Population-Based Parameter Identification for Dynamical Models of Biological Networks with an Application to Saccharomyces cerevisiae

Authors:

Ewelina Weglarz-Tomczak, Jakub M. Tomczak, Agoston E. Eiben, Stanley Brul

Date Submitted: 2022-01-24

Keywords: dynamic models, evolutionary computing, derivative-free optimization, metabolism, glycolysis, yeast

Abstract:

One of the central elements in systems biology is the interaction between mathematical modeling and measured quantities. Typically, biological phenomena are represented as dynamical systems, and they are further analyzed and comprehended by identifying model parameters using experimental data. However, all model parameters cannot be found by gradient-based optimization methods by fitting the model to the experimental data due to the non-differentiable character of the problem. Here, we present POPI4SB, a Python-based framework for population-based parameter identification of dynamic models in systems biology. The code is built on top of PySCeS that provides an engine to run dynamic simulations. The idea behind the methodology is to provide a set of derivative-free optimization methods that utilize a population of candidate solutions to find a better solution iteratively. Additionally, we propose two surrogate-assisted population-based methods, namely, a combination of a k-nearest-neighbor regressor with the Reversible Differential Evolution and the Evolution of Distribution Algorithm, that speeds up convergence. We present the optimization framework on the example of the well-studied glycolytic pathway in Saccharomyces cerevisiae.

Record Type: Published Article

Submitted To: LAPSE (Living Archive for Process Systems Engineering)

Citation (overall record, always the latest version):	LAPSE:2022.0005
Citation (this specific file, latest version):	LAPSE:2022.0005-1
Citation (this specific file, this version):	LAPSE:2022.0005-1v1

DOI of Published Version: https://doi.org/10.3390/pr9010098

License: Creative Commons Attribution 4.0 International (CC BY 4.0)

processes

MDPI

Article

Population-Based Parameter Identification for Dynamical Models of Biological Networks with an Application to *Saccharomyces cerevisiae*

Ewelina Weglarz-Tomczak [1,*,†], Jakub M. Tomczak [2,*,†], Agoston E. Eiben [2] and Stanley Brul [1]

[1] Swammerdam Institute for Life Sciences, Faculty of Science, University of Amsterdam, 1090 GE Amsterdam, The Netherlands; s.brul@uva.nl
[2] Department of Computer Science, Faculty of Science, Vrije Universiteit Amsterdam, 1081 HV Amsterdam, The Netherlands; a.e.eiben@vu.nl
* Correspondence: ewelina.weglarz.tomczak@gmail.com (E.W.-T.); jmk.tomczak@gmail.com (J.M.T.)
† These authors contributed equally to this work.

Abstract: One of the central elements in systems biology is the interaction between mathematical modeling and measured quantities. Typically, biological phenomena are represented as dynamical systems, and they are further analyzed and comprehended by identifying model parameters using experimental data. However, all model parameters cannot be found by gradient-based optimization methods by fitting the model to the experimental data due to the non-differentiable character of the problem. Here, we present `POPI4SB`, a Python-based framework for population-based parameter identification of dynamic models in systems biology. The code is built on top of `PySCeS` that provides an engine to run dynamic simulations. The idea behind the methodology is to provide a set of derivative-free optimization methods that utilize a population of candidate solutions to find a better solution iteratively. Additionally, we propose two surrogate-assisted population-based methods, namely, a combination of a k-nearest-neighbor regressor with the Reversible Differential Evolution and the Evolution of Distribution Algorithm, that speeds up convergence. We present the optimization framework on the example of the well-studied glycolytic pathway in *Saccharomyces cerevisiae*.

Keywords: dynamic models; evolutionary computing; derivative-free optimization; metabolism; glycolysis; yeast

check for updates

Citation: Weglarz-Tomczak, E.; Tomczak, J.M.; Eiben, A.E.; Brul, S. Population-Based Parameter Identification for Dynamical Models of Biological Networks with an Application to *Saccharomyces cerevisiae*. *Processes* **2021**, *9*, 98. https://doi.org/10.3390/pr9010098

Received: 14 December 2020
Accepted: 30 December 2020
Published: 5 January 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Mathematical models in systems biology are mostly represented by ordinary differential equations (ODEs). They provide a representation of the information obtained from experimental observations about the structure and function of a particular biological network [1,2]. The integral component of ODEs is parameters that correspond to the kinetic characteristics of a reaction catalyzed by a specific enzyme in particular conditions. Typically, the parameters are identified by fitting the model to experimental data or are measured for individual reactions separately. Once parameter values are determined, dynamic models could be used to confirm hypotheses, draw predictions and find such (time-varying) stimulation conditions that result in a particular desired behavior of a system [2–4]. However, the problem of fitting a dynamical model to experimental data is non-differentiable, thus, derivative-free optimization methods should be used instead of gradient-based or higher-order optimizers [5,6].

Here, we present a framework that implements a set of population-based optimization methods to identify parameters in a dynamic model of a biological network of interest, from limited available experimental data. In other words, the presented framework allows finding parameter values of a dynamical model while only selected quantities are observed. This could drastically decrease the time of fitting separate reactions to data and improve

estimation quality because all reactions are considered as a whole, thus, it takes into account interaction among reactions. The implementation of the approach is a stand-alone Python program. It utilizes `PySCeS` (Python Simulator for Cellular System) [7], a modeling tool for formulating dynamical models of biological networks and running simulations by solving ODEs numerically. Our framework loads a model developed using `PySCeS` or from the JWS database [8] together with experimental data, and outputs parameter values for which a difference between the experimental data and the simulation is smallest. Moreover, the framework allows adding new optimizers to a single file, without the necessity of changing any other parts of the program. Please see *Supplementary Data* for details. We refer to this framework as `POPI4SB`, see its schematic representation in Figure 1.

In this study, we chose *glycolysis* that is a crucial metabolic pathway and its upregulation is correlated with diseases like cancer [9,10]. Nearly all living organisms carry out glycolysis as a part of cellular metabolism. One of the most intensively studied organisms in the context of, among others, glycolysis is *Saccharomyces cerevisiae* species, also known as baker's yeast [11–15]. We applied our optimization framework to a model of glycolysis in yeast proposed in [16]. This model contains lumped reactions of the glycolytic pathway and includes production of glycerol, fermentation to ethanol and exchange of acetaldehyde between the cells, and trapping of acetaldehyde by cyanide.

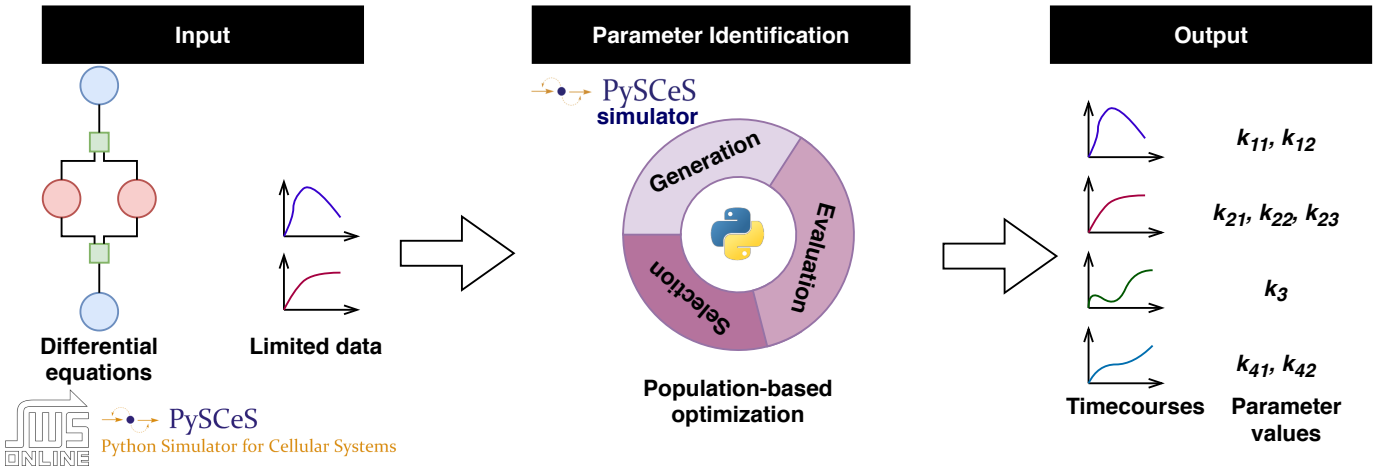

Figure 1. A schematic representation of our framework. A dynamic model in the `PySCeS` format and experimental data are inputs to the program. The core component is the parameter identification with population-based optimization methods. Eventually, parameters values are returned, for with the lowest error (i.e., the difference between simulated data and experimental data) was achieved.

The contribution of the paper is threefold:

- We provide a population-based optimization framework for parameter identification and showcase its performance on the example of the glycolysis of *Saccharomyces cerevisiae*, one of the most studied species in biology.
- We analyze the performance of the population-based optimization framework in the considered problem and indicate its high potential for future research.
- We extend the Python framework `PySCeS` [7] by implementing the population-based optimization methods (four methods known in the literature, and two new methods) in Python. The code for the methods together with the experiments is available online: https://github.com/jmtomczak/popi4sb.

2. Materials and Methods

2.1. Derivative-Free Optimization

We consider an optimization problem of a function $f : \mathbb{X} \to \mathbb{R}$, where $\mathbb{X} \subseteq \mathbb{R}^D$ is the search space. In this paper we focus on the minimization problem, namely:

$$\mathbf{x}^* = \arg\min_{\mathbf{x} \in \mathbb{X}} f(\mathbf{x}; \mathcal{D}), \tag{1}$$

where $\mathcal{D}$ denotes observed data.

Further, we assume that the analytical form of the function f is unknown or cannot be used to calculate derivatives, however, we can query it through a simulation or experimental measurements. Problems of this sort are known as *derivative-free* or *black-box* (In general, a *black-box* problem means that a formal description of a problem is unknown, however, very often non-differentiable problems with known mathematical representation (e.g., differential equations) are treated as black-box) optimization problems [5,17]. Additionally, we consider a bounded search space, i.e., we include inequality constraints for all dimensions in the following form: $l_d \leq x_d \leq u_d$, where $l_d, u_d \in \mathbb{R}$ and $l_d < u_d$, for $d = 1, 2, \ldots, D$.

2.2. Population-Based Optimization Methods

One group of widely-used methods for derivative-free optimization problems is population-based optimization algorithms. The idea behind these methods is to use a *population* of *individuals*, i.e., a collection of candidate solutions $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, instead of a single individual in the iterative manner. The premise of utilizing the population over a single candidate solution is to obtain better exploration of the search space and exploiting potential local optima [18,19].

In the essence, every population-based algorithm consists of three following steps that utilize a procedure for generating new individuals G, and a selection procedure S, that is:

(Init) Initialize $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ and evaluate all individuals $\mathcal{F}_x = \{f_n : f_n = f(x_n), x_n \in \mathcal{X}\}$.
(Generation) Generate new candidate solutions using the current population, $\mathcal{C} = G(\mathcal{X}, \mathcal{F}_x)$.
(Evaluation) Evaluate all candidates solutions:

$$\mathcal{F}_c = \{f_n : f_n = f(x_n),\ x_n \in \mathcal{C}\}.$$

(Selection) Select a new population using the candidate solutions and the old population

$$\mathcal{X} := S(\mathcal{X}, \mathcal{F}_x, \mathcal{C}, \mathcal{F}_c).$$

Go to **Generate** or terminate.

An exemplary population-based optimization approach is depicted in Figure 2.

In general, the population-based optimization methods are favorable over standard derivative-free optimization (DFO) algorithms in problems when querying the objective function is relatively cheap. Their computational complexity depends mainly on the population size, i.e., it is linear with respect to the size of the population N. Other DFO methods are typically more expensive. Bayesian Optimization, for instance, is known to give a good performance, but its complexity typically scales cubically with respect to the number of queries [20]. Here, we take advantage of the very low execution time of running a simulator (the glycolysis model) and propose to use the population-based methods for the parameter identification task.

There are a plethora of population-based DFO algorithms [5,6,18,21,22], however, our goal is to verify whether this approach, in general, could be successfully used in the considered task. Therefore, we decide to choose four instances of a group of methods that are easy-to-use and are proven to work well in practice: evolutionary strategies (ES), differential evolution (DE), estimation of distribution algorithms (EDA), and recently

proposed reversible differential evolution (RevDE). Moreover, we propose to enhance EDA and RevDE with a surrogate model to allow better exploration and speed up calculations.

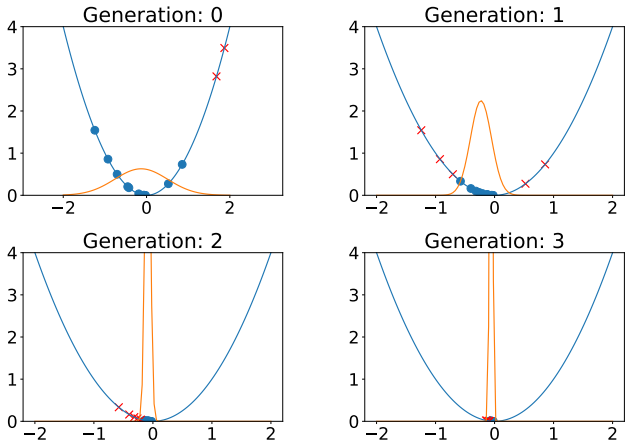

Figure 2. An illustration of a population-based optimization of a quadratic function (blue solid line) using the Estimation of Distribution Algorithm. At each generation a population is selected (blue nodes) and weakest individuals are discarded (red crosses). New candidate solutions are generated by sampling from the normal distribution fit to the previous population (orange solid line).

2.2.1. Evolutionary Strategies (ES)

Evolutionary strategies can be seen as a specialization of evolutionary algorithms with very specific choices of G and S. The core of ES is to formulate G using the multivariate Gaussian distribution. Here, we follow the widely-used (1 + 1)-ES that generates a new candidate using the Gaussian mutation parameterized by $\sigma > 0$, namely:

$$\mathbf{x}' = \mathbf{x} + \sigma \cdot \varepsilon, \tag{2}$$

where $\varepsilon \sim \mathcal{N}(0, \mathrm{I})$, and $\mathcal{N}(0, \mathrm{I})$ denotes the Gaussian distribution with zero mean and the identity covariance matrix I. Next, if the fitness value of $\mathbf{x}'$ is smaller than the value of fitness function of $\mathbf{x}$, the new candidate is accepted and the old one is discarded.

The crucial element of this approach is determining the value of σ. In order to overcome possibly time-consuming hyperparameter search, the following adaptive procedure is proposed [21]:

$$\sigma := \begin{cases} \sigma \cdot c & \text{if } p_s < 1/5, \\ \sigma / c & \text{if } p_s > 1/5, \\ \sigma & \text{if } p_s = 1/5. \end{cases} \tag{3}$$

where p_s is the number of accepted individuals of the offspring divided by the population size N, and c is equal 0.817 following the recommendation in [23].

2.2.2. Differential Evolution (DE)

Differential evolution is another population-based method that is loosely based on the Nelder-Mead method [24,25]. A new candidate is generated by randomly picking a triple from the population, $(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k) \in \mathcal{X}$, and then $\mathbf{x}_i$ is perturbed by adding a scaled difference between $\mathbf{x}_j$ and $\mathbf{x}_k$, that is:

$$\mathbf{y} = \mathbf{x}_i + F(\mathbf{x}_j - \mathbf{x}_k), \tag{4}$$

where $F \in (0, 2]$ is the scaling factor. This operation could be seen as an adaptive *mutation operator* that is widely known as *differential mutation* [25].

Further, the authors of [24] proposed to sample a binary mask $\mathbf{m} \in \{0,1\}^D$ according to the Bernoulli distribution with probability $p = P(m_d = 1)$ shared across all D dimensions, and calculate the final candidate according to the following formula:

$$\mathbf{v} = \mathbf{m} \odot \mathbf{y} + (1 - \mathbf{m}) \odot \mathbf{x}_i, \tag{5}$$

where $\odot$ denotes the element-wise multiplication. In the evolutionary computation literature this operation is known as *uniform crossover operator* [18]. In this paper, we fix $p = 0.9$ following general recommendations in literature [26] and use the uniform crossover in all methods.

The last component of a population-based method is a selection mechanism. There are multiple variants of selection [18], however, here we use the "survival of the fittest" approach, i.e., we combine the old population with the new one and select N candidates with highest fitness values, i.e., the deterministic $(\mu + \lambda)$ selection.

This variant of DE is referred to as "DE/*rand*/*1*/*bin*", where *rand* stands for randomly selecting a base vector, *1* is for adding a single perturbation and *bin* denotes the uniform crossover. Sometimes it is called *classic DE* [25].

2.2.3. Reversible Differential Evolution (RevDE)

The mutation operator in DE perturbs candidates using other individuals in the population to generate a single new candidate. As a result, having too small population could limit exploration of the search space. In order to overcome this issue, a modification of DE was proposed that utilized all three individuals to generate three new points in the following manner [27]:

$$\begin{aligned} \mathbf{y}_1 &= x_i + F(\mathbf{x}_j - \mathbf{x}_k) \\ \mathbf{y}_2 &= x_j + F(\mathbf{x}_k - \mathbf{y}_1) \\ \mathbf{y}_3 &= x_k + F(\mathbf{y}_1 - \mathbf{y}_2). \end{aligned} \tag{6}$$

New candidates $\mathbf{y}_1$ and $\mathbf{y}_2$ could be further used to calculate perturbations using points outside the population. This approach does not follow a typical construction of an EA where only evaluated candidates are mutated. Further, we can express (6) as a linear transformation using matrix notation by introducing matrices as follows:

$$\begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \mathbf{y}_3 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & F & -F \\ -F & 1 - F^2 & F + F^2 \\ F + F^2 & -F + F^2 + F^3 & 1 - 2F^2 - F^3 \end{bmatrix}}_{=\mathbf{R}} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \end{bmatrix}. \tag{7}$$

In order to obtain the matrix $\mathbf{R}$, we need to plug $\mathbf{y}_1$ to the second and third equation in (6), and then $\mathbf{y}_2$ to the last equation in (6). As a result, we obtain $M = 3N$ new candidate solutions. This version of DE is called *Reversible Differential Evolution*, because the linear transformation $\mathbf{R}$ is reversible [27].

2.2.4. Estimation of Distribution Algorithms (EDA)

Most of the population-based optimization methods aim at finding a solution and the information about the distribution of the search space and the fitness function is represented implicitly by the population. However, this distribution could be modeled explicitly using a probabilistic model [19]. These methods have become known as the estimation of distribution algorithms [28–30].

The key difference between EDA and EA is the generation step. While an EA uses evolutionary operators like mutation and cross-over to generate new candidate solutions, EDA fits a probabilistic model to the population, and then new individuals are sampled from this model.

Therefore, fitting a distribution to the population is the crucial part of an EDA. There are various probabilistic models that could be used for this purpose. Here, we propose to fit the multivariate Gaussian distribution $\mathcal{N}(\bar{}, \Sigma)$ to the population $\mathcal{X}$. For this purpose, we can use the empirical mean and the empirical covariance matrix:

$$\hat{\mu} = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n, \tag{8}$$

and

$$\hat{\Sigma} = \frac{1}{N} \sum_{n=1}^{N} (\mathbf{x}_n - \hat{\mu})(\mathbf{x}_n - \hat{\mu})^\top. \tag{9}$$

An efficient manner of sampling new candidates is to first calculate the Cholesky decomposition of the covariance matrix, $\hat{\Sigma} = \mathbf{L}\mathbf{L}^\top$, where $\mathbf{L}$ is the lower-triangular matrix, and then computing:

$$\mathbf{x}' = \hat{\mu} + \mathbf{L}\varepsilon, \tag{10}$$

where $\varepsilon \sim \mathcal{N}(0, \mathrm{I})$. The Equation (10) is repeated M times to generate a new set of candidate solutions. Here, we set M to the size of the population, i.e., $M = N$. Once new candidate solutions are generated, the selection mechanism is applied. In this paper, we use the same selection procedure as the one used for DE.

2.2.5. Population-Based Methods with Surrogate Models (RevDE+ & EDA+)

Surrogate models: A possible drawback of population-based methods is the necessity of evaluating large populations that, even though we assume a low time cost per a single evaluation, could significantly slow down the whole optimization process. To overcome this issue, a surrogate model could be used to partially replace querying the fitness function [31]. The surrogate model is either a probabilistic model or a machine learning model (e.g., a neural network) that gathers previously evaluated populations and allows to mimic the behavior of the fitness function. While applying the surrogate model, it is assumed that its utilization cost (e.g., training) is lower or even significantly lower than the computational cost of running the simulator.

There are multiple possible surrogate models, however, non-parametric models, e.g., Gaussian processes [20], are preferable, because they do not suffer from catastrophic forgetting (i.e., overfitting to the last population and forgetting first populations). Here, we consider another non-parametric model, namely, K-Nearest-Neighbor (K-NN) regression model that stores all previously seen individuals with evaluations, and the prediction of a new candidate solution is an average over K (e.g., $K = 3$) closest previously seen individuals. Current implementations of the K-NN regressor provide efficient search procedures that result in the computational complexity better than $N \cdot D$, e.g., using KD-trees results in $O(D \log N)$. This computational complexity is significantly better than the computational complexity of Gaussian processes, $O(N^3)$.

RevDE+: In the RevDE approach, we generate $3N$ new candidate solutions and all of them are further evaluated. However, this introduces an extra computational cost of running the simulator. This issue could be alleviated by using the K-NN regressor to approximate the fitness values of the new candidates. Further, we can select N most promising points. We refer to this approach as RevDE+.

EDA+: The outlined procedure of EDA produces M new candidate solutions and to keep a similar computational cost as ES and DE, we set M to N. However, this could significantly limit the potential of modeling a search space, because sampling in high-dimensional search spaces requires a significantly large number of points. A potential solution to this problem could be the application of the K-NN regressor to quickly verify the L new points. As long as the time cost of providing the approximated value of the fitness function is lower than the running time of the simulator, we can afford to take $L > N$ (e.g., $L = 5N$). We refer to this approach as EDA+.

2.3. The Model of Glycolysis in Saccharomyces Cerevisiae

Introduction: As an example, we chose *glycolysis* that is a crucial metabolic pathway and its upregulation is correlated with diseases like cancer [9,10]. Nearly all living organisms carry out glycolysis as a part of cellular metabolism. A glycolytic path that consists of a series of reactions breaks down glucose into two three-carbon compounds and extracts energy for cellular metabolism. Therefore, glycolysis is at the heart of classical biochemistry and, as such, it is very well described. One of the most intensively studied organisms in the context of, among others, glycolysis is *Saccharomyces cerevisiae* species, also known as baker's yeast [11–15]. Whereas, the dynamic model of glycolysis in *Saccharomyces cerevisiae* is of big interest in systems biology dynamic modeling literature [16,32–35].

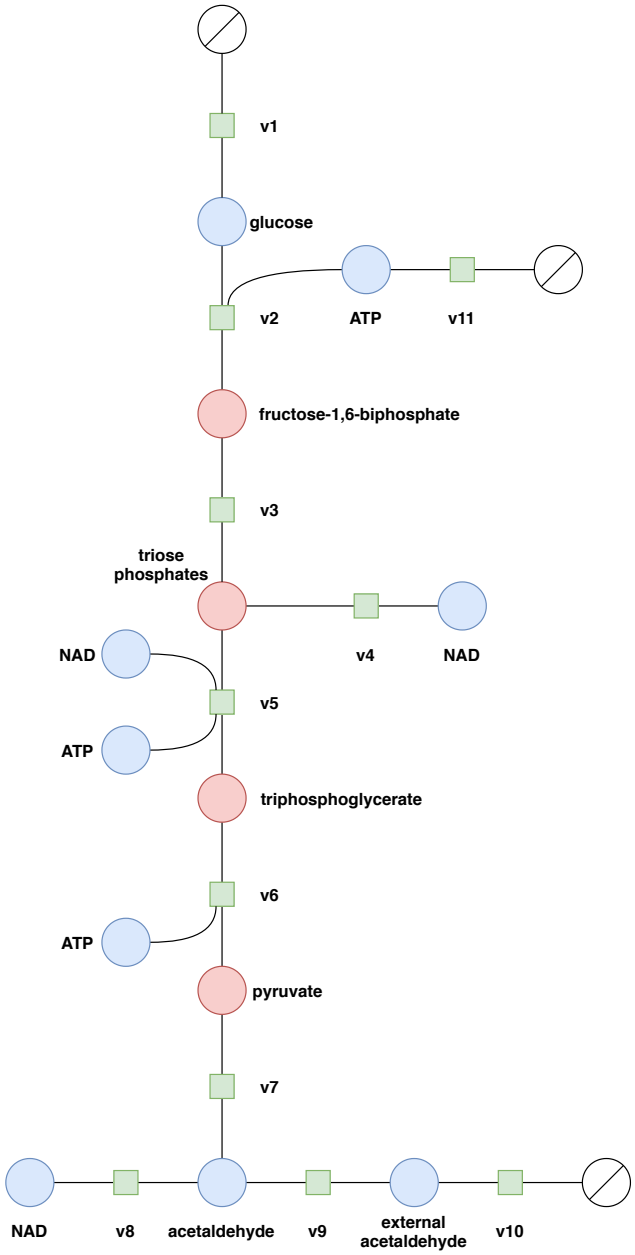

Figure 3. The glycolysis process in the yeast *Saccharomyces cerevisiae* proposed in [16]. There are 11 reactions governing the process with 18 parameters in total, and 9 metabolites. Blue circles depict observable metabolites, red circles denote unobservable metabolites, and green squares represent reactions. A white circle with a diagonal line corresponds to a sink. The model is taken from the JWS database [8].

We applied our optimization framework to a model of glycolysis in yeast proposed in [16], see Figure 3, that suffices to present the essence of our framework. This model contains lumped reactions of the glycolytic pathway and includes the production of glycerol, fermentation to ethanol, and exchange of acetaldehyde between the cells, and trapping of acetaldehyde by cyanide.

A system of Ordinary Differential Equations: In the considered model of the glycolysis we distinguish the following metabolites: glycolysis (*glu*), fructose-1,6-bisphosphate (*fru*), triosephosphates (*triop*), triphosphoglycerate (*tp*), pyruvate (*pyr*), acetaldehyde (*ac*), external acetaldehyde (*ace*).

Following the same assumptions as in [16] (i.e., a homogeneous distribution of the metabolites in the intracellular and in the extracellular solution), the system of ordinary differential equations of the glycolysis model in *Saccharomyces cerevisiae* is the following [36]:

$$\dot{glu} = v_1 - v_2 \tag{11}$$
$$\dot{fru} = v_2 - v_3 \tag{12}$$
$$\dot{triop} = 2v_3 - v_4 - v_5 \tag{13}$$
$$\dot{tp} = v_5 - v_6 \tag{14}$$
$$\dot{pyr} = v_6 - v_7 \tag{15}$$
$$\dot{ac} = v_7 - v_8 - v_9 \tag{16}$$
$$\dot{ace} = 0.1v_9 - v_{10} \tag{17}$$
$$\dot{atp} = -2v_2 + v_5 + v_6 - v_{11} \tag{18}$$
$$\dot{nad} = v_4 - v_5 - v_8 \tag{19}$$

with the rate equations:

$$v_1 = k_0 \tag{20}$$
$$v_2 = \frac{k_1 \cdot glu \cdot at}{1 + (at/k_i)^n} \tag{21}$$
$$v_3 = k_2 \cdot fru \tag{22}$$
$$v_4 = \frac{k_{31} \cdot k_{32} \cdot triop \cdot nadA - k_{33} \cdot k_{34} \cdot tp \cdot atp\ N}{k_{33} \cdot N + k_{32} \cdot A} \tag{23}$$
$$v_5 = k_4 \cdot tp \cdot A \tag{24}$$
$$v_6 = k_5 \cdot pyr \tag{25}$$
$$v_7 = k_6 \cdot ac \cdot nad \tag{26}$$
$$v_8 = k_7 \cdot atp \tag{27}$$
$$v_9 = k_8 \cdot triop \cdot nad \tag{28}$$
$$v_{10} = k_9 \cdot ace \tag{29}$$
$$v_{11} = k_7 \cdot atp \tag{30}$$

where $A = (a_{tot} - atp)$ and $N = (n_{tot} - nad)$.

Initial conditions

The initial conditions are the following:

$$\begin{array}{lll} atp = 2.0 & nad = 0.6 & glu = 5.0 \\ fru = 5.0 & triop = 0.6 & tp = 0.7 \\ pyr = 8.0 & ac = 0.08 & ace = 0.02\ . \end{array}$$

Real parameter values

The real values of the parameters are the following [36]:

$a_{tot} = 4 \in [0, 10]$ $k_0 = 0 \in [0, 10]$ $k_1 = 550 \in [550, 600]$
$k_2 = 9.8 \in [0, 10]$ $k_{31} = 323.8 \in [300, 350]$ $k_{32} = 76411.1 \in [76,400, 76,450]$
$k_{33} = 57823.1 \in [57800, 57850]$ $k_{34} = 23.7 \in [20, 50]$ $k_4 = 80 \in [80, 100]$
$k_5 = 9.7 \in [0, 10]$ $k_6 = 2000 \in [2000, 2050]$ $k_7 = 28.0 \in [20, 50]$
$k_8 = 85.7 \in [80, 100]$ $k_9 = 0 \in [0, 10]$ $k_{10} = 375 \in [350, 400]$
$k_i = 1 \in [0, 10]$ $n = 4 \in [0, 10$ $n_{tot} = 1 \in [0, 10]$

where we indicate the set of possible values of the parameters in the square brackets.

We note that for the sake of our experiments, we set k_0 to 0 (originally: $k_0 = 50$ [36]) in order to forbid a constant injection of glu, and k_9 to 0 (originally: $k_9 = 80$ [36]) in order to avoid oscillatory behavior of the system.

3. Experimental Setup

The experiments have been carried *in silico* in which the performance of the selected algorithms has been evaluated.

3.1. Implementation

`POPI4SB` is implemented in Python, and utilizes `PySCeS` for running simulations. The code for carrying out experiments is available online: https://github.com/jmtomczak/popi4sb. The list of requirements is provided therein.

3.2. Parameter Identification & the Fitness Function

We consider the glycolysis process in yeast as a biochemical system with inputs and outputs (see Figure 3). The input to the system is glucose (*glu*), and the outputs are ATP (*atp*), NAD (*nad*), acetaldehyde (*ac*), and external acetaldehyde (*ace*). The other metabolites, i.e., triose phosphates (*triop*), pyruvate (*pyr*), fructose-1,6-biphosphate (*fru*) and triphosphoglycerate (*tp*) are considered to be unobserved quantities. The system is governed by 11 reactions with 18 parameters in total (see Appendix for details). Each reaction is represented by an ordinary differential equation that is known. We assume that we have inputs and outputs, namely, i.e., *glu*, *atp*, *nad*, *ac*, and *ace*, and each quantity is represented as a timecourse of length T. We denote these measurements by

$$\mathcal{D} = \{glu, atp, nad, ac, ace\}.$$

Further, following the nomenclature presented in [37], we consider the system of differential equations representing the glycolysis process as the **simulator** that for given values of parameters and initial conditions provides timecourses of all metabolites. Then, we can denote parameters by $\mathbf{x}$ and the simulator by $\text{sim} : \mathcal{X} \rightarrow \mathbb{R}^{9 \times T}$, i.e., sim takes parameters $\mathbf{x}$ and simulates timcourses of length T for all 9 metabolites, including glu, atp, nad, ac, ace. In order to calculate the objective (or the fitness) of the parameter values, we use the following function:

$$f(\mathbf{x}; \mathcal{D}) = \sum_{i=1}^{5} \frac{1}{\gamma \cdot T} \sum_{t=1}^{T} \|\mathbf{y}_{i,t} - \text{sim}_{i,t}(\mathbf{x})\|_2^2, \tag{31}$$

where $\mathbf{y}_{i,t}$ corresponds to one of the five observed metabolites at the t-th time step, and $\text{sim}_{i,t}(\mathbf{x})$ is the corresponding synthetically generated signal given by the simulator with parameters $\mathbf{x}$, $\gamma > 0$ specifies the strength of penalizing a mistake. Notice that this is the (unnormalized) logarithm of the product of Gaussian distributions with means given by $\text{sim}(\mathbf{x})$ and the diagonal covariance matrix with shared variance γ.

3.3. Simulated Data

In the experiments, we assume that *glu*, *atp*, *nad*, *ac*, and *ace* are observed. We generate the observed metabolites by running the simulator with the real parameter values. To mimic real measurements that are typically noisy, we add a Gaussian noise with zero mean and the standard deviation equal 3% of a generated value of a metabolite at a given time step.

Adding noise prohibits finding a solution (i.e., values of parameters) that achieves error defined in Equation (31) equal zero. We repeat all experiments three times. For each repetition, we set the length of a timecourse to $T = 30$.

3.4. Settings

For all optimization methods, we set the population size to $N = 100$. All optimizers run maximally 1000 generations. In the case of ES, we use the initial value of σ equal 0.1. For DE, RevDE, and RevDE+, we use $F = 0.5$, and $p = 0.9$. For EDA we take $M = 100$. In the case of EDA+ and RevDE+, we use the K-NN as the surrogate model with $K = 3$, and we do not store more than 10,000 evaluated individuals.

4. Results & Discussion

Fitness value: In Figure 4 we present convergence of the methods in Figure 4. We notice that all methods were able to converge and achieve very similar fitness values. However, the (1 + 1)-ES method was slowest due to the slow exploration capabilities. EDA also required more evaluations to obtain better results. Interestingly, DE, RevDE, RevDE+, and EDA+ achieved almost identical values of the fitness function (the differences were beyond the three-digit precision). An important observation is that application of the surrogate model (the K-NN regressor) allowed to significantly speed up the convergence of RevDE+ and EDA+ compared to RevDE and EDA, respectively. We conclude that all population-based methods were able to converge and achieved almost identical scores, and our proposition of applying the surrogate model led to improving both RevDE and EDA.

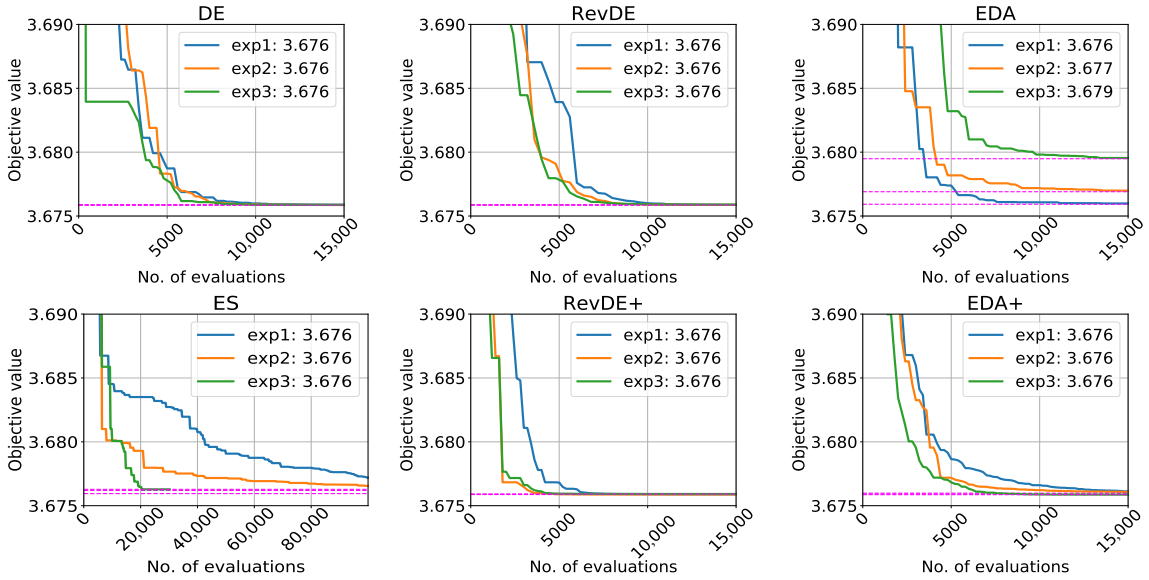

Figure 4. The convergence of the population-based optimization methods over 3 runs. In the legends, we indicate the value of the fitness function after the methods converged.

Timecourses: The final value of the fitness function tells us how well the simulator models the observed timecourses for given parameters provided by an optimizer. Additionally, we can also qualitatively inspect the timecourses both the observed and unobserved metabolites. In Figure 5 we present timecourses for the unobserved metabolites, for parameter values found the five methods.

For all unobserved metabolites, the average over 3 repetitions of the experiments overlapped with the real value or laid within the confidence interval (3× standard deviation). This is a result that we hoped for since being able to generate unobserved metabolite is extremely important for analyzing biological systems. However, we notice that DE and

RevDE+ led to almost identical timecourses, thus, they were able to properly identify parameters.

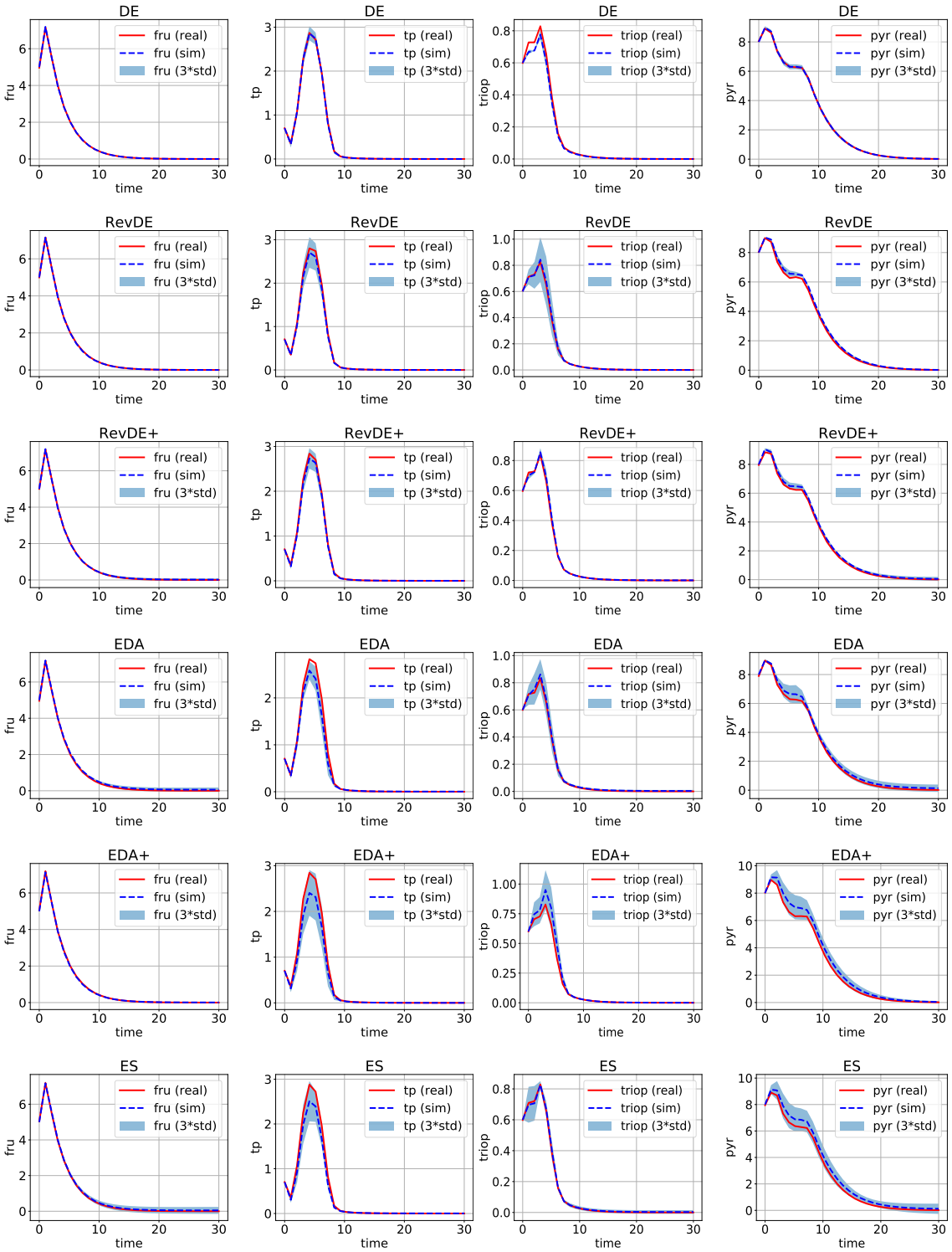

Figure 5. A comparison of the timecourses of the unobserved metabolites. Real timecourses are depicted in red, and the average value and a confidence interval (3× standard deviation) over 3 runs of the simulator is depicted in blue. The titles of the plots indicate optimization methods.

Differences in parameters: In this paper, we know precisely the values of the parameters since they were measured in [16]. Hence, we can compare the parameter values found by the optimization methods with the real parameter values. We use the absolute

value of the difference of two values. We calculate the mean and the standard deviations of the difference from three runs, and use the cumulative distribution function of the folded normal distribution to visualize the distribution of differences (the ideal case is 0). The difference between two real-valued random variable is normally distributed. However, taking the absolute value of a normally distributed random variable results in the folded normal distribution.

In Figure 6 we present difference of all parameters. In general, the differences are marginal and we can conclude that all parameter values were rather properly identified. The biggest problems though appear for parameters that have very large values, e.g., k_8 or k_{33}. This result is very promising because it seems to confirm the promise of the paper that it is possible to identify parameters of a complex biological network for only partially observable metabolites.

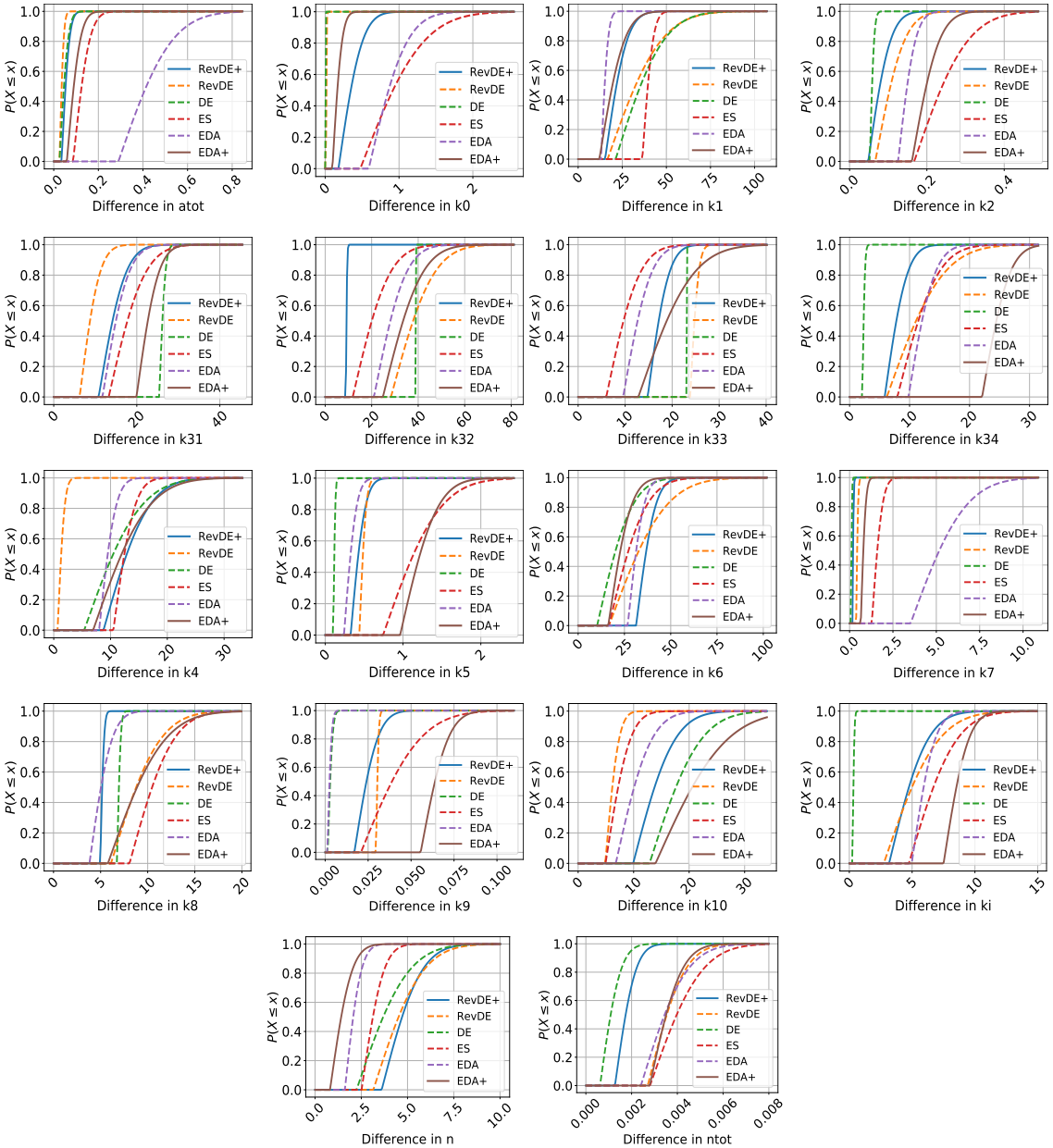

Figure 6. The cumulative distribution functions (cdfs) of the differences for all parameters. Ideally, a cdf of an optimization method should resemble a step-function centered at 0. The averages and the scales are calculated over 3 repetitions of the experiment.

5. Conclusions

In this work, we present a population-based framework for parameter identification of biological networks described as dynamic models. The obtained results indicate the great potential of population-based optimization methods in the field of biology and biochemistry. In the case of relatively low computational costs of obtaining an evaluation of parameters, the population-based methods seem to be sufficient to solve the parameter identification problem. Moreover, our results for applying surrogate models to the optimizers can be highly effective (i.e., speeding up convergence). It is a known fact (e.g., see [31,38]), nevertheless, we believe that the optimization with surrogate models has a great future and should be further investigated. For instance, considering other classes of surrogate models like Gaussian processes or (Bayesian) neural networks opens new opportunities and research questions worth following.

Additionally, the development of our framework in Python, an open-source platform, simplifies its distribution and enables its use on most operating systems. `POPI4SB` is easy-to-use and since the code is freely available, it constitutes a platform for developing new population-based optimizers. Therefore, the proposed framework can be relatively easily extended and serve for future research.

Author Contributions: Conceptualization, E.W.-T.; methodology, E.W.-T. and J.M.T.; software, J.M.T.; validation, E.W.-T. and J.M.T.; formal analysis, E.W.-T. and J.M.T.; investigation, E.W.-T. and J.M.T.; resources, E.W.-T. and J.M.T.; writing—original draft preparation, E.W.-T. and J.M.T.; writing—review and editing, A.E.E. and S.B.; visualization, J.M.T.; supervision, S.B.; project administration, E.W.-T. All authors have read and agreed to the published version of the manuscript.

Funding: EW-T was financed by a grant within Mobilność Plus V from the Polish Ministry of Science and Higher Education (Grant 1639/MOB/V/2017/0).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The code is available at: https://github.com/jmtomczak/popi4sb.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Ingalls, B.P. *Mathematical Modeling in Systems Biology: An Introduction*; MIT Press: Cambridge, MA, USA, 2013.
2. Nielsen, J. Systems biology of metabolism. *Annu. Rev. Biochem.* **2017**, *86*, 245–275. [CrossRef] [PubMed]
3. Ideker, T.; Galitski, T.; Hood, L. A new approach to decoding life: systems biology. *Annu. Rev. Genom. Hum. Genet.* **2001**, *2*, 343–372. [CrossRef] [PubMed]
4. Westerhoff, H.V.; Palsson, B.O. The evolution of molecular biology into systems biology. *Nat. Biotechnol.* **2004**, *22*, 1249–1252. [CrossRef] [PubMed]
5. Audet, C.; Hare, W. *Derivative-Free and Blackbox Optimization*; Springer: Berlin/Heidelberg, Germany, 2017.
6. Larson, J.; Menickelly, M.; Wild, S.M. Derivative-free optimization methods. *arXiv* **2019**, arXiv:1904.11585.
7. Olivier, B.G.; Rohwer, J.M.; Hofmeyr, J.H.S. Modelling cellular systems with PySCeS. *Bioinformatics* **2005**, *21*, 560–561. [CrossRef]
8. Olivier, B.G.; Snoep, J.L. Web-based kinetic modelling using JWS Online. *Bioinformatics* **2004**, *20*, 2143–2144. [CrossRef]
9. Gatenby, R.A.; Gillies, R.J. Why do cancers have high aerobic glycolysis? *Nat. Rev. Cancer* **2004**, *4*, 891–899. [CrossRef]
10. Pelicano, H.; Martin, D.; Xu, R.; Huang, P. Glycolysis inhibition for anticancer treatment. *Oncogene* **2006**, *25*, 4633–4646. [CrossRef]
11. Duarte, N.C.; Herrgård, M.J.; Palsson, B.Ø. Reconstruction and validation of Saccharomyces cerevisiae iND750, a fully compartmentalized genome-scale metabolic model. *Genome Res.* **2004**, *14*, 1298–1309. [CrossRef]
12. Lee, T.I.; Rinaldi, N.J.; Robert, F.; Odom, D.T.; Bar-Joseph, Z.; Gerber, G.K.; Hannett, N.M.; Harbison, C.T.; Thompson, C.M.; Simon, I.; others. Transcriptional regulatory networks in Saccharomyces cerevisiae. *Science* **2002**, *298*, 799–804. [CrossRef]
13. Mensonides, F.I.; Brul, S.; Hellingwerf, K.J.; Bakker, B.M.; Teixeira de Mattos, M.J. A kinetic model of catabolic adaptation and protein reprofiling in Saccharomyces cerevisiae during temperature shifts. *Febs. J.* **2014**, *281*, 825–841. [CrossRef] [PubMed]
14. Nielsen, J. Yeast systems biology: model organism and cell factory. *Biotechnol. J.* **2019**, *14*, 1800421. [CrossRef] [PubMed]
15. Orij, R.; Urbanus, M.L.; Vizeacoumar, F.J.; Giaever, G.; Boone, C.; Nislow, C.; Brul, S.; Smits, G.J. Genome-wide analysis of intracellular pH reveals quantitative control of cell division rate by pH c in Saccharomyces cerevisiae. *Genome Biol.* **2012**, *13*, R80. [CrossRef] [PubMed]

16. Wolf, J.; Passarge, J.; Somsen, O.J.; Snoep, J.L.; Heinrich, R.; Westerhoff, H.V. Transduction of intracellular and intercellular dynamics in yeast glycolytic oscillations. *Biophys. J.* **2000**, *78*, 1145–1153. [CrossRef]
17. Jones, D.R.; Schonlau, M.; Welch, W.J. Efficient global optimization of expensive black-box functions. *J. Glob. Optim.* **1998**, *13*, 455–492. [CrossRef]
18. Eiben, A.E.; Smith, J.E. *Introduction to Evolutionary Computing*; Springer: Berlin/Heidelberg, Germany, 2015; Volume 53.
19. Gallagher, M.; Frean, M. Population-based continuous optimization, probabilistic modelling and mean shift. *Evol. Comput.* **2005**, *13*, 29–42. [CrossRef]
20. Shahriari, B.; Swersky, K.; Wang, Z.; Adams, R.P.; De Freitas, N. Taking the human out of the loop: A review of Bayesian optimization. *Proc. IEEE* **2015**, *104*, 148–175. [CrossRef]
21. Bäck, T.; Foussette, C.; Krause, P. *Contemporary Evolution Strategies*; Springer: Berlin/Heidelberg, Germany, 2013.
22. Moré, J.J.; Wild, S.M. Benchmarking derivative-free optimization algorithms. *SIAM J. Optim.* **2009**, *20*, 172–191. [CrossRef]
23. Schwefel, H.P. *Numerische Optimierung von Computer-Modellen Mittels der Evolutionsstrategie*; Springer: Berlin/Heidelberg, Germany, 1977.
24. Storn, R.; Price, K. Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
25. Price, K.; Storn, R.M.; Lampinen, J.A. *Differential Evolution: A Practical Approach to Global Optimization*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2006.
26. Pedersen, M.E.H. *Good Parameters for Differential Evolution*; Technical Report HL1002; Hvass Laboratories. 2010. Available online: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.298.2174&rep=rep1&type=pdf (accessed on 4 January 2021).
27. Tomczak, J.M.; Weglarz-Tomczak, E.; Eiben, A.E. Differential Evolution with Reversible Linear Transformations. *arXiv* **2020**, arXiv:2002.02869.
28. Larrañaga, P.; Lozano, J.A. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2001.
29. Mühlenbein, H.; Paass, G. From recombination of genes to the estimation of distributions I. Binary parameters. In *International Conference on Parallel Problem Solving from Nature*; Springer: Berlin/Heidelberg, Germany, 1996; pp. 178–187.
30. Pelikan, M.; Hauschild, M.W.; Lobo, F.G. Estimation of distribution algorithms. In *Springer Handbook of Computational Intelligence*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 899–928.
31. Jin, Y. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm Evol. Comput.* **2011**, *1*, 61–70. [CrossRef]
32. Hynne, F.; Danø, S.; Sørensen, P.G. Full-scale model of glycolysis in Saccharomyces cerevisiae. *Biophys. Chem.* **2001**, *94*, 121–163. [CrossRef]
33. Kourdis, P.D.; Goussis, D.A. Glycolysis in saccharomyces cerevisiae: algorithmic exploration of robustness and origin of oscillations. *Math. Biosci.* **2013**, *243*, 190–214. [CrossRef]
34. Teusink, B.; Passarge, J.; Reijenga, C.A.; Esgalhado, E.; Van der Weijden, C.C.; Schepper, M.; Walsh, M.C.; Bakker, B.M.; Van Dam, K.; Westerhoff, H.V.; et al. Can yeast glycolysis be understood in terms of in vitro kinetics of the constituent enzymes? Testing biochemistry. *Eur. J. Biochem.* **2000**, *267*, 5313–5329. [CrossRef]
35. Van Eunen, K.; Bouwman, J.; Daran-Lapujade, P.; Postmus, J.; Canelas, A.B.; Mensonides, F.I.; Orij, R.; Tuzun, I.; Van Den Brink, J.; Smits, G.J.; et al. Measuring enzyme activities under standardized in vivo-like conditions for systems biology. *FEBS J.* **2010**, *277*, 749–760. [CrossRef] [PubMed]
36. Available online: https://jjj.bio.vu.nl/models/wolf/ (accessed on 7 August 2020).
37. Cranmer, K.; Brehmer, J.; Louppe, G. The frontier of simulation-based inference. *Proc. Natl. Acad. Sci. USA* **2020**, *117*, 30055–30062. [CrossRef] [PubMed]
38. Gatopoulos, I.; Lepert, R.; Wiggers, A.; Mariani, G.; Tomczak, J. Evolutionary Algorithm with Non-parametric Surrogate Model for Tensor Program optimization. In Proceedings of the 2020 IEEE Congress on Evolutionary Computation (CEC), Glasgow, UK, 19–24 July 2020.