a reconfiguration, and they end up executing a conservative DE. Since the most promising configuration can change during the execution progress, not all the processes executing a particular method are allowed to be reconfigured. In this example, two eSS islands are not reconfigured, one conservative (process 10) and one aggressive (process 11).

3 REVERSE ENGINEERING OF BIOLOGICAL NETWORKS

Reverse engineering aims to infer, analyze and understand the functional and regulatory mechanisms that govern the behavior of biological systems. This is addressed combining mathematical modeling with experiments. Most of this models need to explain dynamic behavior, and they are usually composed of different types of differential equations [22].

The logic-based ordinary differential equations (ODE) framework has been found particularly useful in modeling cell signalling [12, 23]. Here, we follow a mixed-integer global optimization approach for the problem of reverse engineering signaling [12, 18]. The problem of identifying the logic gates is formulated as a simultaneous model selection and parameter identification problem. From the optimization point of view, this corresponds to a mixed integer dynamic optimization (MIDO) problem.

The general mixed-integer dynamic optimization problem (MIDO), also called mixed-integer optimal control (MIOC) problem, is usually formulated as finding the set of discrete (integer or binary), time-dependent (stimuli or controls) and time-independent parameters, to optimize (minimize or maximize) a pre-defined cost function (which in optimal control is generally called performance index), while satisfying a set of dynamic and algebraic constraints. In mathematical form, it is usually formulated as follows:

Find $\mathbf{u}(t)$, $\mathbf{i}(t)$, $\mathbf{p}$ and t_f so as to minimize (or maximize):

$$J = G_{t_f}(\mathbf{x}, \mathbf{u}, \mathbf{i}, \mathbf{p}, t_f) + \int_{t_0}^{t_f} F(\mathbf{x}(t), \mathbf{u}(t), \mathbf{i}(t), \mathbf{p}, t)\mathrm{d}t \tag{1}$$

subject to:

$$\mathbf{f}(\dot{\mathbf{x}}(t), \mathbf{x}(t), \mathbf{u}(t), \mathbf{i}(t), \mathbf{p}, t) = 0, \qquad \mathbf{x}(t_0) = \mathbf{x}_0 \tag{2}$$

$$\mathbf{g}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{i}(t), \mathbf{p}, t) \leq 0, \qquad l = \overline{1, m_e + m_i} \tag{3}$$

$$\mathbf{u}_L \leq \mathbf{u}(t) \leq \mathbf{u}_U, \tag{4}$$

$$\mathbf{i}_L \leq \mathbf{i}(t) \leq \mathbf{i}_U, \tag{5}$$

$$\mathbf{p}_L \leq \mathbf{p} \leq \mathbf{p}_U, \tag{6}$$

where $\mathbf{x}(t) \in X \subseteq \mathrm{R}^{n_x}$ is the vector of state variables, $\mathbf{u}(t) \in U \subseteq \mathrm{R}^{n_u}$ is the vector of real valued control variables, $\mathbf{i}(t) \in I \in \mathrm{Z}^{n_i}$ is the vector of integer control variables, $\mathbf{p} \in P \subseteq \mathrm{R}^{n_p}$ is the vector of time-independent parameters, t_f is the final time of the process, m_e, m_i represent the number of equality and inequality constraints, $\mathbf{f}$ is the set ordinary differential equations describing the dynamics of the system (plus the corresponding initial conditions), $\mathbf{g}$ is the set of state constraints (path, pointwise and final time constraints), and $\mathbf{u}_L, \mathbf{i}_L, \mathbf{p}_L, \mathbf{u}_U, \mathbf{i}_U, \mathbf{p}_U$ correspond to the lower and upper bounds for the control variables and the time-independent parameters.

Methods for the numerical solution of DO problems can be broadly classified under three categories: dynamic programming, indirect and direct approaches. Indirect and direct approaches are the most promising strategies for realistic problems, due to the *curse of dimensionality* that suffer the dynamic programming [4] methods. Indirect approaches are based on the transformation of the original problem into a multi-point boundary value problem using Pontryagin's necessary conditions [13]. Direct methods are based on discretization of the control (sequential strategy [21]), or both the control and the states (simultaneous strategy [5]).

In our solution we use the direct approach described in [12], that consists of a transformation step, transcribing the original MIDO problem into a mixed-integer nonlinear programming (MINLP) problem, and a numerical solution step, where the actual solution of the MINLP problem is obtained by means of the multimethod proposed.

4 EXPERIMENTAL RESULTS

The new saCMM method described in Section 2 has been applied to the synthetic signaling pathway (SSP) [15] case study. This benchmark considers a dynamic model composed of 26 ordinary differential equations and 86 continuous parameters. It was initially used to illustrate the capabilities and limitations of different formalisms related with logic-based models. Although this is a synthetic problem, it was designed to be a plausible representation of a signaling transduction pathway. The model was used to generate pseudo-experimental data for 10 combinations of perturbations with two extracellular ligands (TNF and EGF) and two kinase inhibitors (for PI3K and RAF1). From a total of 26 dynamic states, 6 were observed (NFKB, P38, AP1, GSK3, RAF1 and ERK) and 5% of Gaussian noise was added to the data.

Following the methodology described in [19], we obtained an expanded version of this model containing every possible AND/OR logic gate given the initial graph structure. This so-called expansion procedure generated a nested model comprising 34 additional variables, one for each hyperedge. Thus, the obtained optimization problem contains 120 parameters, being 86 continuous and 34 binaries. The model and experimental setup were implemented using AMIGO [2] and exporting C code which could be used with the saCMM method presented here.

Experiments were deployed with default settings in the North Europe region of the Microsoft Azure public cloud using a virtual cluster with A9 instances. The A9 instances are for compute-intensive workloads having 16 cores each, Intel Xeon E5-2670 @2.60GHz processors, and 112GB of RAM.

The results shown in this section were analyzed both from a horizontal view [11], that is, assessing the performance by measuring the time needed to reach a given target value, and from a vertical view, that is, assessing the performance for a predefined effort. To evaluate the efficiency from a horizontal view, a stopping criteria based on a *value-to-reach (VTR)* is used. The VTR used was set to a low (challenging) value of 10. To evaluate the efficiency from a vertical view, the stopping criteria used is a predefined execution time. Thus, the experiments combine the two stopping criteria, a VTR and a maximum execution time. Due to the substantial dispersion of the results, because of the stochastic nature of these methods, each experiment was performed 20 times and a statistical study was carried out. Different experiments have been conducted to compare the proposed saCMM multimethod versus single method performance and to assess its scalability when the number of available computational resources increases.

First, the saCMM performance has been compared with two different single method parallel strategies:

- a self-adaptive cooperative strategy using only DE (saCDE). Diversity is introduced alike in saCMM but using only DE, that is, each island executes a separate DE with different configuration parameters (i.e. mutation factor and/or crossover constant).
- a self-adaptive cooperative strategy using only eSS (saCeSS). Diversity is introduced alike in saCMM but using only eSS, that is, each island executes a separate eSS with different configuration parameters (i.e. dimension of the reference set or balance parameter for the selection of initial points for the local solvers).

Table 1: Performance of the proposed self-adaptive multimethod compared with other parallel single method approaches. Stopping criteria using VTR=10.

method	#iter.	#evals	avg. exec. time (s)
saCDE	207	457708,10	5861±1256
saCeSS	67	607193,80	7819±5864
saCMM	104	339786,42	4429±1166

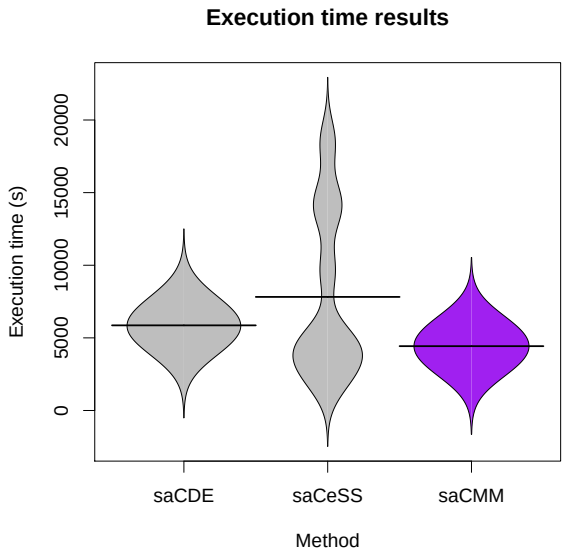

Figure 4: Beanplots of the execution time results in the experiments reported in Table 1.

Table 1 shows results for an horizontal evaluation, that is, using as stopping criterium only a challenging VTR. The table includes results of the average number of iterations performed in each experiment (*#iter.*), the average number of evaluations performed (*#evals*), and the final execution time when the VTR is reached (*avg. exec. time*). As it can be observed, the saCMM outperforms the other two methods in average execution time, but it is also noticiable the reduction in the dispersion of the results.

Since the results reported in Table 1 hide the underlying distribution, that in this kind of stochastic problems is very important, in Figure 4 the distribution of the results is shown using beanplots. The figure illustrates that saCMM reduces the variability of the results. As it can be seen, saCeSS presents a large variability for

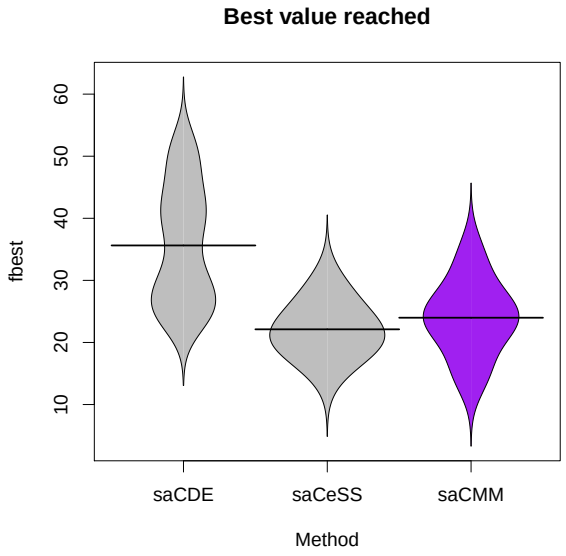

Figure 5: Beanplots of the best value achieved in the experiments reported in Table 2.

this problem. It is the method that achieves the minimum execution time among all the experiments, but also the one that achieves the worst execution times in some experiments. On its turn, saCDE seem to be more robust than saCeSS for this problem with less variability. Finally, saCMM presents the best results, both for average execution time and low disperion of the results.

Table 2 shows results for a vertical view, that is a maximum time (*max. time*) allowed is used as stopping criteria. The table shows results for the average value of the best value obtained (*avg. fbest*), the average number of iterations performed in each experiment (*iter.*), the average number of evaluations performed (*#evals*), the final execution time if the VTR is reached (*avg. exec. time*), and the percentage of executions that achieve the VTR before the maximum time allowed (*%hits*). The VTR for these experiments was relaxed to a VTR=15. It can be observed that, in these vertical evaluation, saCeSS outperforms saCDE, since this method obtains a lower average best value in the maximum effort (time) allowed. Figure 5 shows the beanplots for the distribution of the best value achieved in the different instances of this experiments. Although this results may seem inconsistent with the conclusion obtained from the horizontal view above, Figure 6 explains the situation. As it can be seen, saCDE and saCeSS perform different in different moments of the search. For instance, saCeSS outperforms saCDE at the begining of the search, while saCDE outperforms saCeSS later on. The objective of the multimethod algorithm is to follow (and sometimes even improve) the most sucessful approach at each time step, thanks to the cooperation among the islands and the reconfiguration on execution time.

Table 3 and Table 4 show for saCMM the impact in the search of increasing the diversity by increasing the number of islands, both from a vertical view and from a horizontal one, respectively.

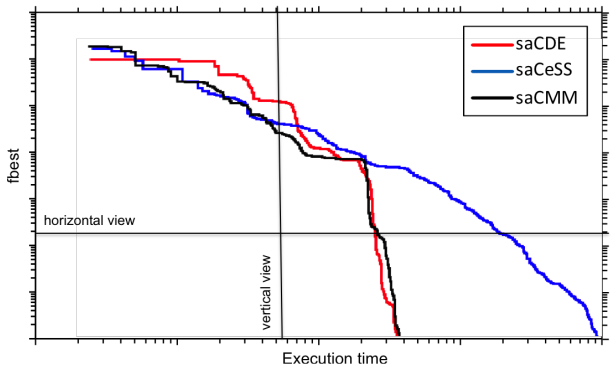

Figure 6: Vertical and horizontal views illustrated in a convergence graph.

Different experiments were performed using 10, 20 and 40 islands to assess the scalability of the proposal. Initially, for all the experiments, half of the islands perform DE and the other half eSS with different initial populations and configuration parameters. As can be seen, an increase in the number of islands, for these experiments, does not really impact neither the average best value achieved when the stopping criteria used is a predefined time effort, nor the average execution time when the stopping criteria is a challenge VTR. This is not surprising because, at the moment, we are assesing the performance of the multimethod with only two different metaheuristics. Note that the goal of the method is the injection of diversity in the search, and, thus, the inclusion of more metaheuristics in a near future is a must. However, these results also point to another future direction, which is the implementation of a fine-grain parallelization for each metaheuristic executed in the islands. This way, resources could be balanced between those dedicated to accelerate the cost function evaluation (which is the most time consuming part of metaheuristics, and can be done in parallel), and those dedicated to improve results through diversification in the search.

5 CONCLUSIONS

In this paper, we propose an extension of the self-adaptive cooperative multimethod (saCMM), a parallel cooperative strategy for NLP problems, with new mechanisms and extensions to handle MINLP problems. To this end, the following features have been included in the new implementation: (1) an efficient mixed-integer local solver (MISQP), (2) a novel self-adaption mechanism to avoid convergence stagnation, and (3) the injection of extra diversity during the adaptation steps, restarting most of reference set of the reconfigured processes.

The evaluation has been carried out in a virtual cluster built in the Microsoft Azure cloud. The proposal shows good performance results, compared to other self-adaptive parallel single methods, when applied to a signaling pathway case study from the domain of computational systems biology. The computational results show that the proposal reduces the execution time needed to obtain a reasonable quality solution. These results confirm that the method

Table 2: Performance of the proposed self-adaptive multimethod compared with other parallel single method approaches. Stopping criteria using VTR and maximum time allowed.

method	VTR	avg. fbest	#iter.	#evals	max. time (s)	avg. exec. time (s)	%hits
saCDE	15	35,63±9,91	47	121717,30	1500	1500	0%
saCeSS	15	22,12±4,08	12	122571,00	1500	1500	0%
saCMM	15	23,99±6.06	34	120749,60	1500	1479	10%

Table 3: Impact in saCMM of the diversification by increasing the number of islands. Vertical evaluation, using a maximum time effort as stopping criteria.

#Islands	VTR	avg. fbest	iter.	#evals	max. time (s)	avg. exec. time (s)	%hits
10	15	23,99±6.06	34	120749,60	1500	1479	10%
20	15	25,99±2,06	29	241733,00	1500	1500	0%
40	15	22,41±5,08	26	482820,83	1500	1374	10 %

Table 4: Impact in saCMM of the diversification by increasing the number of islands. Horizontal evaluation, using VTR=10 as stopping criteria.

method	#iter.	#evals	avg. exec. time (s)
10	104	339786,42	4429±1166
20	88	708206,57	4895±1664
30	91	1368397,00	4164±3358

can be used to reverse engineer dynamic models of biological pathways. Future work will focus on improving the current implementation by means of an hybrid scheme that combines the current coarse-grained parallelization with a fine grain parallelization in each island. This will improve the scalability of the approach allowing to balance the computational resources between diversity and intensity in the search.

ACKNOWLEDGMENTS

This research received financial support from the Spanish Government through the projects DPI2017-82896-C2-2-R and TIN2016-75845-P (AEI/FEDER, UE), and from the Galician Government under the Consolidation Program of Competitive Research Units (Network Ref. R2016/045 and Project Ref. ED431C 2017/04), all of them co-funded by FEDER funds of the EU. We also acknowledge Microsoft Research for being awarded with a sponsored Azure account.

REFERENCES

[1] Bree B Aldridge, John M Burke, Douglas A Lauffenburger, and Peter K Sorger. 2006. Physicochemical modelling of cell signalling pathways. *Nature cell biology* 8, 11 (2006), 1195–1203.

[2] Eva Balsa-Canto and Julio R Banga. 2011. AMIGO, a toolbox for advanced model identification in systems biology using global optimization. *Bioinformatics* 27, 16 (2011), 2311–2313.

[3] Julio R Banga. 2008. Optimization in computational systems biology. *BMC Systems Biology* 2, 1 (2008), 47.

[4] Dimitri P Bertsekas, Dimitri P Bertsekas, Dimitri P Bertsekas, and Dimitri P Bertsekas. 1995. *Dynamic programming and optimal control.* Athena Scientific Belmont, MA.

[5] Lorenz T Biegler, Arturo M Cervantes, and Andreas Wächter. 2002. Advances in simultaneous strategies for dynamic process optimization. *Chemical Engineering Science* 57, 4 (2002), 575–593.

[6] Samuel Burer and Adam N. Letchford. 2012. Non-convex mixed-integer nonlinear programming: A survey. In *Surveys in Operations Research and Management Science*. 97–106.

[7] Jose A Egea, Eva Balsa-Canto, María S G García, and Julio R Banga. 2009. Dynamic optimization of nonlinear processes with an enhanced scatter search method. *Industrial & Engineering Chemistry Research* 48, 9 (2009), 4388–4401.

[8] Oliver Exler, Thomas Lehmann, and Klaus Schittkowski. 2012. A comparative study of SQP-type algorithms for nonlinear and nonconvex mixed-integer optimization. *Mathematical Programming Computation* 4, 4 (2012), 383–412.

[9] Oliver Exler and Klaus Schittkowski. 2007. A trust region SQP algorithm for mixed-integer nonlinear programming. *Optimization Letters* 1, 3 (2007), 269–280.

[10] Patricia González, David R. Penas, Xoan C. Pardo, Julio R. Banga, and Ramón Doallo. 2018. Multimethod optimization in the cloud: A case study in systems biology modelling. *Concurrency and Computation: Practice and Experience* 30, 12 (2018).

[11] N. Hansen, A. Auger, S. Finck, and R Ros. 2009. *Real-Parameter Black-Box Optimization Benchmarking 2009: Experimental Setup.* Technical Report RR-6828. INRIA.

[12] David Henriques, Miguel Rocha, Julio Saez-Rodriguez, and Julio R. Banga. 2015. Reverse engineering of logic-based differential equation models using a mixed-integer dynamic optimization approach. *Bioinformatics* 31, 18 (2015), 2999–3007.

[13] Daniel Liberzon. 2012. *Calculus of variations and optimal control theory: a concise introduction.* Princeton University Press.

[14] Sean Luke. 2009. *Essentials of metaheuristics.* Vol. 113. Lulu Raleigh.

[15] Aidan MacNamara, Camille Terfve, David Henriques, Beatriz Peñalver Bernabé, and Julio Saez-Rodriguez. 2012. State–time spectrum of signal transduction logic models. *Physical Biology* 9, 4 (2012), 045003.

[16] D.R. Penas, J.R. Banga, P. González, and R. Doallo. 2015. Enhanced parallel Differential Evolution algorithm for problems in computational systems biology. *Applied Soft Computing* 33 (2015), 86–99.

[17] D.R. Penas, P. González, J. A. Egea, R. Doallo, and J.R. Banga. 2017. Parameter estimation in large-scale systems biology models: a parallel and self-adaptive cooperative strategy. *BMC Bioinformatics* 18, 1 (2017), 52.

[18] David R. Penas, David Henriques, Patricia González, Ramón Doallo, Julio Saez-Rodriguez, and Julio R. Banga. 2017. A parallel metaheuristic for large mixed-integer dynamic optimization problems, with applications in computational biology. *Plos One* 12, 8 (2017).

[19] Julio Saez-Rodriguez, Leonidas G Alexopoulos, Jonathan Epperlein, Regina Samaga, Douglas A Lauffenburger, Steffen Klamt, and Peter K Sorger. 2009. Discrete logic modelling as a means to link protein signalling networks with functional analysis of mammalian signal transduction. *Molecular Systems Biology* 5 (2009), 331.

[20] R. Storn and K Price. 1997. Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization* 11, 4 (1997), 341–359.

[21] VS Vassiliadis, RWH Sargent, and CC Pantelides. 1994. Solution of a class of multistage dynamic optimization problems. 1. Problems without path constraints. *Industrial & Engineering Chemistry Research* 33, 9 (1994), 2111–2122.

[22] Alejandro F Villaverde and Julio R Banga. 2014. Reverse engineering and identification in systems biology: strategies, perspectives and challenges. *Journal of the Royal Society Interface* 11, 91 (2014), 20130505.

[23] Dominik M. Wittmann, Jan Krumsiek, Julio Saez-Rodriguez, Douglas A. Lauffenburger, Steffen Klamt, and Fabian J. Theis. 2009. Transforming boolean models to continuous models: methodology and application to T-cell receptor signaling. *BMC Systems Biology* 3, 1 (2009), 98.