RESEARCH ARTICLE

A parallel metaheuristic for large mixed-integer dynamic optimization problems, with applications in computational biology

David R. Penas[1], David Henriques[1], Patricia González[2]*, Ramón Doallo[2], Julio Saez-Rodriguez[3,4]*, Julio R. Banga[1]*

1 BioProcess Engineering Group. IIM-CSIC, Spanish National Research Council. C/Eduardo Cabello, 6. E-36208 Vigo, Spain, **2** Computer Architecture Group. Department of Electronics and Systems. University of A Coruña. Campus de Elviña s/n. 15071 A Coruña, Spain, **3** RWTH Aachen University, Faculty of Medicine, Joint Research Centre for Computational Biomedicine, Aachen D-52074, Germany, **4** European Molecular Biology Laboratory, European Bioinformatics Institute (EMBL-EBI), Hinxton, CD10 1SD, United Kingdom

* pglez@udc.es (PG); saezrodriguez@combine.rwth-aachen.de (JSR); julio@iim.csic.es (JRB)

OPEN ACCESS

Citation: Penas DR, Henriques D, González P, Doallo R, Saez-Rodriguez J, Banga JR (2017) A parallel metaheuristic for large mixed-integer dynamic optimization problems, with applications in computational biology. PLoS ONE 12(8): e0182186. https://doi.org/10.1371/journal.pone.0182186

Editor: Jordi Garcia-Ojalvo, Universitat Pompeu Fabra, SPAIN

Received: March 24, 2017

Accepted: July 13, 2017

Published: August 15, 2017

Data Availability Statement: All relevant data are within the paper and its Supporting Information files.

Funding: This research received financial support from the Spanish Ministerio de Economía y Competitividad (and the FEDER) through projects "SYNBIOFACTORY" (DPI2014-55276-C5-2-R) and TIN2016-75845-P (AEI/FEDER, UE), and from the Galician Government under the Consolidation Program of Competitive Research Units (Network

Abstract

Background

We consider a general class of global optimization problems dealing with nonlinear dynamic models. Although this class is relevant to many areas of science and engineering, here we are interested in applying this framework to the reverse engineering problem in computational systems biology, which yields very large mixed-integer dynamic optimization (MIDO) problems. In particular, we consider the framework of logic-based ordinary differential equations (ODEs).

Methods

We present saCeSS2, a parallel method for the solution of this class of problems. This method is based on an parallel cooperative scatter search metaheuristic, with new mechanisms of self-adaptation and specific extensions to handle large mixed-integer problems. We have paid special attention to the avoidance of convergence stagnation using adaptive cooperation strategies tailored to this class of problems.

Results

We illustrate its performance with a set of three very challenging case studies from the domain of dynamic modelling of cell signaling. The simpler case study considers a synthetic signaling pathway and has 84 continuous and 34 binary decision variables. A second case study considers the dynamic modeling of signaling in liver cancer using high-throughput data, and has 135 continuous and 109 binaries decision variables. The third case study is an extremely difficult problem related with breast cancer, involving 690 continuous and 138 binary decision variables. We report computational results obtained in different infrastructures, including a local cluster, a large supercomputer and a public cloud platform. Interestingly, the results show how the cooperation of individual parallel searches modifies the systemic properties of the sequential algorithm, achieving superlinear speedups compared

ref. R2016/045 and competitive reference groups GRC2013/055). DRP acknowledges financial support from the MICINN-FPI programme.

Competing interests: The authors have declared that no competing interests exist.

to an individual search (e.g. speedups of 15 with 10 cores), and significantly improving (above a 60%) the performance with respect to a non-cooperative parallel scheme. The scalability of the method is also good (tests were performed using up to 300 cores).

Conclusions

These results demonstrate that saCeSS2 can be used to successfully reverse engineer large dynamic models of complex biological pathways. Further, these results open up new possibilities for other MIDO-based large-scale applications in the life sciences such as metabolic engineering, synthetic biology, drug scheduling.

Introduction

Global optimization is being increasingly used in engineering and across most basic and applied sciences [1–4], including areas from life sciences such as bioinformatics and computational systems biology [5–9]. In the case of chemical and biological processes, during the last decade there has been a growing interest in modelling their dynamics [10], i.e. developing kinetic models which are able to encapsulate the time-varying nature of these systems. As a consequence, many research efforts are now being invested in exploiting those dynamic models by mathematical optimization techniques. These formulations belong to the class of dynamic optimization problems (or open loop optimal control). The most general formulation is that of mixed-integer dynamic optimization (MIDO), where part of decision variables are discrete (binary or integer) [11].

Although many dynamic optimization problems consider how to extract useful operating policies and/or designs from a dynamic model, such formulations can also be applied to the model building process itself, i.e. to the so-called reverse engineering problem [12–19], which is known to be extremely hard [10].

Here we consider this general problem of reverse engineering in computational biology by means of mixed-integer nonlinear dynamic optimization (MIDO). Our goal is to be able to handle large-scale nonlinear kinetic models, so we focus on the solution of this class by means of suitable global optimization methods. Broadly speaking, MIDO problems can be solved using deterministic or stochastic global optimization methods. In the case of deterministic methods, many advances have been made in recent years (see [11, 20–22] and references therein). Although these deterministic methods can guarantee global optimality in some cases, unfortunately they suffer from lack of scalability, i.e. the associated computational effort increases very rapidly with problem size.

Alternatively, although stochastic algorithms for global optimization cannot offer guarantees of global optimality, they usually reach a region near to the global solution in acceptable execution time, at least for small and medium scale problems. However, for larger problems the computational cost of purely stochastic methods can be very large [5, 23]. Several hybrid approaches [24–28] have tried to benefit from the best of both approaches by combining global stochastic methods with efficient (local) deterministic optimization methods. In this context, metaheuristics (i.e. guided heuristics) have been particularly successful, ensuring the proper solution of these problems by adopting a global stochastic optimization approach, while keeping the computational effort under reasonable values thanks to efficient local optimization solvers [29, 30].

Here we consider the general MIDO problem as described above. We present a new parallel method based on extensions of an enhanced scatter search metaheuristic [30, 31] which has shown good performance in simpler non-linear programming and optimal control problems. We formulate the reverse engineering problem as a MIDO using the framework of logic-based ordinary differential equations (ODEs) [32, 33]. In this framework, the logic components need to be estimated along with the continuous parameters in order to describe the dynamic behavior defined by the structure of the system of ODEs. The resulting problem is very hard due to its multimodal, non-linear and highly constrained nature.

Our contribution resides at the interface between computational systems biology and computer science (high performance computing), with an emphasis on using the latter to help the former. The merits of the logic-based ODEs framework have been illustrated previously [32, 33]. But, as already recognized in Henriques et al [33], more work was needed regarding the computational efficiency of the optimization methods. This has been precisely our main objective here: how to improve the robustness and efficiency of the numerical solution of these problems by developing a better algorithm and its corresponding high-performance computing implementation.

In order to be able to handle these more complex MIDO problems in realistic times, we have focused on developing a parallel cooperative strategy that scales up well with problem size. Parallel strategies for metaheuristics have been a very active research area during the last decade [34, 35]. In the area of computational biology, parallel methods have already shown promising results in non-linear optimization problems ([36, 37]). For the case of mixed-integer nonlinear programming, a few researchers have considered the development of parallel methods [38–40], but there is a lack of studies in the case of the mixed-integer nonlinear dynamic optimization problems considered here.

The aim of this paper is to explore this direction further considering extensions of a recent parallel self-adaptive implementation of the enhanced Scatter Search algorithm [41] so it can handle general MIDO problems of realistic size. It should be noted that there are several key differences and important novel aspects with respect to our previous work:

- we extend and generalize the formulation considered in Henriques et al [33] by adopting a generic mixed-integer optimal control approach, without using relaxations and/or transformations of the original problem during the solution strategy
- we present a new solution strategy based on a parallel cooperative optimization method with specific extensions to handle large mixed-integer problems, and new mechanisms of self-adaptation tailored to this class of problems
- we illustrate the performance of our approach by considering a set of very challenging case studies, obtained in different high-performance computing infrastructures, including traditional parallel machines and a public cloud-computing platform. In particular, we show how our new method can successfully exploit cloud computing resources

The organization of this paper is as follows. In the Background section we present the general mixed-integer dynamic optimization formulation and we outline the solution strategy, which is based on the control vector parameterization direct method. This solution approach transforms the problem into a master mixed-integer nonlinear programming problem with an inner initial value problem. We solve the outer problem by means of an improved parallel metaheuristic, which is described in the following section. We then evaluate this approach considering several challenging reverse engineering case studies. We evaluate the performance of the proposal using these cases on a local cluster, a large supercomputer and in the cloud (using Microsoft Azure), demonstrating its good efficiency and scalability, as discussed in the

Results and Discussion Section. Finally, in the Conclusions Section we summarize the main contributions of this work.

Background

In this section we present the general statement of the class of problems considered. Next, we describe the numerical approach used to solve it, based on the so-called control vector parameterization direct method. We then describe the background regarding the scatter search metaheuristic which will used as the basis for the parallel method presented in the following section.

Mixed integer dynamic optimization problem

The general mixed-integer dynamic optimization problem (MIDO), also called mixed-integer optimal control (MIOC) problem [21], 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 generically 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. In the formulation above, known as the general Bolza problem, G_{t_f} is a terminal cost function, and F is an integral cost function.

The MIDO formulation above can be used to solve problems from widely different areas, including aeronautics, chemical engineering, mechanical engineering, transport, medicine, systems biology, synthetic biology and industrial biotechnology [11, 20, 33, 42–50]. In the particular context of reverse engineering complex biological networks [33], our aim is to use the above framework to simultaneously identify the underlying network topology, its regulatory structure, the time-dependent controls (e.g. stimuli) and time-invariant model parameters, consistent with existing experimental data (time-series). An alternative would be to carry out parameter estimation based on real-values for each individual and possible model structure, but this option becomes prohibitively expensive for any realistic case.

Solution strategy

Methods for the numerical solution of dynamic optimization (optimal control) problems can be broadly classified under three categories: dynamic programming, indirect and direct approaches. Dynamic programming [51, 52] suffers form the so called *curse of dimensionality*, so the latter two are the most promising strategies for realistic problems. Indirect approaches were historically the first developed, and are based on the transformation of the original optimal control problem into a multi-point boundary value problem using Pontryagin's necessary conditions [53, 54]. Direct methods are based on discretization of the control (sequential strategy [55]), or both the control and the states (simultaneous strategy [56]).

Here, our strategy uses a direct approach and consists of a first (transformation) step, (transcribing the original MIDO problem into a mixed-integer nonlinear programming problem, MINLP), followed by a second (numerical solution) step (the actual solution of the MINLP by the novel cooperative scatter search metaheuristic). We have chosen the control parameterization approach [11], that consists in discretizing the control variables ($\mathbf{u}(t)$ and $\mathbf{i}(t)$) into a number of elements, and then approximating the controls in each element by means of certain basis functions. The control variables are, thus, parameterized using $\mathbf{w_u} \in R^{\rho}$ and $\mathbf{w_i} \in Z^{\rho}$, which become time-invariant decision variables. With this approach, the original problem is transformed from an infinite dimensional problem into a finite dimension mixed-integer non-linear programming outer problem, that can be solved using a suitable MINLP solver. Note that, as a consequence, the evaluation of the objective function and constraints requires the solution of an inner problem (the system dynamics), by a suitable initial value problem (IVP) solver.

In summary, our strategy results in a numerical optimization problem composed of:

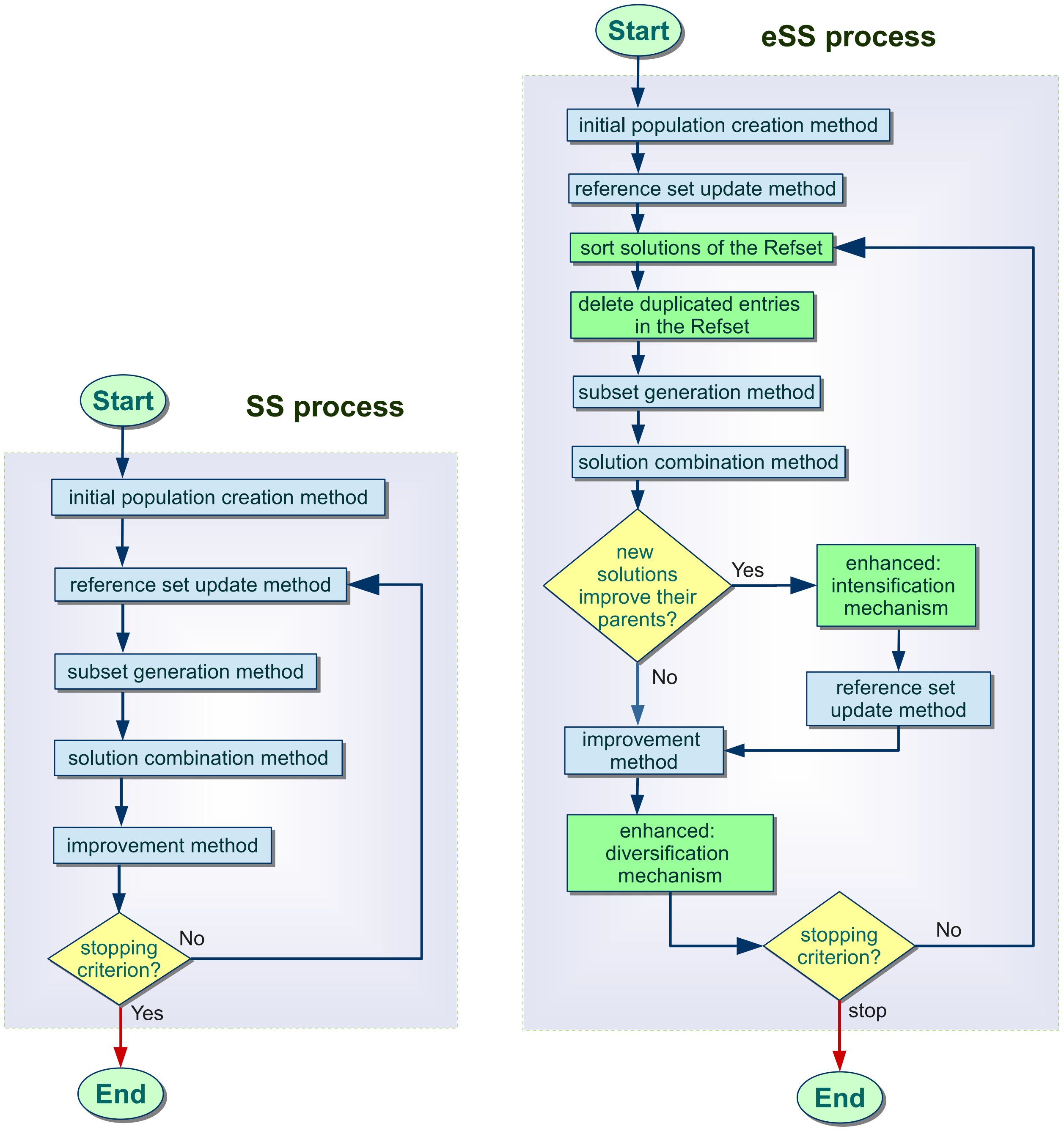
- an outer mixed-integer nonlinear programming (MINLP) problem: due its non-convexity, we need to use global optimization methods, as already mentioned in the introduction. Based on our previous experiences with different stochastic global methods and their hybrids with local solvers [24, 26, 30, 31, 41, 57, 58], here we decided to extend a metaheuristic based on scatter search, combining it with a efficient local MINLP solver [59], as described below.
- an inner initial value problem (IVP), i.e. the nonlinear dynamics that need to be integrated for each evaluation of the cost functional and constraints. Here we solve the IVP using the state-of-the-art solvers for numerical integration of differential equations included in the SUNDIALS package [60]. It should be noted that local optimization methods to be used require the numerical computation of gradients of the objective and/or constraints with respect to the decision variables. If this is the case, an efficient procedure is to use first order parametric sensitivities to compute such information [61]. The sensitivity equations can be obtained by a chain rule differentiation of the system defined in Eq 2 with respect to the decision variables. They can be efficiently solved in combination with the original system. Here we have used SUNDIALS [60], which includes CVODES, an efficient sensitivity solver.

We now proceed to detail the parallel metaheuristic method developed for solving this problem.

Scatter search and recent extensions

Scatter Search (SS) [62] is one of the most popular metaheuristics to solve global optimization problems. It can be regarded as a population-based algorithm that creates new solutions through iterative steps of diversification, improvement, combination and population update. Compared to other metaheuristics, SS uses a low number of population members.

The SS strategy involves 5 steps, illustrated in Fig 1(a): (1) SS begins by producing an initial population of solutions within the search space; (2) the initial *Reference Set* is then created with


Start
SS process
initial population creation method
reference set update method
subset generation method
solution combination method
improvement method
stopping criterion?
No
Yes
End
Start
eSS process
initial population creation method
reference set update method
sort solutions of the Refset
delete duplicated entries in the Refset
subset generation method
solution combination method
new solutions improve their parents?
Yes
enhanced: intensification mechanism
No
reference set update method
improvement method
enhanced: diversification mechanism
stopping criterion?
No
stop
End

(a) Schematic representation of a basic SS algorithm

(b) Schematic representation of a basic eSS algorithm

Fig 1. Schematic representation of sequential algorithms.

https://doi.org/10.1371/journal.pone.0182186.g001

a set of representative solutions of the population; (3) a generation method selects a subset of solutions from the reference set; (4) the solutions in this subset are combined to obtain new solutions; (5) finally, an improvement procedure is applied to the previous solutions. The update method (step 2) creates again the reference set for the next iteration, and this procedure is repeated until the end of the search.

A recent implementation of this procedure, named *enhanced Scatter Search* (eSS) [30, 31], presents a straightforward yet effective design that helps to beat well known issues in nonlinear dynamic systems optimization such as flat areas, noise, nonsmoothness, and/or discontinuities. Fig 1(b) graphically shows the main functionalities of the eSS. Novel mechanisms, highlighted in green in the figure, are included to achieve a good trade-off between intensification (local search) and diversification (global search):

- A rather small population size is used, even for large-scale problems. However, more search directions are allowed than in the original SS by means of a new combination scheme. The diversity in the search is preserved while the number of evaluations required is not increased.
- A new intensification mechanism in the global phase exploits the promising directions defined by a pair of solutions in the reference set.
- A heuristic local search method accelerates the convergence, specially for large-scale problems.
- It makes use of memory to infer whether a solution become stagnant in a local optimum or whether it is alongside of already visited solutions.

Despite the success of the strategies included in the eSS, for large-scale problems involving dynamic systems, it still requires significant computation times. Besides, the eSS method needs the tuning of a number of configuration settings that may have a great impact in the algorithm performance, thus requiring a number of initial exploratory runs and, therefore, further increasing the computation times. With the aim of solving these issues, we recently developed a parallel method named self-adaptive Cooperative enhanced Scatter Search (saCeSS) [41] and demonstrated its advantages for the solution of hard parameter estimation problems involving nonlinear dynamic models. Essentially, the saCeSS method is a novel parallel metaheuristic that follows an island-model strategy where a set of independent eSS threads (islands) exchange information (solutions and settings) among them to improve the convergence through cooperation, effectively implementing a self-tuning mechanism of the algorithm. Several key functionalities have been included in saCeSS in order to overcome the limitations of eSS:

- a coarse-grained parallelization following a master-slave model, where the master manages the control of the cooperation between slaves (islands), since an excessive of cooperation results in adverse impacts on diversity
- an exchange of information handled taking into account the quality of the solutions obtained by each individual process, as an alternative to time elapsed, to achieve more effective cooperation between processes
- an asynchronous communication protocol to tackle the exchange of information between processes, avoiding inactive processes when they are waiting for information exchanged from other processes
- a self-adaptive mechanism in the master process that performs a scoreboard used to dynamically tune the settings of the islands based on their individual progress

A parallel cooperative scatter search for mixed integer optimization

Since the saCeSS method described above has demonstrated its potential for solving very challenging non-linear programming (NLP) problems, there was an interest in extending this method so it can be applied to large mixed-integer nonlinear programming and mixed-integer dynamic optimization problems. As a result, we present here a new method, saCeSS2, resulting from modifications and extensions of the original saCeSS method along three main directions:

1. addition of an efficient local solver for mixed-integer optimization problems
2. changes to the self-adaption mechanisms in order to avoid premature stagnation of the convergence
3. addition of new mechanisms to ensure diversity while keeping parallel cooperation

Regarding the local solver, we have incorporated a trust region sequential quadratic programming solver, called *Mixed-Integer Sequential Quadratic Programming* (MISQP) [59, 63]. 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.

The preliminary tests applying the previous saCeSS algorithm to mixed-integer problems using the MISQP local solver brought to light a problem of premature convergence due to a quick lose of diversity in the islands. Although both eSS and saCeSS algorithms include their own mechanisms to maintain the desired diversity during the algorithm progress, we observed that in mixed-integer problems a promising incoming solution in an island acted as an attractor for the members of the *RefSet*, bringing them fast to the vicinity of this new value. Thus, we introduced two new strategies in the saCeSS2 method to allow for a dynamic break-out from local optima, and to further preserve the diversity in the search for these problems, avoiding prematurely stagnation:

- first, we needed to avoid the premature convergence observed in MINLP problems (cooperation between islands decreases too fast as the algorithm converges, since many of them stagnate). Thus, the criteria used in the original saCeSS method to trigger the reconfiguration (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.
- Second, we observed that in mixed-integer problems, when an island stagnates, most of the times is due to the lost of diversity in the *RefSet*. Thus, we decided to further promote diversity by a modified strategy: once an island requests a reconfiguration, most of the members of the *RefSet*, except for two solutions, are randomly initialized again.

Fig 2 summarizes the new saCeSS2 method for MINLP/MIDO problems. The master process is in charge of the control of the cooperation and the scoreboard for the islands' tuning. At the beginning of the algorithm, both at master and at slaves, 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 the 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 appearance of good solutions received from slaves. Then, with the aim of controlling the cooperation between slaves, only when the incoming candidate solution significantly improves the current *BestKnownSol*, this variable is updated and broadcasted. The master

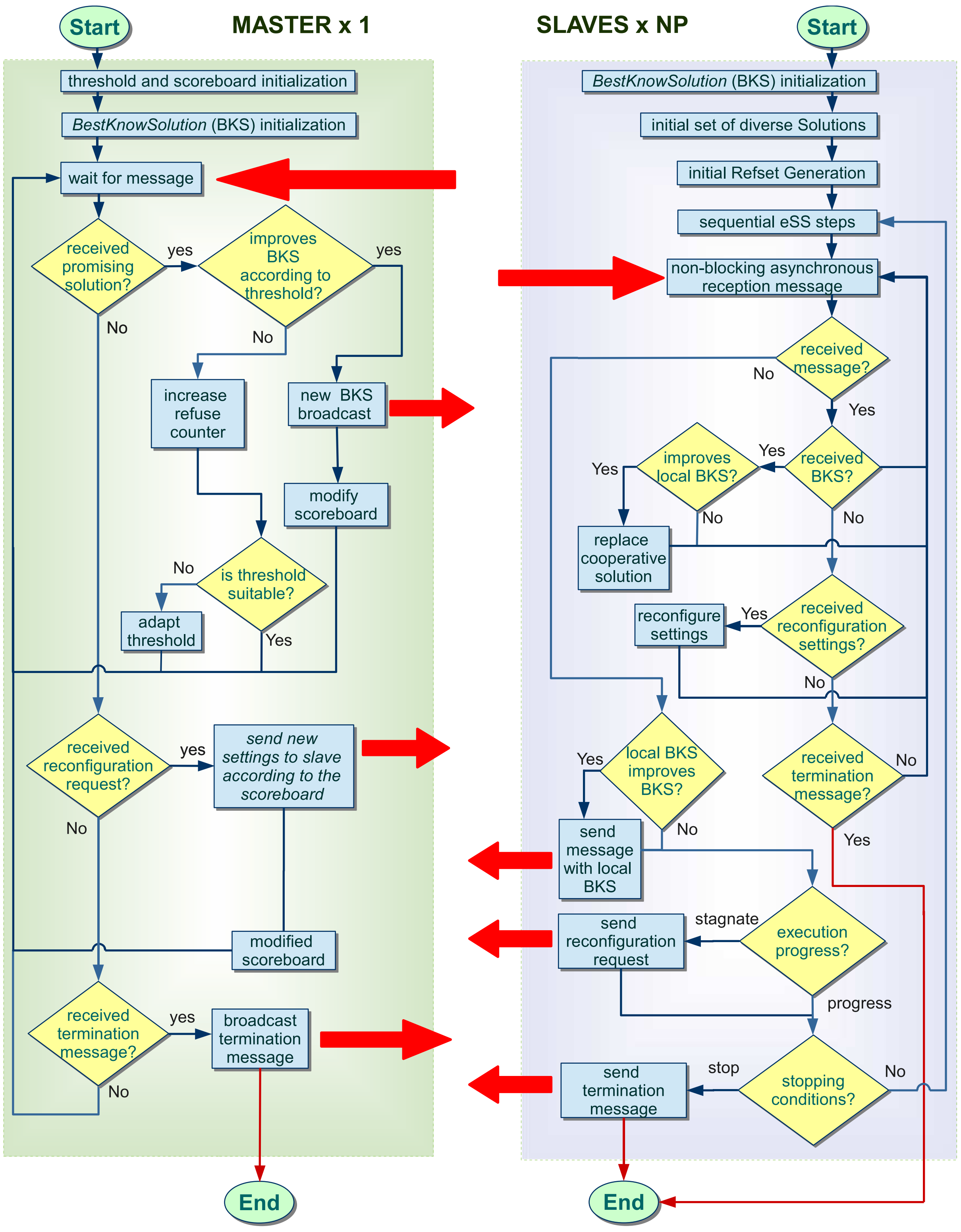

MASTER x 1
Start
threshold and scoreboard initialization
BestKnowSolution (BKS) initialization
wait for message
received promising solution?
yes
improves BKS according to threshold?
yes
No
No
increase refuse counter
new BKS broadcast
modify scoreboard
is threshold suitable?
No
adapt threshold
Yes
received reconfiguration request?
yes
send new settings to slave according to the scoreboard
No
modified scoreboard
received termination message?
yes
broadcast termination message
No
End
SLAVES x NP
Start
BestKnowSolution (BKS) initialization
initial set of diverse Solutions
initial Refset Generation
sequential eSS steps
non-blocking asynchronous reception message
received message?
No
Yes
received BKS?
Yes
improves local BKS?
Yes
replace cooperative solution
No
No
received reconfiguration settings?
Yes
reconfigure settings
No
local BKS improves BKS?
Yes
send message with local BKS
No
received termination message?
No
Yes
execution progress?
stagnate
send reconfiguration request
progress
stopping conditions?
stop
send termination message
No
End

Fig 2. Schematic representation of saCeSS2 algorithm.

https://doi.org/10.1371/journal.pone.0182186.g002

process is able to self-tuning the cooperation threshold based on the number of incoming solutions that are refused with the current criterion. Besides, when a new incoming solution deserves to become 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. To accurately identify those islands that are not progressing in the search, the master process would need additional information from slaves. The solution implemented is that each slave resolves whether it has stagnated or not. If promising cooperative solutions are arriving from the master but the island cannot improve its local best known solution, it will ask the master for a reconfiguration. Then, the master will communicate to that island one of the configuration settings of the islands on the top of the scoreboard. Finally, if the master receives a termination message from one of the slaves, it broadcast the termination request to the rest.

The slaves perform the classic steps of the sequential eSS. Additionally new steps are included to implement cooperation and self-tuning. First, a reception memory buffer retains 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*. Then, the slave also checks the reception of new reconfiguration settings. Note that, as already explained, all the communications between slaves and master are asynchronous, thus, 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. The ϵ used in the slaves is different from the ϵ used in the master process. The slaves use a smaller ϵ so that many *promising* solutions are sent to the master. The master has to make a decision on which of those incoming solutions should be spread to the rest of the slaves. This decision is based on the quality of the incoming solutions. Thus, the ϵ used by the master begins with high value and decreases as long as the number of refused solutions get larger and no incoming solution overcomes the current ϵ.

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*:

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

  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*:

$$\textit{recvSolutions} > (4 \times \textit{sendSolutions}) + 10$$

  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 *RefSet* are randomly re-initialized.

The saCeSS2 algorithm repeats the external loop until the stopping criterion is met. Three different stopping criteria (or any combination among them) may be used in current saCeSS2 implementation: maximum number of evaluations, maximum execution time and a *value-to-reach* (*VTR*). While the *VTR* is usually known in benchmark problems, for a new problem, the *VTR* will be, in general, unknown.

Applications in computational systems biology

The aim of reverse engineering in biological systems is to infer, analyze and understand the functional and regulatory mechanisms that govern their behavior, using the interplay between mathematical modeling with experiments. Most of this models need to explain dynamic behavior, so they are usually composed of different types of differential equations. However, reverse engineering in systems biology has to face many pitfalls and challenges, especially regarding the ill-conditioning and multimodality of these inverse problems [10]. Below we consider several cases related with cell signalling processes and show how these issues can be surmounted with the methodology presented here.

Reverse engineering of cell signalling

Reverse engineering of cell signaling phenomena is a particularly important area in systems biology [64]. In complex organisms, signaling pathways play a critical role in the behavior of individual cells and, ultimately, in the organism as a whole. Cells adapt to the environmental conditions through the integration of signals released by other cells via endocrine or paracrine secretion as well as other environmental stimuli. Fundamental cellular decisions such as replicate, differentiate or die (apoptosis) are largely controlled by these signals [65].

Many of the interactions involved in signaling are commonly grouped in pathways. Pathways are typically depicted as sequences of steps where the information is relayed upon activation by an extracellular receptor promoting several downstream post translational modifications, which will ultimately end by modifying gene expression or some other effector. These interactions are dynamic, i.e. the behavior of such pathways is known to be highly dependent on the cell type and context [66], which change with time [67]. Additionally, many of these pathways interact with each other in ways that are often described as analog to a decision making process [68]. Further, the dynamics of cell signaling are rather fast processes, specially if compared with metabolism or even gene expression.

There are at least three good reasons to infer a dynamic model of a signaling pathway. The first, and perhaps most obvious one, is to find novel interactions. The second is model selection, defined as the process of using data to select (or exclude) a number of model features which are consistent with the current knowledge about a given system. This is particularly relevant when comparing different cell types or a specific cell type in its healthy and diseased status, such as cancer. The third one is the usage of such a model to predict how the system will behave in new conditions that have not been tested before.

In order to build a mechanistic dynamic model for a given cell type or tissue, we need values for its parameters. These are rarely available, and a common strategy is to find them by training the model to data. The most informative data for signal transduction is obtained upon perturbation experiments, where typically a system (assumed to be homeostatic initially) is stimulated with different chemicals to which the cell may (or not react), and the variations in the cell biochemistry are recorded. The resulting time-series of data are then used to reverse engineer a dynamic model of the signalling process.

Following subsections describe the so called logic-based ordinary differential equations (ODE) framework, which has been found particularly useful in modeling cell signalling, and its problem statement as a MIDO. Then, in the results, we present three very challenging case studies of increasing complexity, which are then solved with the parallel metaheuristic presented in this study.

Logic-based dynamic models

Logic models were first applied to biological systems by [69] to model gene regulatory networks. Since then, applications to multiple contexts have been made [70, 71] and diverse modifications from the original formalism have been developed [72]. In particular, various extensions have been developed to accommodate continuous values (e.g. [32, 73–78]). Amongst these formalisms, logic-based ODEs are one of the best options to handle time series with precision [33, 77]. The main idea is to convert the logic models into their continuous ODE-based equivalent but without the need of mechanistic (kinetic information). However, since it is composed of differential equations, we can use it to carry out dynamic simulations and e.g. predict dynamic trajectories of variables of our interest. A number of different methods have been proposed transform Boolean logic models into ODE homologues [74, 75, 77].

Basically, Boolean models describe the flow of information in a biological system using discrete binary states (logic decisions). In other words, each state $x_i \in \{0, 1\}$ is represented by a binary variable can be updated according to a Boolean function $B_i(x_{i1}, x_{i2}, \ldots, x_{iN}) \in \{0, 1\}$ of its N inputs (x_{ij}). A typical simple example is the situation where a protein can be phosphorylated in two sites by different kinases, and both interactions are needed to activate the protein. This can be modeled as a logic AND gate. Alternatively, when two different kinases can phosphorylate the same site, independently activating the downstream signaling, we can describe it as a logic OR gate. In another situation, if a signal inhibits the propagation of another one, we can describe it with a NOT gate. In summary, logic models can be represented by an hypergraph with AND/OR/NOT gates.

In the logic-based ODE formalism, we transform each Boolean update function into a continuous equivalent $\bar{B}_i \in [0, 1]$, where the states $\bar{x}_i \in [0, 1]$ can take continuous values between 0 and 1. Their dynamic behaviour is then modelled as:

$$\dot{\bar{x}}_i = \frac{1}{\tau_i} \cdot (\bar{B}_i(\bar{x}_{i1}, \bar{x}_{i2}, \ldots, \bar{x}_{ij}) - \bar{x}_i) \tag{7}$$

where τ_i can be regarded as a sort of life-time of x_i.

HillCubes [77] have been developed, based on multivariate polynomial interpolation, for the above purpose. They include Hill kinetics (which are known to provide a good approximation of the dynamics of e.g. gene regulation). HillCubes are obtained via a transformation method from the Boolean update function. An example is shown in Table 1, illustrating how an OR gate would be transformed by multi-linear interpolation [77] into a BoolCube ($\bar{B}^I$):

$$\bar{B}^I \quad (\bar{x}_1, \ldots, \bar{x}_N) = \sum_{x_1=0}^{1} \cdots \sum_{x_N=0}^{1} \left[B(x_1, \ldots, x_N) \cdot \prod_{i=1}^{N} (x_i\bar{x}_i + [1 - x_i][1 - \bar{x}_i]) \right] \tag{8}$$

Table 1. Relation between functions $B(x_1, x_2)$ and $\bar{B}'(\bar{x}_1, \bar{x}_2)$.

x_1	x_2	$B(x_1, x_2)$	$\bar{B}'(\bar{x}_1, \bar{x}_2) = \ldots$
0	0	0	$0 \cdot (1-\bar{x}_1) \cdot (1-\bar{x}_2)+$
0	1	1	$1 \cdot (1-\bar{x}_1) \cdot \bar{x}_2+$
1	0	1	$1 \cdot \bar{x}_1 \cdot (1-\bar{x}_2)+$
1	1	1	$1 \cdot \bar{x}_1 \cdot \bar{x}_2$

A truth table helps to understand the relationship between the OR Boolean update function $B(x_1, x_2)$ and its continuous homologue $\bar{B}'(\bar{x}_1, \bar{x}_2)$. For every combination of the Boolean variables x_1 and x_2, a term is added to $\bar{B}'(\bar{x}_1, \bar{x}_2)$) depending on $B(x_1, x_2)$.

https://doi.org/10.1371/journal.pone.0182186.t001

Although BooleCubes are accurate homologues of Boolean functions, they fail to represent the typical sigmoid shape switch-like behavior often present in molecular interactions [79]. The latter can be achieved by replacing the $\bar{x}_i$ by a Hill function:

$$f^H(\bar{x}_i) = \frac{\bar{x}_i^{\,n}}{\bar{x}_i^{\,n} + k^n} \tag{9}$$

or the normalized Hill function:

$$f^{Hn}(\bar{x}_i) = \frac{f^H(\bar{x}_i)}{f^H(1)} \tag{10}$$

Further details regarding logic-based ODE models can be found in [77].

Problem statement as a MIDO

In order to find the best logic-based dynamic model to represent the behavior of a given biological network, we developed a formulation extending previous works that used a Boolean logic framework [80] or a constrained fuzzy-logic formalism [81]. The idea here is that starting from a directed graph containing only the interactions and their signs (activation or inhibition) we can build an expanded hypergraph containing all the possible logic gates.

The problem can be formulated as the following for case studies 1 and 2:

$$\begin{aligned}
\underset{n,k,\tau,w}{\text{minimize}} \quad & F(n,k,\tau,w) = \sum_{\epsilon=1}^{n_\epsilon} \sum_{o=1}^{n_{\mathrm{O}}{}^{\epsilon}} \sum_{s=1}^{n_{\mathrm{S}}{}^{\epsilon,o}} (\tilde{y}_{\mathrm{S}}{}^{\epsilon,o} - y_{\mathrm{S}}{}^{\epsilon,o})^2 \\
\text{subject to} \quad & \mathcal{E}_{\text{sub}} = \{e_{\mathrm{i}} | w_{\mathrm{i}} = 1\},\ i = 1,\ldots,n_{\text{hyperedges}} \\
& \mathcal{H}_{\text{sub}} = (V, \mathcal{E}_{\text{sub}}) \\
& \mathrm{LB}_n \le n \le \mathrm{UB}_n \\
& \mathrm{LB}_k \le k \le \mathrm{UB}_k \\
& \mathrm{LB}_\tau \le \tau \le \mathrm{UB}_\tau \\
& \dot{\bar{x}} = f(\mathcal{H}_{\text{sub}}, \bar{x}, n, k, \tau, t) \\
& \bar{x}(t_0) = \bar{x}_0 \\
& y = g(\mathcal{H}_{\text{sub}}, \bar{x}, n, k, \tau, t)
\end{aligned} \tag{11}$$

where $\mathcal{H}_{\text{sub}}$ is the subgraph containing only the hyperedges ($\mathcal{E}_{\text{sub}}$), defined by the binary variables w. Additionally n, k and τ are the continuous variables required for the logic-based ODE scheme. Upper and lower bounds represent the limits to these parameters. The model dynamics ($\dot{x}$) are given by the function f. This set of differential equations varies according to the subgraph (and therefore also according to the integer variables vector w). Predictions for the systems dynamics are obtained by solving the initial value problem given by the ODEs. The objective function is the mismatch (e.g. norm-2) between the simulated (y) and the experimental data ($\tilde{y}$), and we seek to minimize this metric for every experiment (ϵ), observed species (o) and sampling point (s). The simulation data y is given by an observation function g of the model dynamics at time t.

In case study 3 we also consider a model reduction problem where additional decision variables are used to remove the influence of a regulator $\bar{x}_i$ from the model. As starting point we consider a model derived with SELDOM [82], where a mutual information strategy, combined with dynamic optimization, was used to find an ensemble of dynamic models that can explain the data from four breast-cancer cell-lines used in the DREAM-HPN challenge [83]. One of the critical steps in SELDOM was to perform model reduction using a greedy heuristic. Here we consider instead the application of mixed-integer global optimization with saCeSS2 to the problem of model reduction. To find a reduced model we use the Akaike information criterion (AIC), which for the purpose of model comparison is defined as:

$$AIC = 2K + 2n \cdot ln\left(\frac{F}{n}\right), \tag{12}$$

where K is the number of active parameters. The theoretical foundations for the AIC can be found in [84].

Results and discussion

The new saCeSS2 method described above has been applied to a set of case studies from the domain of computational systems biology with the goal of assessing its efficacy and efficiency for solving these very difficult MIDO/MINLP problems. The method has been compared with both the sequential eSS [31] and with an embarrassingly parallel non-cooperative version of the eSS called *np*-eSS. The *np*-eSS method consists of *np* separated eSS runs performed in parallel but without cooperation among them. The results reported for *np*-eSS correspond to the best value achieved in *np* runs. Diversity is introduced in these *np* eSS runs by allowing different settings to each one of the individual searches. The performance of saCeSS2 was also evaluated considering a different number of processors in order to study its scalability and the dispersion of results.

The original reported implementation of eSS [31] was coded in Matlab, thus, for a fair comparison with saCeSS2, it has been here implemented in F90. In the saCeSS2 algorithm the MPI library [85] has been employed for the cooperation between islands.

For the experimental testbed different platforms have been used. First, most of the experiments were conducted in a local cluster (NEMO) that consists of three nodes powered with two deca-core Intel Xeon E5-2650 CPUs with 30GB of RAM connected through a Gigabit Ethernet network. With the aim of assessing the scalability of the proposal we also performed some experiments in a larger infrastructure, the cluster from European Bioinformatics Institute (EBI) [86], that consists of 222 nodes powered with two octa-core Intel Xeon E5-2680 CPUs with 30GB of RAM, connected through a Gigabit Ethernet network. Finally, in order to evaluate the performance of the proposal in a public cloud, some experiments were conducted in the Microsoft Azure public cloud.

The saCeSS2 library has been compiled with the Intel implementations for C, FORTRAN and MPI library, except in the EBI Cluster, where GNU compilers and openMPI had to be used. We remark this fact due to the well-known differences in the performance obtained using different compilers.

The computational results shown in this paper were analyzed both from a horizontal view [87], that is, assessing the performance by measuring the time needed to reach a given target value, and from a vertical view [87], that is, evaluating how far a method has advanced in the search for a predefined effort. Thus, two different stopping criteria were considered in these experiments: solution quality based on a *value-to-reach* (*VTR*), for an horizontal view, and pre-defined effort using a maximum execution time, for a vertical approach. The *VTR* used was the optimal fitness value reported in [33]. Also, due to the natural dispersion in the results of stochastic methods, each experiment reported in this section has been performed 20 times and a statistical study was carried out.

Case study 1: Synthetic signaling pathway (SSP)

The synthetic signaling pathway (SSP) [72] case study 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 [80], 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. We proceeded by implementing the model and experimental setup using AMIGO [88] and exporting C code which could be used with the saCeSS2 method presented here.

Considering saCeSS2, it is important to note that the cooperation between processes changes the systemic properties of the eSS algorithm and therefore its macroscopic behavior. The same happens with the self-adaptive mechanism proposed. Table 2 displays for each method (sequential, parallel non-cooperative, and saCeSS2) the number of processors used (#np), the mean and standard deviation value of the achieved tolerances (*fbest*), the mean and standard deviation number of external iterations (*iter*) performed, the mean and standard

Table 2. Case study 1 SSP: Performance analysis from a horizontal view.

method	#np	mean fbest±std	mean iter±std	mean evals±std	mean time±std(s)	speedup
eSS	1	9.8±0.3	261±636	345989±829560	54885±131153	-
np-eSS	10	9.5±0.6	35±16	356583±127682	4546±1592	12.0
	20	9.8±0.2	29±6	626150±120338	4193±907	13.0
	40	9.8±0.1	33±9	876800±228596	2901±765	18.9
saCeSS2	10	9.7±0.2	25±9	246082±68925	3478±1114	15.7
	20	9.8±0.1	18±6	402613±120260	2779±870	19.7
	40	9.9±0.1	19±8	470746±142702	1602±523	34.2

Performance of the saCeSS2 and scalability analysis when the number of processors grows. Stopping criteria: VTR = 10.

https://doi.org/10.1371/journal.pone.0182186.t002

deviation number of evaluations (*evals*) needed to achieve the *VTR*, the mean and standard deviation execution time, and the speedup achieved versus the sequential method. As it can be seen, there is a notable reduction in the execution time required by the parallel methods against the sequential one, and there is also a significant reduction between the saCeSS2 method and the non-cooperative *np*-eSS. Note that, in the parallel methods (*np*-eSS and saCeSS2), the initial population, and, thus, the computational load, is not spread among processors. The population size is the same in the sequential method as in each of the islands in the parallel methods. That is, the parallel methods allow for a diversification in the search. Therefore, the speedup achieved versus the sequential method is due to the impact of this diversification, and the speedup achieved by saCeSS2 over the *np*-eSS is due to the impact of the cooperation between different searches, that produces results of higher quality performing less evaluations and, hence, providing a better performance. In short, these results show the effectiveness of the cooperative parallel algorithm proposed compared to a non-cooperative parallel version.

Table 3 shows results for experiments that include as stopping criterion a predefined effort of maximum execution time of 4000 seconds. This table displays the percentage of executions (% hit) that achieve a very high quality solution (VTR = 9.0). It can be observed that the sequential implementation never achieved the VTR in the maximum allowed time, while, for the parallel implementations, when the number of processes grows the number of the executions that achieved the quality solution increased. Again, the cooperative proposed saCeSS2 implementation achieved better results than the non-cooperative parallel version when using the same number of processors.

When dealing with stochastic optimization solvers, it is important to evaluate the dispersion of the computational results. Fig 3 illustrates with beanplots how the parallel algorithms (*np*-eSS and saCeSS2) reduce the variability of execution time and obtain less number of outliers when the number of cores increases. Hybrid violin/boxplot graphs for these results are included in Fig A in S1 File for a in depth insight. The proposed saCeSS2 method outperforms significantly the non-cooperative *np*-eSS method (note the logarithmic scale in axis y). This is an important feature of the saCeSS2, because it reduces the average execution time.

To better illustrate the goal of saCeSS2 method vs the non-cooperative parallel *np*-eSS implementation, Fig 4 shows the convergence curves, which represent the logarithm of the objective function value against the execution time. Fig 4 illustrates, for both saCeSS2 and *np*-eSS methods, the region between the lower and upper bounds of the 20 runs performed for each experiment, with a strong line representing the median value for each time moment.

In order to evaluate the scalability of the proposed saCeSS2, Fig 5 shows the convergence curves for those experiments that fall in the median values of the results distribution using 10,

Table 3. Case study 1 SSP: Performance analysis from a vertical perspective.

method	#np	mean fbest±std	mean iter±std	mean evals±std	mean time±std(s)	hits%
eSS	1	20.1±4.5	19±3	27289±4478	4000±0	0%
np-eSS	10	10.6±1.6	26±3	261664±24780	3966±119	10%
	20	10.31±0.91	24±3	522194±47322	3862±410	15%
	40	9.5±0.7	36±4	964168±99650	3681±510	30%
saCeSS2	10	10.2±1.4	23±4	256872±28063	3822±476	15%
	20	9.6±0.7	22±4	471948±82282	3532±706	35%
	40	8.9±0.2	32±19	621118±232204	2258±973	85%

Performance of saCeSS2 using as stopping criteria: VTR = 9 and maximum time = 4000 seconds.

https://doi.org/10.1371/journal.pone.0182186.t003

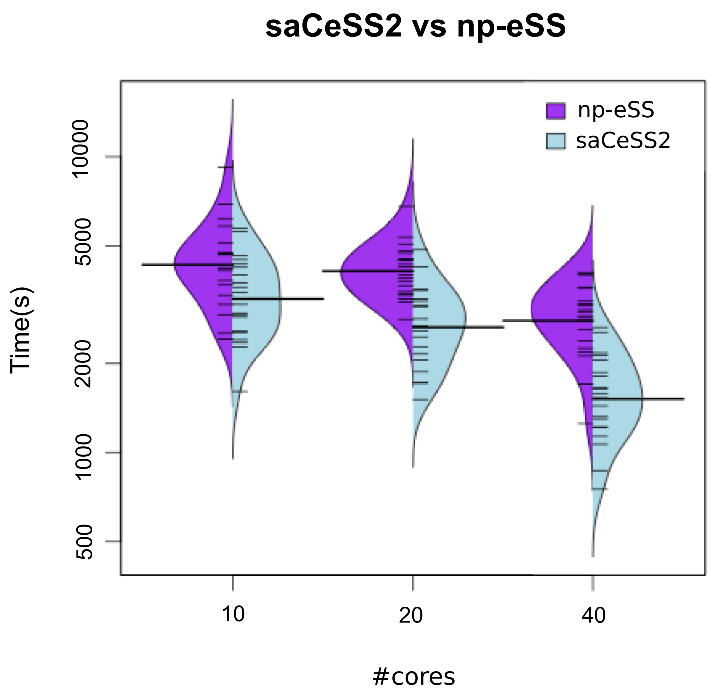


Fig 3. Case study 1 SSP: Bean plots of execution time for *np*-eSS vs saCeSS2 using 10, 20 and 40 MPI processors. VTR = 10 and 20 independent runs for each experiment.

https://doi.org/10.1371/journal.pone.0182186.g003

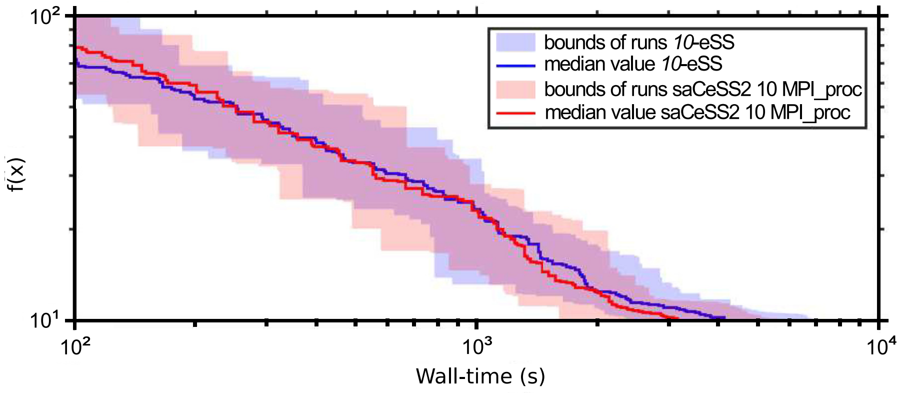
20 and 40 processors. It can be seen that the saCeSS2 still improves the convergence results when the number of processors grows. This improvement comes from the cooperation between islands and the diversification obtained through the exploration in parallel of different search regions using different algorithm settings.

Case study 2: HepG2

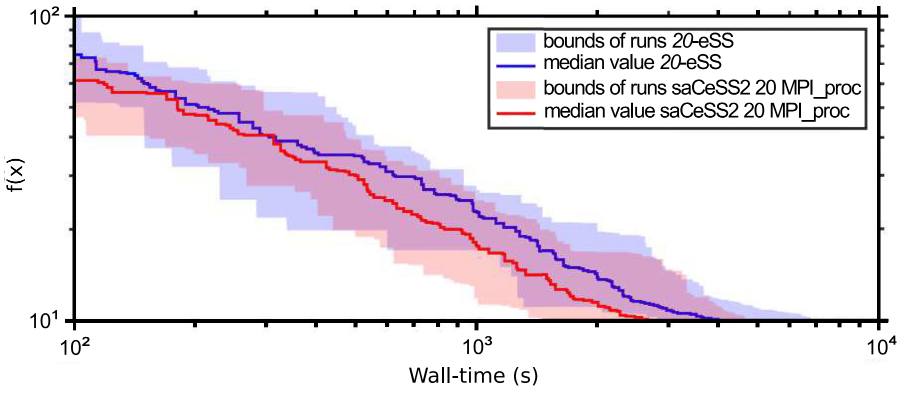
As a second case study, we consider the reverse engineering of a logic-based ODE model using liver cancer data (a subset of the data generated by [89]). The dataset consists of phosphorylation measurements from a hepatocellular carcinoma cell line (HepG2) at 0, 30 and 180 minutes after perturbation.

To preprocess the network, we used CellNOptR, the R version of CellNOpt [90]. Basically, the network was compressed to remove as many non-observable/non-controllable species as possible. Subsequently, we generated all gates that were compatible with the network; for this we added hyperedges (AND gates) from all pair of inputs (the OR gates are implicit). The expanded network has 109 hyperedges and 135 continuous parameters. To transform this model into a logic-based ODE model, we developed a parser that generates a C model file and Matlab scripts compatible with the AMIGO toolbox [88].

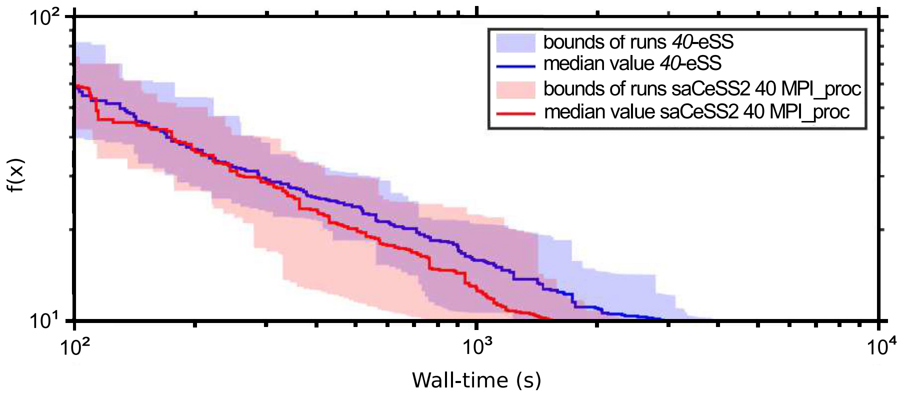
Consequently, in this case the optimization problem to solve contains a total of 244 parameters, being 135 continuous and 109 binaries. Although the time-series data contains only three sampling time points, it is quite rich from the point of view of information content: it includes 64 perturbations comprising 7 ligands and 7 small-molecule inhibitors. The ligands were chosen to activate inflammation and proliferation pathways, and the inhibitors to block the activity of specific kinases. We normalized the data by rescaling it to this range. This is required to as the models are semi-quantitative and hence the data has to be between 0 and 1. There are a



(a) Convergence curves for *10*-eSS vs saCeSS2 using 10 cores



(b) Convergence curves for *20*-eSS vs saCeSS2 using 20 cores



(c) Convergence curves for *40*-eSS vs saCeSS2 using 40 cores

Fig 4. Case study 1 SSP: Convergence curves.

https://doi.org/10.1371/journal.pone.0182186.g004

total of 25 states present in the model, 16 corresponded to observed species. The initial conditions for the other 9 species are not measured and we had to estimate them. To avoid increasing the problem size and multi-modality unnecessarily, the estimated initials conditions were assumed the same for each of the 64-experiments.

Table 4, similarly to Table 2, displays the performance of the different methods based on the number of external iterations, function evaluations and total execution time, for a different

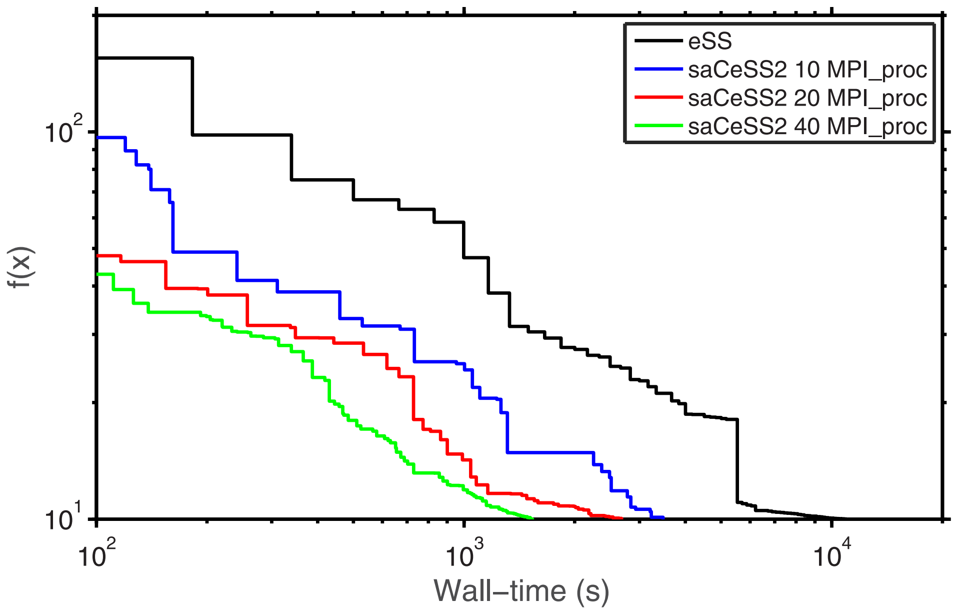


Fig 5. Convergence curves for saCeSS2 using 1, 10, 20 and 40 processors corresponding to the runs in the median values of the results distribution.

https://doi.org/10.1371/journal.pone.0182186.g005

number of processors. Note that results for the sequential method are not reported due to the unreasonable amount of time to reach convergence. Again, it can be seen that the saCeSS2 method outperforms, not only the sequential eSS, but also a parallel eSS without cooperation between islands. The cooperative strategy, along with the self-adaptive mechanism, leads to an important improvement in the convergence rate and the execution time required.

Table 5 shows results including as stopping criterion a lower VTR and a predefined effort of 30 hours. Since it is very difficult to reach a point of very high quality in this problem, this table displays the percentage of hits that achieve a VTR = 30. It can be observed that the sequential eSS never achieved the VTR in the maximum allowed time, while the parallel implementations achieve more hits as the number of processors grows. The saCeSS2 method clearly outperforms the embarrassingly parallel eSS: not only the mean time improves (around a 67% for 40 processors), but, which is more important, the number of runs that achieve the high quality VTR is larger (65% versus 35% for 40 processors).

Table 4. Case study 2 HepG2: Performance analysis from a horizontal view.

method	#np	mean fbest±std	mean iter±std	mean evals±std	mean time±std(s)
eSS	1	-	-	-	-
np-eSS	10	32.4±0.8	1493±2975	13581782±10598705	230483±365129
	20	32.5±0.7	527±381	20267424±12791177	142996±93617
	40	32.8±0.1	434±246	22157687±11660663	70221±35565
saCeSS2	10	32.5±0.5	1056±1873	13637565±20933642	167880±242658
	20	32.3±0.9	396±496	13196677±15577101	89433±108108
	40	32.4±1.0	560±431	11959105±9383238	44037±32346

Performance of the saCeSS2 and scalability analysis when the number of processors grows. Stopping criteria: VTR = 33.

https://doi.org/10.1371/journal.pone.0182186.t004

Table 5. Case study 2 HePG2: Performance analysis from a vertical view.

method	#np	mean fbest±std	mean iter±std	mean evals±std	mean time±std(s)	hits%
eSS	1	48.3±6.2	342±47	1010744±122417	108000±0	0%
np-eSS	10	34.9±3.7	482±112	8329751±1519410	103847±14515	10%
	20	32.9±1.9	418±65	16612721±2292759	103614±14763	10%
	40	31.1±1.3	604±180	30606532±8576193	93874±25243	35%
saCeSS2	10	34.7±3.8	403±146	7359986±1468640	103052±12461	10%
	20	32.0±4.4	339±163	12057460±4431722	85436±28200	45%
	40	30.4±1.2	786±478	20231060±11664025	63153±34832	65%

Convergence of saCeSS2 until a high quality solution is reached. Stopping criteria: VTR = 30 and maximum time = 108000 seconds.

https://doi.org/10.1371/journal.pone.0182186.t005

Fig 6 shows beanplots comparing the distribution of the execution times in the saCeSS2 method versus the non-cooperative parallel version. The figure illustrates not only the improvement in the mean execution time, but also the reduction in the variability of the execution times due to the cooperation and self-adaptive mechanism included in the saCeSS2 method. Hybrid violin/boxplots for this data are also provided in Fig B in S1 File for a thoroughly comprehension. Notice that the less the number of cores used the more outliers we obtain in the distribution.

Finally, Fig 7 shows the convergence curves for the previous experiments. Fig 7 shows the region between the lower and upper bounds of the 20 runs for each experiment. Fig 8, demonstrate the scalability of the proposal when the number of processors grows.

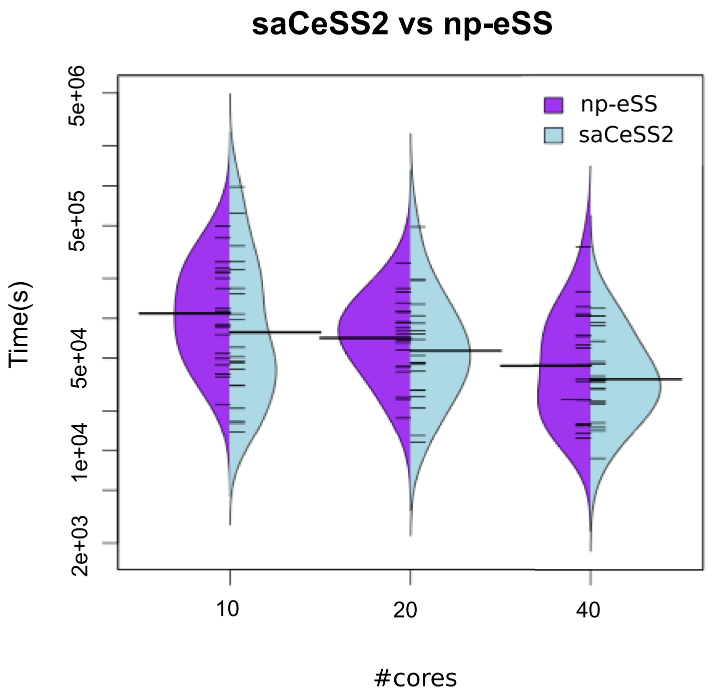
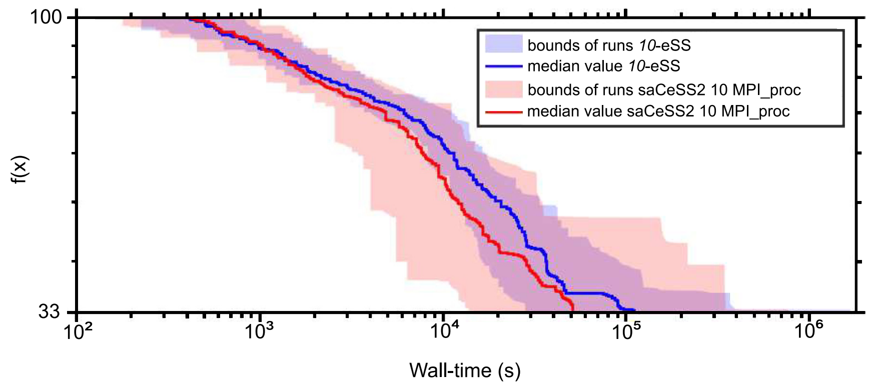
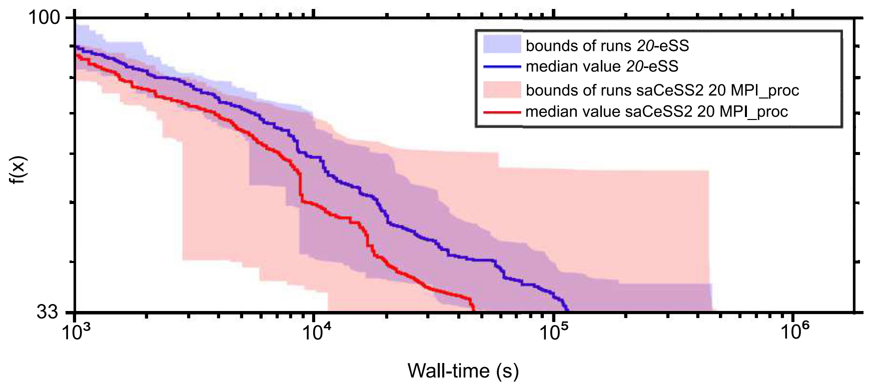


Fig 6. Case study 2 HePG2: Bean plots of execution time for *NP*-eSS vs saCeSS2 using 10, 20 and 40 MPI processors in the HePG2 problem. VTR = 33 and the number of runs is equal to 20.

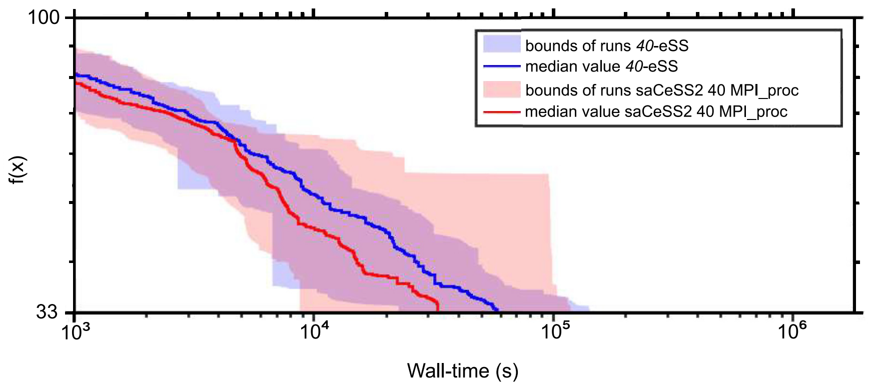
https://doi.org/10.1371/journal.pone.0182186.g006



(a) Convergence curves for *10*-eSS vs saCeSS2 using 10 cores



(b) Convergence curves for *20*-eSS vs saCeSS2 using 20 cores



(c) Convergence curves for *40*-eSS vs saCeSS2 using 40 cores

Fig 7. Case study 2 HePG2: Convergence curves.

https://doi.org/10.1371/journal.pone.0182186.g007

Case study 3: Breast cancer network inference challenge (HPN-DREAM)

We finally consider an extremely difficult problem which has been recently made publicly available in the context of the DREAM challenges [91]. The DREAM challenges provide a forum to crowdsource fundamental problems in systems biology and medicine, such as the inference of signaling networks [83, 92], in the form of collaborative competitions. This data-

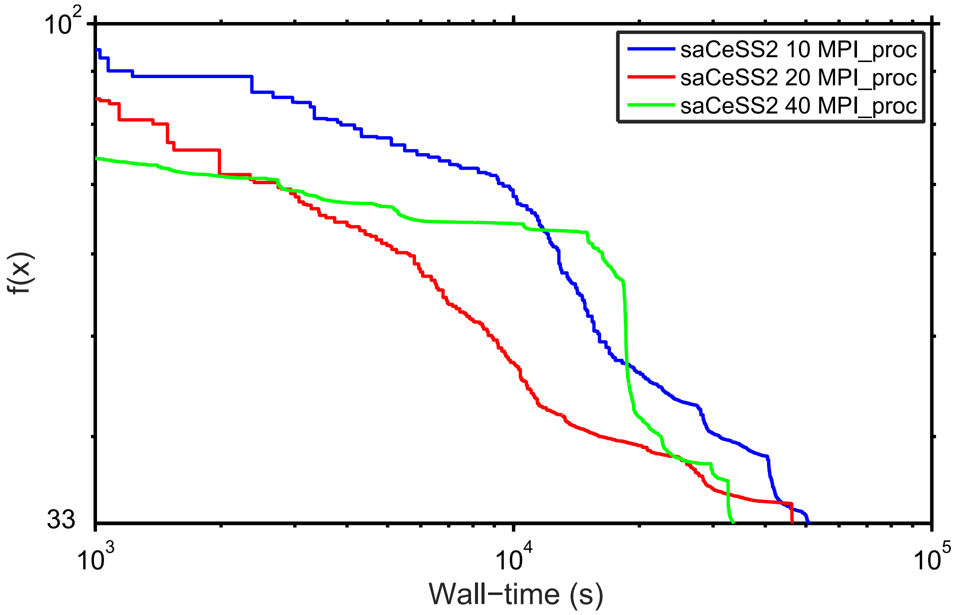


Fig 8. Convergence curves for saCeSS2 using 10, 20 and 40 processors corresponding to the runs in the median values of the results distribution.

https://doi.org/10.1371/journal.pone.0182186.g008

set comprised time-series acquired under eight extracellular stimuli, under four different kinase inhibitors and a control, in four breast cancer cell lines [83].

The HPN-DREAM breast cancer challenge is composed of two sub-challenges: (i) an experimental sub-challenge where the participants were asked to make predictions for 44 observed phosphoproteins (although the complete data-set was larger); and (ii) an *in silico* sub-challenge, where the participants were encouraged to exploit all the prior knowledge they could use and the experimental protocol along with the real names of the measured quantities, used reagents, inhibitors, etc. Using different combinations of inhibitors and ligands (on and off), the organizers if the challenge generated a data-set for several cell-lines. An additional data-set generated with the help of a fourth inhibitor was kept unknown to the participants, who were asked to deliver predictions for several possible inhibitors.

Overall, the problem contains a total of 828 decision variables (690 continuous and 138 binaries). Thus, the HPN-DREAM is an extremely challenging problem also from the computational view, with an enormous expected execution time and an unknown final target value. In a preliminary step, we carried out different experiments using np = 10, 20, and 40 cores in our NEMO local cluster to solve this problem. We used the np cores to run in parallel np independent eSS searches, without cooperation between them, and we also run a saCeSS2 execution using np processes. We used as stopping criterion for all the experiments a predefined effort of 10 days and we studied the convergence curves (shown in Fig 9). The blue region represents the bounds of the 40 sequential eSS runs, while the blue solid line represents the median value for each time moment of these 40 runs. The other solid lines represent the convergence curve of a single saCeSS2 performed using 10, 20, and 40 cores. The saCeSS2 method clearly outperforms the embarrassingly parallel eSS and shows a good scalability when the number of processes increases. We then performed new experiments using a larger number of cores in the EBI cluster. Fig 10 show the convergence curves using 100 and 300 cores. Due to the large amount of resources employed and the cluster policy, the length of the job (and, thus,

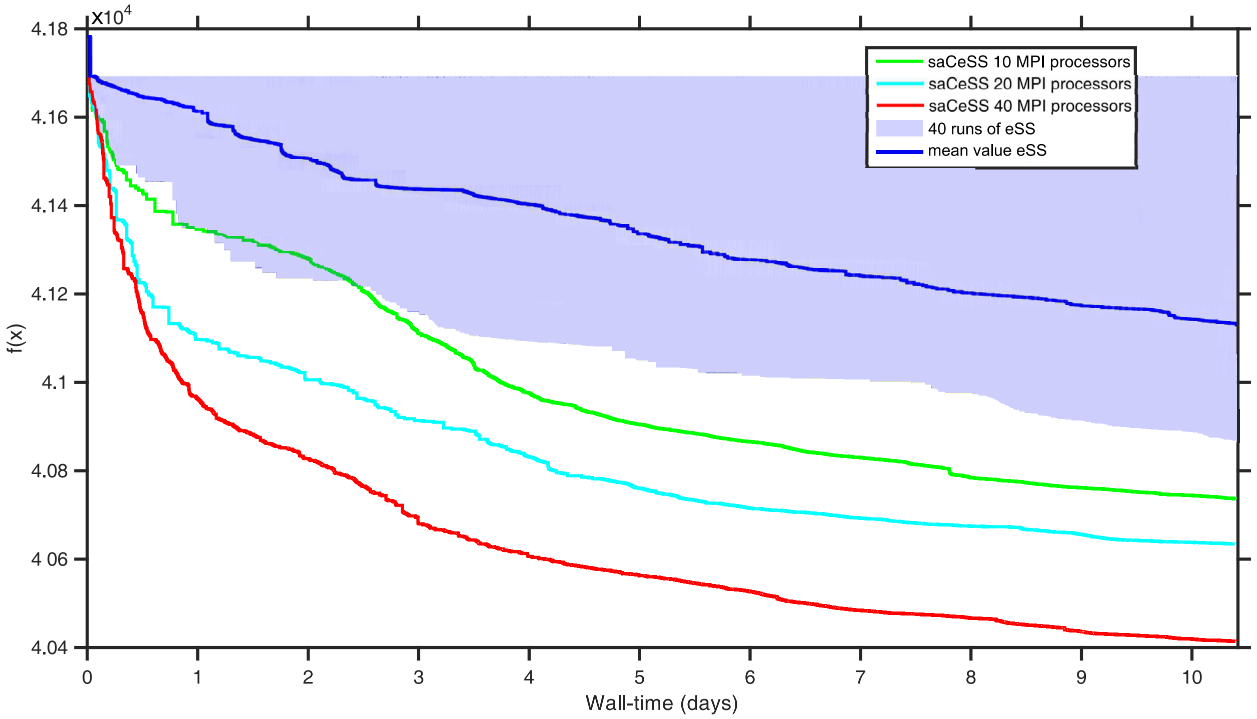


Fig 9. Case study 3: HPN-DREAM convergence curves in the NEMO local cluster. Convergence curves using 10, 20, 40 and 60 cores.

https://doi.org/10.1371/journal.pone.0182186.g009

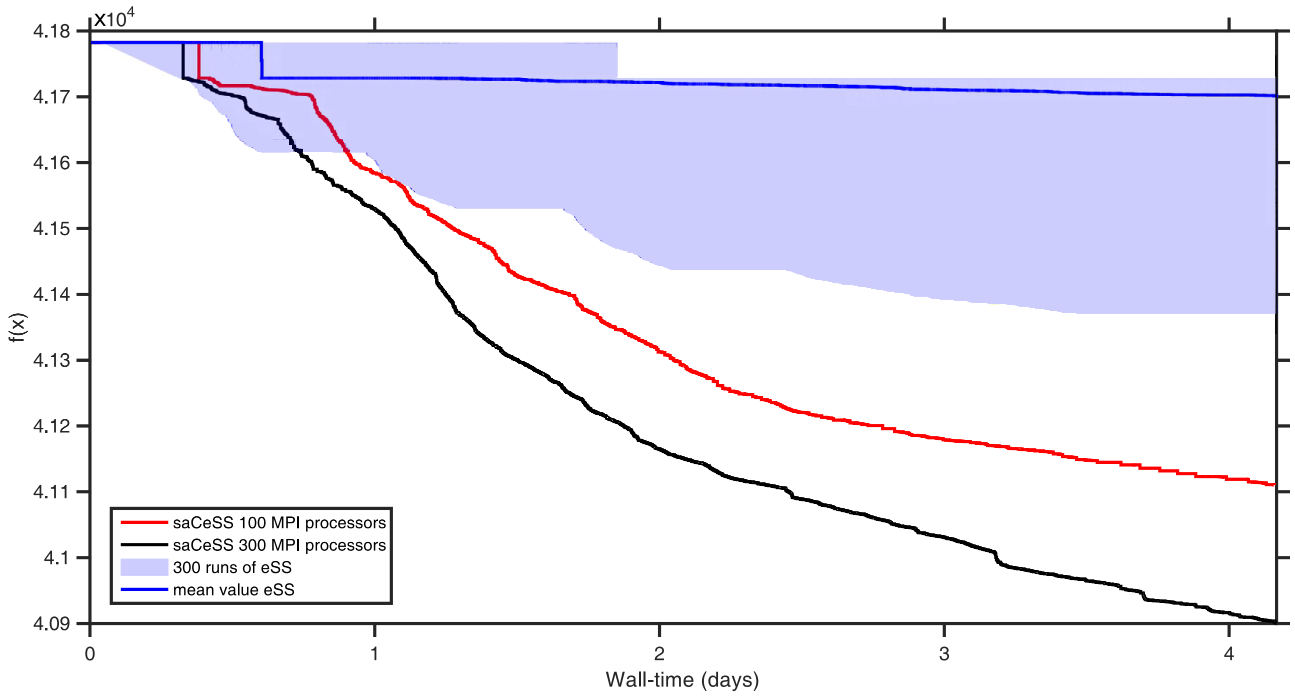


Fig 10. Case study 3: HPN-DREAM convergence curves in the EBI cluster. Convergence curves using 100, and 300 cores.

https://doi.org/10.1371/journal.pone.0182186.g010

the stopping criterion used) had to be set to 4 days. Note that, due to the differences between both infrastructures, it is quite difficult to perform a fair comparison with our local cluster. Although the convergence rate seems to be slower in the EBI cluster, the results obtained still demonstrate the good scalability of saCeSS2. The lower convergence rate in the EBI cluster is due to the architectural and performance differences with respect to our local cluster, and also to the use of GNU compilers instead of the Intel compilers used in our local cluster. Nevertheless, the scalability of saCeSS2 is maintained: the more resources we can use for the cooperative method, the larger improvement we will obtain versus executing the sequential method with the same computational resources.

Performance evaluation of saCeSS2 in the Cloud

As it was already demonstrated in previous subsections, though saCeSS2 clearly outperforms the sequential and the non-cooperative parallel versions of the eSS, it still requires large computational times to achieve convergence in very hard problems. Additionally, it has been shown that the diversity introduced by the increase in the number of islands clearly improves the algorithm convergence rate. However, an increase in the number of islands should be attended by an increase in the number of computational resources (cores), and this is not always practicable.

With the advent of Cloud Computing, effortless access to a large number of distributed resources has become more feasible. However, the scientific computing community has been quite hesitant in using the Cloud, because traditional programming models do not fit well with the new paradigm. In the last decade, several researchers have studied the performance of HPC applications in the cloud environment [93–97]. Most of these studies use classical MPI benchmarks to compare the performance of MPI on public cloud platforms. These works conclude that the lack of high-bandwidth, low-latency networks, as well as the virtualization overhead, has a large effect on the performance of MPI applications on the cloud. In this section we explore the use of a cloud platform, the Microsoft Azure public cloud, for deploying saCeSS2 experiments. The performance was compared to the one obtained in the NEMO local cluster in terms of computational time. Finally, the cost of cloud resources were also analyzed. Thus, this study could be useful for those researchers interested in the performance of traditional parallel metaheuristics in new cloud platforms and its market price.

Some of the previous experiments were deployed in the Microsoft Azure public cloud using clusters with compute-intensive A9 instances (16 cores, 112GB). These instances are designed and optimized for compute-intensive and network-intensive applications. Each A9 instance uses an Intel Xeon E5-2670 @2.6GHz CPUs with 112GB of RAM. Additionally, A9 instances feature a second network interface for remote direct memory access (RDMA) connectivity. This interface allows instances to communicate with each other over an InfiniBand network, operating at QDR rates, boosting the scalability and performance of many MPI applications.

Table 6 shows the performance of the saCeSS2 method for both SSP and HePG2 case studies in the Azure public cloud. As it can be seen the behavior of the algorithm differs slightly from the results obtained in the NEMO local cluster and reported in previous subsections. In particular, results for a small number of processors are better in Azure than in the local cluster. However, the results obtained in the local cluster outperforms the ones in Azure when the number of processors grows. In particular, note that the number of function evaluations required for convergence is larger in the experiments carried out in the local cluster than in the same experiments carried out in Azure when the number of processors is small (10 cores), and it is the opposite for the experiments that use 20 and 40 cores. This can be attributed to the efficiency of the inter-node communications (remember that each Azure instance has 16

Table 6. Performance of saCeSS2 for both SSP and HePG2 case studies in azure public cloud.

problem	#np	mean fbest±std	mean iter±std	mean evals±std	mean time±std(s)	mean price
SSP	10	9.8±0.2	23±8	246256±73545	3153±948	1.99 €
	20	9.8±0.2	21±9	470857±175723	3057±1177	1.93 €
	40	9.8±0.1	31±44	571966±423381	1861±1426	1.18 €
HePG2	10	32.6±0.3	807±782	11790096±10574631	151939±128256	96.10 €
	20	32.0±1.2	305±243	10617112±7403497	87802±58619	55.53 €
	40	32.8±0.3	731±707	17438937±19853336	68214±77442	43.14 €

Stopping criteria: $VTR_{SSP} = 10$ and $VTR_{HePG2} = 33$.

https://doi.org/10.1371/journal.pone.0182186.t006

cores). The higher latency in the inter-node communications in Azure leads to a slow propagation of promising results between islands, that results in a slower convergence.

Besides, it is noteworthy that the dispersion in the distribution of the results is larger for experiments carried out in the Azure public cloud, specially when the number of cores grows. Figs 11 and 12 illustrate with beanplots this fact. Note the logarithmic scale in axis y. As it can be seen, the number of outliers increases with the number of cores in Azure. Notice that this is exactly the opposite behavior than in the local cluster, and it can be explained by the virtualization overhead in Azure and the use of non-dedicated resources in a multi-tenant platform. Hybrid violin/boxplots, provided in Fig C in S1 File, contribute to illustrate this issue.

To conclude this evaluation we have found it interesting to carry out a brief study on the cost of these experiments in the Azure public cloud. Conducting a cost analysis comparing the cost of relying on cloud computing and that of owning an in-house cluster would be of particular interest, although is a very difficult task [98]. The acquisition and operational expenses

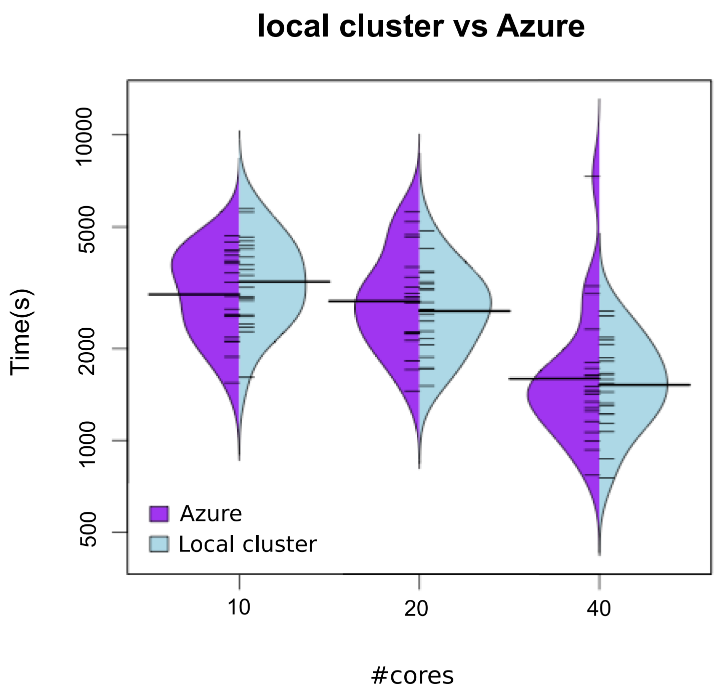


Fig 11. Case study 1: Beanplots comparing results in terms of execution time in azure vs local cluster in the SSP problem.

https://doi.org/10.1371/journal.pone.0182186.g011

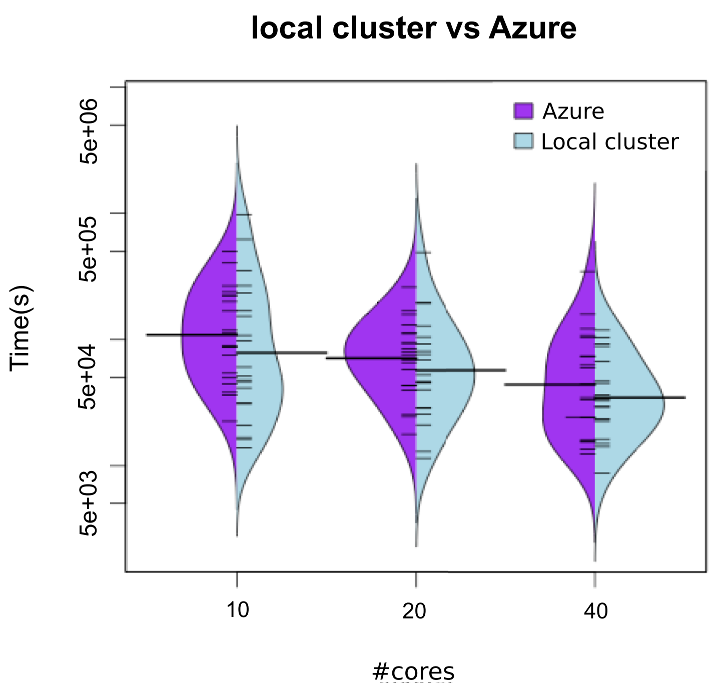


Fig 12. Case study 2: Beanplots comparing results in terms of execution time in azure vs local cluster in the HePG2 problem.

https://doi.org/10.1371/journal.pone.0182186.g012

have to be used in estimating the local clusters' cost. However, the actual cost of local clusters is related to its utilization level. For a local cluster acquired as one unit and maintained for several years, the higher the actual utilization level, the lower the effective cost rate. Besides, labor cost in management and maintenance should also be included, which could be significant. Thus, we found unfeasible an accurate estimation of the cost per hour in our local cluster. Besides, if we take a look to the price of the used instances, we can see that in February 2017 the cost of each A9-instance is 2.2769 EUR/hour. The mean pricing for each experiment is shown in Table 6. In the view of the obtained results we can conclude that, though our experiments in the cloud demonstrates a slightly poorer performance, in terms of execution time, the cloud *pay-as-you-go* model can be potentially a cost-effective and timely solution for the needs of many users.

Conclusions

In this paper, we present a parallel cooperative strategy for the solution of large mixed-integer dynamic optimization problems. This method, saCeSS2, is based on an parallel enhanced scatter search metaheuristic, with new mechanisms and extensions to handle mixed-integer problems. Our strategy shown good performance results when applied to a set of challenging case studies from the domain of computational systems biology. Further, we performed computational runs in different infrastructures (including a local cluster, a large supercomputer and a public cloud platforms) in order to evaluate latency and scalability issues.

This contribution extends the recently developed saCeSS method [41], a parallel cooperative strategy for non-linear programming (NLP) problems, so that it can successfully solve realistic mixed-integer dynamic optimization (MIDO) problems. To this end, the following features have been included in the new saCeSS2 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. In the near future, we plan to generalize saCeSS2 one more level, incorporating additional local MINLP solvers [22], and adopting a hyper-heuristic [99] framework to choose and coordinate them.

The computational results for case studies show that the proposal significantly reduces the execution time needed to obtain a reasonable quality solution. Moreover, the dispersion in the obtained results is narrowed when the number of processors grows. These results confirm that the method can be used to reverse engineer dynamic models of complex biological pathways, and indicates its suitability for other applications based on large-scale mixed-integer optimization, such as metabolic engineering [28], optimal drug scheduling [100, 101] and synthetic biology [102].

Finally, although the approach presented here has been developed taking into account the particular class of logic-based ODE models, it can be applied to any model structure that can be parametrized, i.e. that can be defined by a finite set of structural and dynamic parameters. This direction will be explored in future work.

The code and data files needed to reproduce the results reported here at available at: https://doi.org/10.5281/zenodo.290219.

Supporting information

S1 File. Supplementary info document.
(PDF)

Acknowledgments

This research received financial support from the Spanish Ministerio de Economía y Competitividad (and the FEDER) through projects "SYNBIOFACTORY" (DPI2014-55276-C5-2-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 competitive reference groups GRC2013/055). DRP acknowledges financial support from the MICINN-FPI programme.

Author Contributions

Investigation: David R. Penas, David Henriques, Patricia González, Ramón Doallo, Julio Saez-Rodriguez, Julio R. Banga.

Methodology: Patricia González, Ramón Doallo, Julio Saez-Rodriguez, Julio R. Banga.

Software: David R. Penas, David Henriques.

Supervision: Patricia González, Ramón Doallo, Julio Saez-Rodriguez, Julio R. Banga.

Validation: David R. Penas.

Visualization: David R. Penas, Patricia González.

Writing – original draft: David R. Penas, David Henriques, Patricia González, Ramón Doallo, Julio Saez-Rodriguez, Julio R. Banga.

Writing – review & editing: David R. Penas, David Henriques, Patricia González, Ramón Doallo, Julio Saez-Rodriguez, Julio R. Banga.

References

1. Floudas C. A., Gounaris C. E., A review of recent advances in global optimization, Journal of Global Optimization 45 (1) (2009) 3–38. https://doi.org/10.1007/s10898-008-9332-8
2. Grossmann I. E., Global Optimization in engineering design, Springer Science & Business Media, 2013.
3. Floudas C. A., Pardalos P. M., State of the art in global optimization: computational methods and applications, Springer Science & Business Media, 2013.
4. Horst R., Pardalos P. M., Handbook of global optimization, Springer Science & Business Media, 2013.
5. Mendes P., Kell D., Non-linear optimization of biochemical pathways: applications to metabolic engineering and parameter estimation, Bioinformatics 14 (10) (1998) 869–883. https://doi.org/10.1093/bioinformatics/14.10.869 PMID: 9927716
6. Greenberg H., Hart W., Lancia G., Opportunities for combinatorial optimization in computational biology, INFORMS Journal on Computing 16 (3) (2004) 211–231. https://doi.org/10.1287/ijoc.1040.0073
7. Larrañaga P., Calvo B., Santana R., Bielza C., Galdiano J., Inza I., Lozano J., Armañanzas R., Santafé G., Pérez A., Robles V., Machine learning in bioinformatics, Briefings in Bioinformatics 7 (1) (2006) 86–112. https://doi.org/10.1093/bib/bbk007 PMID: 16761367
8. Banga J. R., Optimization in computational systems biology, BMC Systems Biology 2 (1) (2008) 47. https://doi.org/10.1186/1752-0509-2-47 PMID: 18507829
9. Floudas C. A., Pardalos P. M., Optimization in computational chemistry and molecular biology: local and global approaches, Springer Science & Business Media, 2013.
10. Villaverde A. F., Banga J. R., Reverse engineering and identification in systems biology: strategies, perspectives and challenges, Journal of the Royal Society Interface 11 (91) (2014) 20130505. https://doi.org/10.1098/rsif.2013.0505
11. Chachuat B., Singer A., Barton P., Global methods for dynamic optimization and mixed-integer dynamic optimization, Industrial & Engineering Chemistry Research 45 (25) (2006) 8373–8392. https://doi.org/10.1021/ie0601605
12. Gadkar K. G., Gunawan R., Doyle F. J., Iterative approach to model identification of biological networks, BMC Bioinformatics 6 (1) (2005) 155. https://doi.org/10.1186/1471-2105-6-155 PMID: 15967022
13. Doyle F. J., Stelling J., Systems interface biology, Journal of the Royal Society Interface 3 (10) (2006) 603–616. https://doi.org/10.1098/rsif.2006.0143
14. Kremling A., Saez-Rodriguez J., Systems biology—an engineering perspective, Journal of Biotechnology 129 (2) (2007) 329–351. https://doi.org/10.1016/j.jbiotec.2007.02.009 PMID: 17400319
15. Hasenauer J., Waldherr S., Wagner K., Allgower F., Parameter identification, experimental design and model falsification for biological network models using semidefinite programming, IET Systems Biology 4 (2) (2010) 119–130. https://doi.org/10.1049/iet-syb.2009.0030 PMID: 20232992
16. Jaeger J., Monk N. A., Reverse engineering of gene regulatory networks, Learning and Inference in Computational Systems Biology (2010) 9–34.
17. Kiparissides A., Koutinas M., Kontoravdi C., Mantalaris A., Pistikopoulos E. N., 'closing the loop' in biological systems modeling—from the in silico to the in vitro, Automatica 47 (6) (2011) 1147–1155. https://doi.org/10.1016/j.automatica.2011.01.013
18. Menolascina F., Siciliano V., Di Bernardo D., Engineering and control of biological systems: a new way to tackle complex diseases, FEBS letters 586 (15) (2012) 2122–2128. https://doi.org/10.1016/j.febslet.2012.04.050 PMID: 22580058
19. Sambo F., Montes de Oca M. A., Di Camillo B., Toffolo G., Stutzle T., More: Mixed optimization for reverse engineering—an application to modeling biological networks response via sparse systems of nonlinear differential equations, IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB) 9 (5) (2012) 1459–1471. https://doi.org/10.1109/TCBB.2012.56
20. Guillén-Gosálbez G., Miró A., Alves R., Sorribas A., Jiménez L., Identification of regulatory structure and kinetic parameters of biochemical networks via mixed-integer dynamic optimization, BMC Systems Biology 7 (1) (2013) 113. https://doi.org/10.1186/1752-0509-7-113 PMID: 24176044
21. Sager S., Claeys M., Messine F., Efficient upper and lower bounds for global mixed-integer optimal control, Journal of Global Optimization 61 (4) (2014) 721–743. https://doi.org/10.1007/s10898-014-0156-4
22. Boukouvala F., Misener R., Floudas C. A., Global optimization advances in mixed-integer nonlinear programming, minlp, and constrained derivative-free optimization, cdfo, European Journal of Operational Research 252 (3) (2016) 701–727. https://doi.org/10.1016/j.ejor.2015.12.018

23. Moles C. G., Mendes P., Banga J. R., Parameter estimation in biochemical pathways: a comparison of global optimization methods, Genome Research 13 (11) (2003) 2467–2474. https://doi.org/10.1101/gr.1262503 PMID: 14559783

24. Banga J. R., Balsa-Canto E., Moles C. G., Alonso A. A., Dynamic optimization of bioprocesses: Efficient and robust numerical strategies, Journal of Biotechnology 117 (4) (2005) 407–419. https://doi.org/10.1016/j.jbiotec.2005.02.013 PMID: 15888349

25. Balsa-Canto E., Peifer M., Banga J. R., Timmer J., Fleck C., Hybrid optimization method with general switching strategy for parameter estimation, BMC Systems Biology 2 (1) (2008) 26. https://doi.org/10.1186/1752-0509-2-26 PMID: 18366722

26. Exler O., Antelo L. T., Egea J. A., Alonso A. A., Banga J. R., A tabu search-based algorithm for mixed-integer nonlinear problems and its application to integrated process and control system design, Computers & Chemical Engineering 32 (8) (2008) 1877–1891. https://doi.org/10.1016/j.compchemeng.2007.10.008

27. Schlüter M., Egea J. A., Antelo L. T., Alonso A. A., Banga J. R., An extended ant colony optimization algorithm for integrated process and control system design, Industrial & Engineering Chemistry Research 48 (14) (2009) 6723–6738. https://doi.org/10.1021/ie8016785

28. Sendín J., Exler O., Banga J., Multi-objective mixed integer strategy for the optimisation of biological networks, IET Systems Biology 4 (3) (2010) 236–248. https://doi.org/10.1049/iet-syb.2009.0045 PMID: 20500003

29. Rodriguez-Fernandez M., Egea J. A., Banga J. R., Novel metaheuristic for parameter estimation in nonlinear dynamic biological systems, BMC Bioinformatics 7 (1) (2006) 483. https://doi.org/10.1186/1471-2105-7-483 PMID: 17081289

30. Egea J. A., Balsa-Canto E., García M.-S. G., Banga J. R., Dynamic optimization of nonlinear processes with an enhanced scatter search method, Industrial & Engineering Chemistry Research 48 (9) (2009) 4388–4401. https://doi.org/10.1021/ie801717t

31. Egea J. A., Martí R., Banga J. R., An evolutionary method for complex-process optimization, Computers & Operations Research 37 (2) (2010) 315–324. https://doi.org/10.1016/j.cor.2009.05.003

32. Morris M. K., Saez-Rodriguez J., Sorger P. K., Lauffenburger D. A., Logic-based models for the analysis of cell signaling networks, Biochemistry 49 (15) (2010) 3216–3224. https://doi.org/10.1021/bi902202q PMID: 20225868

33. Henriques D., Rocha M., Saez-Rodriguez J., Banga J. R., Reverse engineering of logic-based differential equation models using a mixed-integer dynamic optimization approach, Bioinformatics 31 (18) (2015) 2999–3007. https://doi.org/10.1093/bioinformatics/btv314 PMID: 26002881

34. Alba E., Parallel Metaheuristics: A New Class of Algorithms, Wiley-Interscience, NJ, USA, 2005.

35. Crainic T. G., Toulouse M., Parallel Strategies for Meta-Heuristics, Springer US, Boston, MA, 2003, pp. 475–513.

36. Jostins L., Jaeger J., Reverse engineering a gene network using an asynchronous parallel evolution strategy, BMC Systems Biology 4 (1) (2010) 17. https://doi.org/10.1186/1752-0509-4-17 PMID: 20196855

37. Perkins T. J., Jaeger J., Reinitz J., Glass L., Reverse engineering the gap gene network of drosophila melanogaster, PLOS Computational Biology 2 (5) (2006) 1–12. https://doi.org/10.1371/journal.pcbi.0020051

38. Goux J.-P., Leyffer S., Solving large minlps on computational grids, Optimization and Engineering 3 (3) (2002) 327–346. https://doi.org/10.1023/A:1021047328089

39. Munawar A., Wahib M., Munetomo M., Akama K., Advanced genetic algorithm to solve minlp problems over gpu, in: 2011 IEEE Congress of Evolutionary Computation (CEC), 2011, pp. 318–325. https://doi.org/10.1109/CEC.2011.5949635

40. Östermark R., Solving difficult mixed integer and disjunctive non-linear problems on single and parallel processors, Applied Soft Computing 24 (2014) 385–405. https://doi.org/10.1016/j.asoc.2014.07.018

41. Penas D., González P., Egea J. A., Doallo R., Banga J., Parameter estimation in large-scale systems biology models: a parallel and self-adaptive cooperative strategy, BMC Bioinformatics 18 (1) (2017) 52. https://doi.org/10.1186/s12859-016-1452-4 PMID: 28109249

42. Mohideen M., Perkins J., Pistikopoulos E., Towards an efficient numerical procedure for mixed integer optimal control, Computers & Chemical Engineering 21 (1997) S457–S462. https://doi.org/10.1016/S0098-1354(97)87544-8

43. Lebiedz D., Sager S., Bock H., Lebiedz P., Annihilation of limit-cycle oscillations by identification of critical perturbing stimuli via mixed-integer optimal control, Physical review letters 95 (10) (2005) 108303. https://doi.org/10.1103/PhysRevLett.95.108303 PMID: 16196975

44. Flores-Tlacuahuac A., Biegler L. T., Simultaneous mixed-integer dynamic optimization for integrated design and control, Computers & Chemical Engineering 31 (5) (2007) 588–600. https://doi.org/10.1016/j.compchemeng.2006.08.010

45. Sager S., A benchmark library of mixed-integer optimal control problems, in: Mixed Integer Nonlinear Programming, Springer, 2012, pp. 631–670.

46. Hirmajer T., Balsa-Canto E., Banga J. R., Dotcvpsb, a software toolbox for dynamic optimization in systems biology, BMC Bioinformatics 10 (1) (2009) 199. https://doi.org/10.1186/1471-2105-10-199 PMID: 19558728

47. Hirmajer T., Balsa-Canto E., Banga J. R., Mixed-integer non-linear optimal control in systems biology and biotechnology: numerical methods and a software toolbox, IFAC Proceedings Volumes 43 (5) (2010) 314–319. https://doi.org/10.3182/20100705-3-BE-2011.00052

48. Sager S., Bock H. G., Diehl M., The integer approximation error in mixed-integer optimal control, Mathematical Programming 133 (1–2) (2012) 1–23. https://doi.org/10.1007/s10107-010-0405-3

49. Otero-Muras I., Banga J. R., Design principles of biological oscillators through optimization: Forward and reverse analysis, PLOS ONE 11 (12) (2016) 1–26. https://doi.org/10.1371/journal.pone.0166867

50. Otero-Muras I., Henriques D., Banga J. R., Synbadm: a tool for optimization-based automated design of synthetic gene circuits, Bioinformatics 32 (21) (2016) 3360–3362. https://doi.org/10.1093/bioinformatics/btw415 PMID: 27402908

51. Bellman R., Dynamic programming and lagrange multipliers, Proceedings of the National Academy of Sciences 42 (10) (1956) 767–769. https://doi.org/10.1073/pnas.42.10.767

52. Bertsekas D. P., Bertsekas D. P., Bertsekas D. P., Bertsekas D. P., Dynamic programming and optimal control, Athena Scientific Belmont, MA, 1995.

53. Bryson A. E., Applied optimal control: optimization, estimation and control, CRC Press, 1975.

54. Liberzon D., Calculus of variations and optimal control theory: a concise introduction, Princeton University Press, 2012.

55. Vassiliadis V., Sargent R., Pantelides C., Solution of a class of multistage dynamic optimization problems. 1. problems without path constraints, Industrial & Engineering Chemistry Research 33 (9) (1994) 2111–2122. https://doi.org/10.1021/ie00033a014

56. Biegler L. T., Cervantes A. M., Wächter A., Advances in simultaneous strategies for dynamic process optimization, Chemical Engineering Science 57 (4) (2002) 575–593. https://doi.org/10.1016/S0009-2509(01)00376-1

57. Schlüter M., Egea J. A., Banga J. R., Extended ant colony optimization for non-convex mixed integer nonlinear programming, Computers & Operations Research 36 (7) (2009) 2217–2229. https://doi.org/10.1016/j.cor.2008.08.015

58. Penas D., Banga J., González P., Doallo R., Enhanced parallel differential evolution algorithm for problems in computational systems biology, Applied Soft Computing 33 (2015) 86–99. https://doi.org/10.1016/j.asoc.2015.04.025

59. Exler O., Schittkowski K., A trust region sqp algorithm for mixed-integer nonlinear programming, Optimization Letters 1 (3) (2007) 269–280. https://doi.org/10.1007/s11590-006-0026-1

60. Hindmarsh A., Brown P., Grant K., Lee S., Serban R., Shumaker D., Woodward C., SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers, ACM Transactions on Mathematical Software 31 (3) (2005) 363–396. https://doi.org/10.1145/1089014.1089020

61. V. S. Vassiliadis, Computational solution of dynamic optimization problems with general differential-algebraic constraints, Ph.D. thesis , University of London, London, U.K. (1993).

62. Glover F., Laguna M., Martí R., Fundamentals of scatter search and path relinking, Control and Cybernetics 29 (3) (2000) 652–684.

63. Exler O., Lehmann T., Schittkowski K., A comparative study of sqp-type algorithms for nonlinear and nonconvex mixed-integer optimization, Mathematical Programming Computation 4 (4) (2012) 383–412. https://doi.org/10.1007/s12532-012-0045-0

64. Aldridge B. B., Burke J. M., Lauffenburger D. A., Sorger P. K., Physicochemical modelling of cell signalling pathways, Nature cell biology 8 (11) (2006) 1195–1203. https://doi.org/10.1038/ncb1497 PMID: 17060902

65. Alberts B., Bray D., Lewis J., Raff M., Roberts K., Watson J. D., Grimstone A., Molecular biology of the cell (3rd edn), Trends in Biochemical Sciences 20 (5) (1995) 210–210. https://doi.org/10.1016/S0968-0004(00)89011-8

66. Jørgensen C., Linding R., Simplistic pathways or complex networks?, Current Opinion in Genetics and Development 20 (1) (2010) 15–22. https://doi.org/10.1016/j.gde.2009.12.003 PMID: 20096559

67. MacNamara A., Henriques D., Saez-Rodriguez J., Modeling Signaling Networks with Different Formalisms: A Preview, Humana Press, 2013, pp. 89–105.

68. Helikar T., Konvalina J., Heidel J., Rogers J. A., Emergent decision-making in biological signal transduction networks, Proceedings of the National Academy of Sciences 105 (6) (2008) 1913–1918. https://doi.org/10.1073/pnas.0705088105

69. Kauffman S. A., Metabolic stability and epigenesis in randomly constructed genetic nets, Journal of Theoretical Biology 22 (3) (1969) 437–467. https://doi.org/10.1016/0022-5193(69)90015-0 PMID: 5803332

70. Wang R.-S., Saadatpour A., Albert R., Boolean modeling in systems biology: an overview of methodology and applications, Physical biology 9 (5) (2012) 055001. https://doi.org/10.1088/1478-3975/9/5/055001 PMID: 23011283

71. Abou-Jaoudé W., Traynard P., Monteiro P. T., Saez-Rodriguez J., Helikar T., Thieffry D., Chaouiya C., Logical modeling and dynamical analysis of cellular networks, Frontiers in Genetics 7 (86) (2016) 94. https://doi.org/10.3389/fgene.2016.00094 PMID: 27303434

72. MacNamara A., Terfve C., Henriques D., Bernabé B. P., Saez-Rodriguez J., State—time spectrum of signal transduction logic models, Physical Biology 9 (4) (2012) 045003. https://doi.org/10.1088/1478-3975/9/4/045003 PMID: 22871648

73. de Jong H., Modeling and simulation of genetic regulatory systems: a literature review, Journal of Computational Biology: a journal of computational molecular cell biology 9 (1) (2002) 67–103. https://doi.org/10.1089/10665270252833208

74. Bonneau R., Reiss D. J., Shannon P., Facciotti M., Hood L., Baliga N. S., Thorsson V., The inferelator: an algorithm for learning parsimonious regulatory networks from systems-biology data sets *de novo*, Genome Biology 7 (2006) R36. https://doi.org/10.1186/gb-2006-7-5-r36 PMID: 16686963

75. Mendoza L., Xenarios I., A method for the generation of standardized qualitative dynamical systems of regulatory networks, Theoretical Biology and Medical Modelling 3 (1) (2006) 13. https://doi.org/10.1186/1742-4682-3-13 PMID: 16542429

76. Aldridge B. B., Saez-Rodriguez J., Muhlich J. L., Sorger P. K., Lauffenburger D. A., Fuzzy logic analysis of kinase pathway crosstalk in TNF/EGF/Insulin-induced signaling, PLOS Computational Biology 5 (4) (2009) 1–13. https://doi.org/10.1371/journal.pcbi.1000340

77. Wittmann D. M., Krumsiek J., Saez-Rodriguez J., Lauffenburger D. A., Klamt S., Theis F. J., Transforming boolean models to continuous models: methodology and application to t-cell receptor signaling, BMC Systems Biology 3 (1) (2009) 98. https://doi.org/10.1186/1752-0509-3-98 PMID: 19785753

78. Bernardo-Faura M., Massen S., Falk C. S., Brady N. R., Eils R., Data-derived modeling characterizes plasticity of mapk signaling in melanoma, PLOS Computational Biology 10 (9) (2014) 1–18. https://doi.org/10.1371/journal.pcbi.1003795

79. Krumsiek J., Poelsterl S., Wittmann D. M., Theis F. J., Odefy—from discrete to continuous models, BMC Bioinformatics 11 (1) (2010) 233. https://doi.org/10.1186/1471-2105-11-233 PMID: 20459647

80. Saez-Rodriguez J., Alexopoulos L. G., Epperlein J., Samaga R., Lauffenburger D. A., Klamt S., Sorger P. K., Discrete logic modelling as a means to link protein signalling networks with functional analysis of mammalian signal transduction, Molecular Systems Biology 5 (2009) 331. https://doi.org/10.1038/msb.2009.87 PMID: 19953085

81. Morris M. K., Saez-Rodriguez J., Clarke D. C., Sorger P. K., Lauffenburger D. A., Training signaling pathway maps to biochemical data with constrained fuzzy logic: quantitative analysis of liver cell responses to inflammatory stimuli, PLOS Computational Biology 7 (3) (2011) 1–20. https://doi.org/10.1371/journal.pcbi.1001099

82. Henriques D., Villaverde A. F., Rocha M., Saez-Rodriguez J., Banga J. R., Data-driven reverse engineering of signaling pathways using ensembles of dynamic models, PLoS computational biology 13 (2) (2017) e1005379. https://doi.org/10.1371/journal.pcbi.1005379 PMID: 28166222

83. Hill S. M., Heiser L. M., Cokelaer T., Unger M., Nesser N. K., Carlin D. E., Zhang Y., Sokolov A., Paull E. O., Wong C. K., et al., Inferring causal molecular networks: empirical assessment through a community-based effort, Nature methods 13 (4) (2016) 310–318. https://doi.org/10.1038/nmeth.3773 PMID: 26901648

84. Burnham K. P., Anderson D. R., Model selection and multimodel inference: a practical information-theoretic approach, Springer Science & Business Media, 2003.

85. Forum M. P., Mpi: A message-passing interface standard, Tech. rep., Knoxville, TN, USA (1994).

86. Website of EMBL-EBI (European bioinformatics institute). URL http://www.ebi.ac.uk

87. Hansen N., Auger A., Finck S., Ros R., Real-parameter black-box optimization benchmarking 2010: Experimental setup, Tech. Rep. Rapports de Recherche RR-6828, Institut National de Recherche en Informatique et en Automatique (INRIA) (2009).

88. Balsa-Canto E., Banga J. R., AMIGO, a toolbox for advanced model identification in systems biology using global optimization, Bioinformatics 27 (16) (2011) 2311–2313. https://doi.org/10.1093/bioinformatics/btr370 PMID: 21685047

89. Alexopoulos L. G., Saez-Rodriguez J., Cosgrove B. D., Lauffenburger D. A., Sorger P. K., Networks inferred from biochemical data reveal profound differences in toll-like receptor and inflammatory signaling between normal and transformed hepatocytes, Molecular & Cell Proteomics 9 (9) (2010) 1849–1865. https://doi.org/10.1074/mcp.M110.000406

90. Terfve C., Cokelaer T., Henriques D., MacNamara A., Goncalves E., Morris M. K., van Iersel M., Lauffenburger D. A., Saez-Rodriguez J., CellNOptR: a flexible toolkit to train protein signaling networks to data using multiple logic formalisms, BMC Systems Biology 6 (1) (2012) 133. https://doi.org/10.1186/1752-0509-6-133 PMID: 23079107

91. Website of DREAM challenges. URL www.dreamchallenges.org

92. Saez-Rodriguez J., Costello J. C., Friend S. H., Kellen M. R., Mangravite L., Meyer P., Norman T., Stolovitzky G., Crowdsourcing biomedical research: leveraging communities as innovation engines., Nature Reviews Genetics 17 (8) (2016) 470–486. https://doi.org/10.1038/nrg.2016.69 PMID: 27418159

93. C. Evangelinos, C. Hill, Cloud computing for parallel scientific HPC applications: Feasibility of running coupled atmosphere-ocean climate models on amazon's EC2, in: 1st Workshop on Cloud Computing and its Applications (CCA'08), 2008, pp. 1–6.

94. J. Ekanayake, G. Fox, High performance parallel computing with clouds and cloud technologies, in: International Conference on Cloud Computing, Springer, 2009, pp. 20–38.

95. J. Napper, P. Bientinesi, Can cloud computing reach the top500?, in: Proceedings of the combined workshops on UnConventional high performance computing workshop plus memory access workshop, ACM, 2009, pp. 17–20.

96. K. R. Jackson, L. Ramakrishnan, K. Muriki, S. Canon, S. Cholia, J. Shalf, H. J. Wasserman, N. J. Wright, Performance analysis of high performance computing applications on the amazon web services cloud, in: Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on, IEEE, 2010, pp. 159–168.

97. Expósito R. R., Taboada G. L., Ramos S., Touriño J., Doallo R., Performance analysis of HPC applications in the cloud, Future Generation Computer Systems 29 (1) (2013) 218–229. https://doi.org/10.1016/j.future.2012.06.009

98. Y. Zhai, M. Liu, J. Zhai, X. Ma, W. Chen, Cloud versus in-house cluster: evaluating amazon cluster compute instances for running mpi applications, in: SC'11: State of the Practice Reports, ACM, 2011, p. 11.

99. Burke E. K., Hyde M., Kendall G., Ochoa G., Özcan E., Woodward J. R., A classification of hyper-heuristic approaches, in: Handbook of Metaheuristics, Springer US, 2010, pp. 449–468.

100. Dua P., Dua V., Pistikopoulos E. N., Optimal delivery of chemotherapeutic agents in cancer, Computers & Chemical Engineering 32 (1) (2008) 99–107. https://doi.org/10.1016/j.compchemeng.2007.07.001

101. Chen C.-L., Tsai H.-W., Model-based insulin therapy scheduling: A mixed-integer nonlinear dynamic optimization approach, Industrial & Engineering Chemistry Research 48 (18) (2009) 8595–8604. https://doi.org/10.1021/ie9005673

102. Otero-Muras I., Banga J. R., Multicriteria global optimization for biocircuit design, BMC Systems Biology 8 (1) (2014) 113. https://doi.org/10.1186/s12918-014-0113-3 PMID: 25248337