Single-objective constrained optimization for Gene Regulatory Networks Modeling

Guillaume Grataloup[1 [0009−0004−9617−4431]], Denis Pallez[1 [0000−0001−5358−8037]], Gilles Bernot[1 [0000−0002−7149−0559]], and Jean-Paul Comet[1 [0000−0002−6681−3501]]

Université Cote d'Azur, CNRS, I3S, Sophia Antipolis, France
firstname.name@univ-cotedazur.fr

Abstract. In the course of apprehending of biological system, the modeling process of gene interactions plays a crucial rocketing role. Unfortunately, the main bottleneck of the modeling process is always the determination of the model parameters. This study focuses on the problem of identifying Gene Regulatory Network (GRN) variables in a discrete framework, represented by gene product concentration thresholds, that separate discrete states of genes contained in the GRN. We propose to compute thresholds from graphs representing interactions between biological genes, gene product concentrations experimentally measured by biologists, and observable behaviors. In this setting, we have developed some adaptations of bio-inspired methods to assist modelers and to propose compatible models to biologists. Since the parameter identification problem is constrained, in this article we focus on adapting and comparing three different heuristics of the CEC'2020 competition for solving single-objective constrained problems using the aforementioned bio-inspired methods by defining a dedicated fitness. To validate our approach, we used an abstract model of the cell cycle and found the performances of the three heuristics consistent with the results of the CEC'2020 competition. This serves as a proof of concept for the development of methods to identify parameters of GRN models using available data and relying less on biological expertise.

Keywords: Single-objective constrained optimization · Biological networks · CEC'2020 competition

1 Introduction

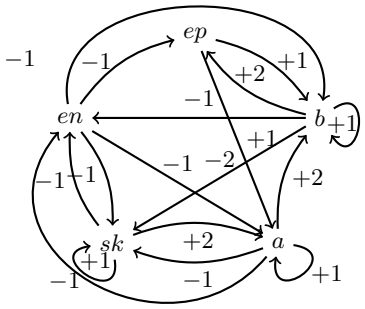
The modeling of biological systems is a fundamental approach to understand complex behaviors at different scales, providing new insights into basic biology, pharmacology and medicine. In particular, GRN is a model of molecular mechanisms that aims to represent interactions between genes and is used as a tool to understand the temporal sequence of system regulations. These regulations are represented by directed graphs where nodes correspond to genes, edges are regulations, and labels indicate whether the regulation is an activation or an inhibition with a + and − label respectively (fig. 1a). The combination of these two basic types of interactions is sufficient to describe the diversity of biological

systems, many of which give rise to complex behavior. In particular, homeostasis (stability properties) can be obtained by a cycle with an odd number of inhibitions, and multistationarity (coexistence of different stable states) can be obtained by a cycle with an even number of inhibitions. Multiple mathematical frameworks and paradigms can be used to apprehend the functioning of GRN. There are four main categories: *discrete* models highlight qualitative nature of the regulations, *stochastic* models consider transitions between states as non-deterministic, *differential equations* use rather precise values of concentrations of gene-associated chemical species, and hybrid models combine aspects of previous categories. Each model has its own strengths and limitations but they all share a main problematic of finding good parameters to accurately match the model dynamics to complex biological behaviors. Previous behaviors are analyzed by biologists through experiments that measure the concentration of gene product of each gene involved in the model (GRN). While a considerable amount of research is done on inference of GRN from expression data [12], here we rather consider the problem of parameter identification on a given GRN. In this work, we focus on a discrete framework where the gene product concentration is discretized into multiple states. Our main issue is to define transition values (*thresholds*) between all states of all genes by analyzing raw data obtained during biological experiments. However, it is subject to constraints, for example the second threshold of a given chemical species cannot be lower than the first. In this paper, we propose to consider threshold identification as a constrained optimization problem. To solve this problem, we decided to use and compare the first three evolutionary algorithms that won the CEC'2020 competition [9] on real world single-objective constrained optimization.

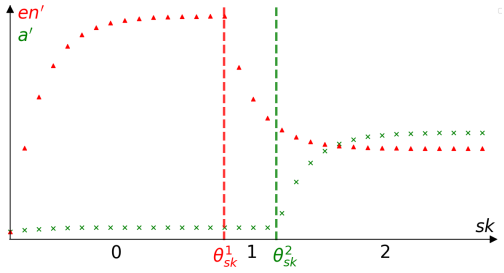
The article is organized as follows: section 2 is devoted to the description of the modeling framework for GRN. Section 3 describes the optimization problem and how constraints are automatically derived from the framework. Three algorithms used to solve this new problem are briefly presented in section 4, focusing on how they handle constraints, and compared through a statistical campaign in section 5. Finally, section 6 concludes.

2 Discrete model for GRN

A GRN is defined as a labeled directed graph $G(V, E)$ where vertices correspond to an abstraction of one or more biological genes, and edges depict activations $(+)$ or inhibitions $(-)$ of gene production. For instance, gene *ep* may activate gene *b* or inhibits gene *a* in the cell cycle GRN showed in fig. 1a. In this study, we consider the R.Thomas' modeling framework [14] which aims at describing qualitatively the dynamics of gene networks by defining discrete states corresponding to regions in the concentration space where all genes are constantly regulated. More precisely, the concentration level of a gene x's product is discretized in several states identified by a number (0, 1, 2 for instance in fig. 1b) such that in state 1 the concentration level is above the concentration level of x's product in state 0 and below state 2. In order to characterize this discretization



(a) Cell cycle GRN model.



(b) Product concentration of genes *en* and *a* exposed to *sk*'s product.

Fig. 1: R. Thomas GRN modeling.

and complete the GRN modeling, one needs to identify values of the transition between states called *thresholds*. The threshold separating states n and $n+1$ of a given gene x is denoted by θ_x^{n+1} in the sequel. R. Thomas formalism also adds discrete values on edges $e \in E$, noted $v_1 \xrightarrow{+n} v_2$ (resp. $v_1 \xrightarrow{-n} v_2$), which means that v_1 activates (resp. inhibits) v_2 only if the concentration of v_1's product has reached at least the n^{th} state concentration level. For instance, when the concentration level of gene *sk* exceeds θ^2_{sk} characterized by the increase of the curve of a (green) in fig. 1b, concentration level of a increases and is represented by an $+2$ edge between *sk* and a, noted $sk \xrightarrow{+2} a$. In this case, the interaction $sk \to a$ is said to be *active*, and *inactive* otherwise.

Although not shown on the graph G, each gene product in the GRN is destined to evolve and will change state over time. The new state towards which a gene will evolve is called the *target* state. The target state of the gene x is entirely determined by the predecessors of that gene in G. In the case of an activation (resp. inhibition), we call *resources* the set of predecessors of x in G, noted ω_x, which have an active (resp. inactive) interaction on x. Moreover, K_{x,ω_x} is the target state of x if its resources are ω_x. Note that there are certain constraints on the values of these parameters: when considering different sets of resources ω_x^1 and ω_x^2 of gene x such as $\omega_x^1 \subset \omega_x^2$, then $K_{x,\omega_x^1} \leq K_{x,\omega_x^2}$. This is due to consideration that the concentration of a gene product will not decrease when gaining resources.

A sequence of gene states in the graph observed by biologists is called a *trace*. It partially describes the dynamics of the system and is formalized based on Hoare's logic [4]. This trace is composed of three elements: a vector of all initial ($(i_1 \cdots i_k)^T \in \mathbb{N}^k$) and final ($(f_1 \cdots f_k)^T \in \mathbb{N}^k$) states of the genes in the GRN and the sequence of state changes and represented by $[v_0 = i_1; \cdots; v_k = i_k]^T (v_2+; v_3-; \cdots; v_1-)[v_0 = f_1; \cdots; v_k = f_k]^T$. A state change is described by the name of the gene changing state and a $+$ (resp. $-$) sign representing the increase (resp. decrease) of the state value by 1.

3 Problem description

Finding the values of the thresholds is the main obstacle to efficiently applying R.Thomas' framework to biological data. To date, the determination of thresholds is done by discussion with biologists and is not automated. Our aim is to propose a way of placing these values with reasonable accuracy using (i) knowledge of the GRN as depicted in fig. 1a, (ii) data representing concentration of gene products over time as shown on fig. 1b named *raw* data and (iii) a trace (cf. section 2) representing an observed interpretation of the raw data by biologists named *observed* trace.

3.1 Modeling problem

To be able to set a threshold automatically, you need some way to derive information from the three pieces of information listed above (GRN, raw data and observed trace). Among the possibilities, one could try to evaluate changes in concentration derivatives. These changes can only be a consequence of a change in resources. For example, suppose that genes sk and ep activate a common target a and that the concentration derivative of a changes sign: this sign change corresponds to a change in the concentration of the product of the predecessor genes of a (sk and ep), which implies a change in state of at least one predecessor of a (sk and/or ep), and thus a threshold crossing for the predecessor(s) that have changed state. So, either θ^2_{ep} or θ^2_{sk} was crossed. As a consequence, either θ^2_{ep} or θ^2_{sk} is close to the value of the concentration of the e or sk product at the time of the sign change of the derivative of the concentration of the product of a. Then, sk and/or ep become or cease to be resources of a. Since the set of resources changes, the target state of a, denoted by K_a, changes. This new target state explains why a's product concentration derivative changes its sign. Although this is a simple example, the actual complexity of GRNs means that the previously explained reasoning cannot be used in practice due to the computational explosion of possible resources that change. However, it is possible to consider a set of thresholds and deduce whether it is consistent with the three available data listed in the introduction paragraph.

Computing a trace from the thresholds. The first step in determining whether the threshold placement is correct is to derive a discrete trace from these thresholds. From these, each concentration measurement in the raw data can be framed between thresholds and thus classified into discrete states for each gene. By following this logic, one can infer all the state changes that occur in sequence, and thus construct a global trace using the raw data, called the *derived* trace. For example, let us consider a very simple case where a inhibits sk in state 1 ($a \xrightarrow{-1} sk$). Let SK and A be the discrete states of sk and a respectively. If the system is in state $[SK = 0; A = 0]$ at a given time and changes states for $[SK = 1; A = 0]$ in the raw data, we can add $(SK+)$ to the derived trace and repeat until all the raw data is considered.

Extracting K parameters from a trace. The discrete states of all genes in the GRN can be derived from a trace. It is then possible to deduce the set of resources for each of the genes and deduce which K parameters are used to describe the target states. For example, if we consider gene a with resources ω_a, then K_{a,ω_a} is the one that describes the target state of a. Furthermore, going from one state to the next allows us to know which K has been used and the sign of the change gives the information whether the K is higher or lower than the considered state. For example, the trace $[SK = 0; A = 0](SK+)[SK = 1; A = 0]$ gave us the information that the target state of sk in the condition where $[SK = 0; A = 0]$ is greater or equal to 1. Since a inhibits sk ($a \xrightarrow{-1} sk$), it is a resource of sk when A is equal to 0 , so $K_{sk,a}$ is greater or equal to 1. By using the same logic for each step in the trace sequence, we can infer multiple K parameter values.

Evaluation of the derived trace. Since the set of thresholds is supposed to be arbitrary, some inconsistencies may occur. There are two ways to detect them: consistency between traces and consistency between θ and K. For the first one, if the derived trace differs from the observed trace, we can exclude the proposed thresholds because they do not give the same biological behavior. The second type of inconsistency arises from the mismatch between the parameters describing the target states (K) extracted from the observed trace, and thoses extracted from the deduced trace. If they are incompatible, there is an inconsistency between the observed trace, the proposed thresholds and the K. The proposed thresholds can be rejected.

Constraints. Regardless of the previous deductions, both the thresholds values and the K parameters are each constrained by nature. Thresholds for a given gene must be ordered in increasing value. For instance θ_a^1 must be inferior to θ_a^2. The K parameters follow the fact that adding resources to a gene cannot decrease the target state. This means that when considering two sets of resources ω_a^1, ω_a^2 of a gene a with one being a subset of the other (ω_x^1 included in ω_x^2 for example) then K_{x,ω_x^1} is lower or equal to K_{x,ω_x^2}.

The total number of thresholds and K parameters can be derived from the GRN. The number of thresholds associated with a given gene is equal to its maximum state on the GRN (the maximum value of the label on an outgoing edge). The number of K parameters for a given gene is equal to 2^n, where n is the number of predecessors. This number of parameters is usually larger than the size of the trace, so reasoning on the trace is limited because the trace does not necessarily go through every possible state of every gene. This means that not all suggested threshold values will be used, and therefore several suggested thresholds may be equivalent. The same reasoning applies to the K parameters, not all possible states are encountered in the trace and therefore not all possible K can be derived. However, it is still useful to consider them as constraints still apply to them.

3.2 Optimization problem

The interest of this work is to identify gene product concentration thresholds in a GRN, given an interaction graph, raw data and a known observable behavior. By considering thresholds and resources as a search space, as discussed in the previous section, this identification can be viewed as a constrained optimization problem thanks to numerous constraints that can be automatically derived. In the following, we expect to use population-based metaheuristics to optimize this problem. We first describe the representation of the individuals before expressing the constraints and the fitness function in detail.

Individuals. The idea here is to define individuals as the concatenations of threshold values and K values in the alphanumeric order. Let $[x_1, x_2, ..., x_n]$ be the list of genes, t_i be the number of thresholds of x_i for $i \in [\![1, n]\!]$, $[r_1^i, r_2^i, ..., r_{j_i}^i]$ be the possible resources (predecessors) of x_i for $i \in [\![1, n]\!]$. The genotype Ψ can be represented as $[\theta_{x_1}^1; \theta_{x_1}^2; ...; \theta_{x_1}^{t_1}; \theta_{x_2}^1; \theta_{x_2}^2; ...; \theta_{x_2}^{t_2}; ...; \theta_{x_n}^{t_n}; K_{x_1,\emptyset}; K_{x_1,r_1^1}; K_{x_1,r_2^1}; ... ; K_{x_1,r_{j_1}^1}; K_{x_1,r_1^1 r_2^1}; ...; K_{x_1,r_1^1 ... r_{j_1}^1}; K_{x_2,\emptyset}; ...; K_{x_n,r_1^n ... r_{j_n}^n}]$. Note that θ are floats, while K are integers.

Constraints. In our problem we have the two types of constraints detailed in section 3.1. Given a gene x, the constraint threshold violation is the sum of the maximum values of θ_x^n minus θ_x^{n+1} and 0, where n is the number of thresholds of x minus 1. Constraint violations of K parameters are the sum of the maximum value between K_{x,ω_x^2} minus K_{x,ω_x^1} and 0, for each set of resources of x, where ω_x^1 is a subset of ω_x^2. An individual's constraint violation score is then calculated as the sum of all previously explained constraint violation scores for each gene in the GRN.

Fitness function. To evaluate the value of a set of thresholds in combination with a set of K parameters, we consider a two-part cost function. First, an individual's thresholds are combined with the raw data to produce a derived trace, as explained in section 3.1. This trace is compared to the observed trace. For each state change in the derived trace, if the changes are different, we calculate how much we should move the proposed threshold of each possible predecessor to make the step identical. We take the smallest change as a value and repeat this for the next step, then take the average of these values as a score. If the sequence of states does not have the same length, we use the same reasoning to find the value of the smallest threshold shift for the missing changes to occur (or the unwanted change to not occur). This calculation will be referred to as f_1. The second part of the fitness, called f_2, checks the coherence between the individual's K parameters and those obtained from the derived trace. If a value is incoherent, the score is increased by one. The fitness function to be minimized is the combination of these two criteria, which are equally weighted.

Finally, threshold identification can be formalized by

$$\begin{aligned} x^* = \underset{x \in \Psi}{\arg\min} \quad & f(x) = 0.5 \times f_1(x) + 0.5 \times f_2(x) \\ \text{subject to} \quad & K_g, \omega_2 - K_g, \omega_1 \leq 0, \quad \forall \omega_1 \subset \omega_2 \in \Omega_g, \quad \forall g \in V, \\ & \theta_g^{i+1} - \theta_g^i \leq 0, \quad \forall i \in [1, t_g - 1], \quad \forall g \in V \end{aligned} \tag{1}$$

where Ω_g is the set of possible resources of gene g and t_g the number of threshold of gene g.

4 Algorithms

To solve the problem presented in the previous section, we consider the first three algorithms of the CEC'2020 competition [9]. A set of 8 algorithms was compared on 57 real problems from different domains (chemistry, mechanics, power electronics... but not bioinformatics), where the number of decision variables varied from 2 to 158, the number of equality constraints from 0 to 148 and the number of inequality constraints from 0 to 91. The algorithms were ranked using a weighted sum of the best, median and mean results obtained on the 57 problems, with higher weights given to higher dimensions. The winner is Self-Adaptive Spherical Search algorithm (SASS) [7] followed by Constrained optimization with Lévy flights Differential Evolution (COLSHADE) [3] and Modified Covariance Matrix Adaptation Evolution Strategy (sCMAgES) [6].

Historically, the most common strategy for dealing with constraints in an optimization problem is to convert the constrained problem into an unconstrained one by adding a penalty term to the objective function that evaluates violations of equality and inequality constraints. Due to numerous limitations of this strategy in defining penalty factors that combine the objective function and the penalty term, several Constraint Handling Techniques (CHT) have been proposed. The most popular CHTs are feasibility rules, self-adaptive penalty function, $\epsilon-$constraint handling, stochastic ranking, combination of several CHTs [10, 11], and more recently boundary updating [2]. The following sections briefly describe each of the 3 algorithms, emphasizing the CHT they use.

4.1 COLSHADE

COLSHADE [3] is based on L-SHADE, a Linear population size reduction and Success History based parameter Adaptation Differential Evolution (DE) algorithm. This means that L-SHADE maintains a historical memory of entries for the control parameters CR and F of the DE algorithm, which are adapted at each generation. The initial population size is set as large as possible to encourage wide exploration and is reduced linearly to speed up convergence and exploit the best solutions found. COLSHADE alternatively combines the levy/1/bin mutation operator, chosen for achieving larger exploration of the search space, with the current-to-$p$best/1/bin mutation operator used by L-SHADE chosen for its exploitation ability.

COLSHADE is adapted to constrained optimization by separating the objective function from the constraints and using the feasibility rules (FR) where no hyperparameters are needed. FR is based on comparisons between pairs of individuals and is therefore well suited to DE-based algorithms when comparing target and trial vectors. In FR, feasible solutions are always considered better than infeasible ones. Two infeasible solutions are compared based on their overall constraint violations only, while two feasible solutions are compared based on their objective function values only. This mechanism aims to push infeasible solutions to the feasible region.

4.2 SASS

Spherical Search (SS) [8] is a recent technique that outperforms the state-of-the-art algorithms for bound-constrained non-convex real-world optimization problems thanks to a smaller number of hyperparameters, a better balance between exploration and exploitation during the search, rotational invariance, and, according to its authors, mapping the contour of the search space. SS has similar characteristics to DE, but differs in the generation of the trial vector by using a spherical boundary on which this vector lies. It also uses a symmetric matrix for the linear transformation from the search space to itself.

SS is adapted to constrained optimization, called SASS [7], by combining the $\epsilon-$constraint (EC) CHT and a gradient-based repair mechanism (RM). In EC, constraints are relaxed under the control of the ϵ parameter, which is adjusted over the evolution process until it reaches 0 at the end of the run. Since solving a constrained optimization problem becomes tedious when many constraints are present, proper control of the ϵ parameter is essential to obtain high quality solutions for problems with equality constraints. The selection of individuals in EC is similar to FR (section 4.1), except that in EC an individual is considered feasible if its total constraint violation is less than ϵ.

The main idea of RM is to use the gradient information derived from the constraint set to systematically repair the infeasible solutions. The gradient information is used to guide the infeasible solutions towards the feasible region. The problem with RM is that it has to be designed for a specific problem, and repairing infeasible solutions can sometimes be as complex as solving the original problem. In SASS, RM is represented as a matrix and is applied with a probability of 0.2 after every D iterations, where D is the number of decision variables.

4.3 sCMAgES

Covariance Matrix Adaptation Evolution Strategies (CMA-ES) is a well-known derivative-free optimization algorithm, mostly dedicated to non-convex continuous search spaces. The algorithm uses only the ranking of individuals to learn the distribution of samples over the fitness landscape. Recently, ϵMAgES has been proposed as a simplified version of CMA-ES for constrained fitness landscapes. sCMAgES [6] improves the time complexity of the sampling scheme and adds

EC and RM strategies to ϵMAgES. In addition, sCMAgES includes a multiple restart scheme to improve performance for multimodal constrained optimization problems. For repairing infeasible individuals in the RM scheme, the same matrix[1] is used for sCMAgES and SASS (section 4.2). This matrix is constructed from a Jacobian matrix of constraints and three other diagonal matrices used to weight the repair process according to the minimum and maximum values of individuals and their constraint values. The repair method used here follows the one in [13].

5 Experiments and Results

In this work, we focus on a qualitative model of the cell cycle which is a series of events leading to correct duplication of DNA of a cell and its division into two genetically identical daughter cells. A 5-variable discrete model of the cell cycle has been designed [1] dedicated for studying the notion of phases in the cell cycle. This GRN (Figure 1a) is composed of 5 genes and has 8 thresholds, 60 K parameters and 222 inequality constraints.
Prior to the application of our approach to the discovery of the underlying thresholds of as yet unknown biological systems, this article is devoted to a proof of concept of our method on simulated raw data: knowing the chosen synthetic parameters, it's easy to assess the performance of our approach. To do so, and keeping in mind that we focus on threshold identification, we systematically generated a differential equation system in accordance with discrete K parameters proposed by D. Boyenval [1], that generate a trace of 11 steps and try to re-identify the chosen thresholds from a numerical solution of the differential equation system.

As mentioned above, we chose algorithms from the CEC competition where the experiments considered 25 independent runs for each algorithm. Note that in the CEC competition, the initial population size differs from each algorithm. SASS starts with a population of 60 individuals, while sCMAgES starts with $4 + \lfloor 3 \times \ln(D) \rfloor$ and COLSHADE with $18 \times D$ where D corresponds to the number of decision variables. COLSHADE's population decreases linearly over the run to 4 while population size does not change for SASS and sCMAgES. The budget allocated to each algorithm was a maximum number of function evaluations (NFE), depending on the number of variables in the problem to be solved (D). For the sake of fairness, we decided to carry out 30 independent runs for the 3 algorithms presented in section 4, with the same initial population size of 100 individuals and a maximal budget of 3×10^5 NFE.

The convergence curve and the percentage of individuals in the population that satisfy the constraints over time are shown in Figure 2. The red horizontal line represents the global optima that need to be identified. We observe that COLSHADE quickly converges to feasible solutions and never considers infeasible solutions afterwards, while SASS and sCMAgES maintain a balance between

[1] available at `https://github.com/GuillaumeGtl/optimization_GNRmodeling`

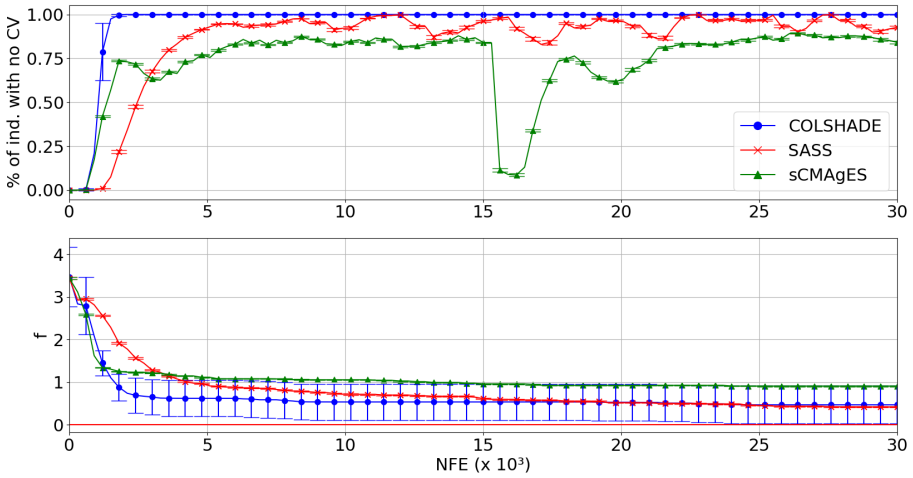


Fig. 2: Average over time of 30 independent runs of the percentage of individuals with no Constraint Violation (top) and the average of the best fitness (bottom). The vertical error bars correspond to the standard deviation.

exploration and exploitation by considering feasible and infeasible individuals throughout the run. It can also be observed that there is a sudden drop in the number of feasible individuals in the population for sCMAgES around 15 000 NFE, probably due to a restart procedure in case of low convergence evolution of the algorithm. This restart procedure discards the current population and initializes a new one, so it is expected that the percentage of individuals satisfying the constraints in this new population will be low. The average best fitness of sCMAgES and COLSHADE seems to stagnate around 0.5. This is explained by the fact that if one of the K parameters of the individual does not correspond to those derived from the trace (see fitness in section 3.2), it makes the second part of the fitness gain a score of 1, thus making the fitness value gain 0.5. In addition, COLSHADE quickly reaches a low fitness value, which is also reached by SASS at a later stage, around 25 000 NFE. The standard devation of COLSHADE also suggest broad differences in results of individual runs. This indicates that COLSHADE tends to stay in local optimum and might benefit from a larger starting population thus encouraging exploration over exploitation in early stage of the optimization. sCMAgES seems to obtain better results in the very beginning but quickly converges towards a local optima. The results of COLSHADE and SASS are very close, which may be due to the fact that they are both based on the DE current-to-$p$best/1/bin algorithm. To distinguish between them, we constructed Cumulative Distribution Function (CDF) curves (fig. 3), which describe the probability of finding a solution at or below a given fitness score for each algorithm. It can be observed that COLSHADE has a probability of 0.7 to obtain a fitness just above 0.5, while SASS has a probability of 0.3 to obtain the

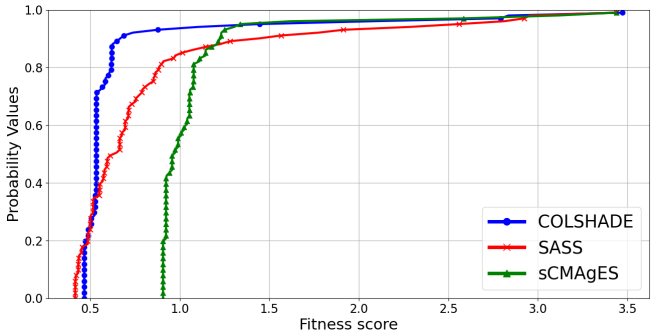


Fig. 3: Cumulative Distribution Function on 30 runs with 3×10^5 NFE.

same fitness. However, SASS algorithm may perform slightly better, but with a probability below 0.2.

Statistical tests. In order to statistically validate our results, we first need to assess whether we can use parametric tests. To do this we need the null hypothesis H_0^1 which states that the data are normally distributed, and H_0^2 which states that the variance are homogeneous. Using the Shapiro-Wilk test, H_0^1 was rejected with a confidence level of $\alpha = 5\%$ for each algorithm, indicating that the 30 runs do not give normally distributed fitness values at 30 000 NFE. Although Levene's test did not reject H_0^2, the rejection of H_0^1 means that we must use nonparametric tests. To conclude on the statistical relevance difference between the observed performance values of the three algorithms, we consider the null hypothesis H_0^3 which states that the performance scores are equal. To test H_0^3 on the three algorithms, we first used Friedman rank-sum test with $\alpha = 5\%$, which rejected the null hypothesis with a p-value of 1.01×10^{-4}. This suggest that the results are significantly different. Section 5 summarizes the results of pairwise comparisons of algorithms using the Wilcoxon test: H_0^3 was rejected between SASS and sCMAgES and between COLSHADE and sCMAgES, but not between SASS and COLSHADE (p-value > 0.05).

This means that there is a significant difference in results between sCMAgES and the other two algorithms, but not between SASS and COLSHADE. Finally, although we only performed 3 pairwise comparisons, we used Bonferroni correction as a post-hoc procedure to avoid false positives. The p-value has been

Fail to reject H0 Reject H0 ($p < 0.05$)

(a)

		SASS
SASS		6.4e-5
COLSHADE	6.3e-1	3.3e-3
	SASS	sCMAgES

(b)

		SASS
SASS		1.9e-4
COLSHADE	1.0e-0	9.9e-3
	SASS	sCMAgES

Table 1: Paired Wilcoxon rank-sum test (a) and Bonferroni post-hoc analysis (b).

changed to adjust for multiple testing (section 5), but the result is the same, i.e. the difference in results between SASS and COLSHADE is not statistically robust, while the results of sCMAgES are considered significantly different. As a result, we can conclude from these tests that SASS and COLSHADE have similar performance, while sCMAgES is considered worse because it converges less than the other two algorithms. The previous results are consistent with those obtained in the CEC competition ranking.

Regarding the thresholds values and "K" parameters found, some are very close to the desired value and others are much further away. This observation is to be expected, since some of the parameters are concerned only with constraints and are very free. Furthermore, as long as a threshold placement proposal generates a trace identical to the expected one, the fitness component that compares the traces (f1) will be equal to 0. This implies that the value that doesn't penalize fitness is only constrained by respecting the order of state changes. So if we consider 3 successive changes of state in the trace, called c1, c2 and c3, if c1 and c3 are very far apart in time, then c2 has a large time interval during which it can take place. Consequently, the threshold associated with c2 can take on any concentration value of the gene product concerned by this change of state between c1 and c3.

6 Conclusion

In the race to understand biological systems, the modeling of regulatory networks plays an important role, as the underlying idea is to reason on the model in order to deduce unobservable functioning of the living system. Unfortunately, the bottleneck of the modeling approach lies in the determination of the parameters. Even in the case of discrete models, this question is limiting, but it also depends on certain fundamental thresholds. In this article, we show that the problem of identifying these thresholds coupled with dynamic parameters can be tackled as an optimization problem by considering the underlying GRN combined with a collection of real concentration measurements and a target trace corresponding to events observed by biologists.

In this context, the problem can be expressed as a constrained optimization problem: some constraints express that thresholds are ordered, and others that the kinetic parameters must satisfy some inequalities. Naturally, we have adapted several single-objective constrained optimization algorithms, selected from among the best of the recent CEC competition, by defining a specific fitness function and repair matrix. We then compared them to our problem of identifying thresholds and kinetic parameters of a 5-gene cell cycle model from which over 200 constraints can be derived. The results show that COLSHADE and SASS are both equivalent approaches and that sCMAgES performs less well. Although the statistical analysis concluded that there was no significant difference between the final results of COLSHADE and SASS, it appears that COLSHADE is more likely to achieve good fitness, although SASS is likely to achieve better fitness, but with a much lower probability.

There are several ways in which this work could be continued. First, it is clear that the problem presented is an optimization problem under constraints with integer and continuous variables. It would therefore be natural to consider specialised mixed-integer algorithms [5]. Secondly, from a modeling point of view, we need to help the modeler to identify the thresholds, but without the information of the observed discrete trace. On the one hand, this information is not necessarily available, but above all, it requires the biologist's expertise on the raw data. It is therefore necessary to modify the fitness function in order to have an evaluation of each individual that requires only the GRN and the raw data.

7 Acknowledgements

This work has been partly funded by Region Sud Provence-Alpes-Côte d'Azur.

References

1. Boyenval, D.: Formal modelling of cyclic biological behaviours with checkpoints: the cell cycle regulation. Phd thesis (2022), `https://theses.hal.science/tel-04032514`
2. Gandomi, A.H., Deb, K.: Implicit constraints handling for efficient search of feasible solutions. Computer Methods in Applied Mechanics and Engineering (2020). https://doi.org/10.1016/j.cma.2020.112917
3. Gurrola-Ramos, J., Hernàndez-Aguirre, A., Dalmau-Cedeño, O.: Colshade for real-world single-objective constrained optimization problems. CEC (2020). https://doi.org/10.1109/CEC48606.2020.9185583
4. Hoare, C.A.R.: An axiomatic basis for computer programming. Commun. ACM (1969). https://doi.org/10.1145/363235.363259
5. Hong, Y., Arnold, D.: Evolutionary mixed-integer optimization with explicit constraints. GECCO'23 (2023). https://doi.org/10.1145/3583131.3590467
6. Kumar, A., Das, S., Zelinka, I.: A modified covariance matrix adaptation evolution strategy for real-world constrained optimization problems. GECCO (2020). https://doi.org/10.1145/3377929.3398185
7. Kumar, A., Das, S., Zelinka, I.: A self-adaptive spherical search algorithm for real-world constrained optimization problems. GECCO (2020). https://doi.org/10.1145/3377929.3398186
8. Kumar, A., Misra, R.K., Singh, D., Mishra, S., Das, S.: The spherical search algorithm for bound-constrained global optimization problems. Appl. Soft Comput. (2019). https://doi.org/10.1016/j.asoc.2019.105734
9. Kumar, A., Wu, G., Ali, M.Z., Mallipeddi, R., Suganthan, P.N., Das, S.: A test-suite of non-convex constrained optimization problems from the real-world and some baseline results. Swarm and Evolutionary Computation (2020). https://doi.org/10.1016/j.swevo.2020.100693
10. Lagaros, N.D., Kournoutos, M., Kallioras, N.A., Nordas, A.N.: Constraint handling techniques for metaheuristics: A state-of-the-art review and new variants (2023). https://doi.org/10.1007/s11081-022-09782-9
11. Rahimi, I., Gandomi, A.H., Chen, F., Mezura-Montes, E.: A review on constraint handling techniques for population-based algorithms: From single-objective to multi-objective optimization (2023). https://doi.org/10.1007/s11831-022-09859-9

12. Spirov, A., Holloway, D.: Using evolutionary computations to understand the design and evolution of gene and cell regulatory networks. Methods (2013). https://doi.org/10.1016/j.ymeth.2013.05.013
13. Takahama, T., Sakai, S.: Constrained optimization by the constrained differential evolution with an archive and gradient-based mutation. In: IEEE Congress on Evolutionary Computation. pp. 1–9 (Jul 2010). https://doi.org/10.1109/CEC.2010.5586484, https://ieeexplore.ieee.org/abstract/document/5586484, iSSN: 1941-0026
14. Thomas, R.: Boolean formalization of genetic control circuits. J.T.B. (1973)