# Comparison of GA and PSO Performance in Parameter Estimation of Microbial Growth Models: A Case-Study Using Experimental Data

Dulce Calçada, Agostinho Rosa, Luís C. Duarte and Vítor V. Lopes

*Abstract*— **In this work we examined the performance of two evolutionary algorithms, a genetic algorithm (GA) and particle swarm optimization (PSO), in the estimation of the parameters of a model for the growth kinetics of the yeast *Debaryomyces hansenii*. Fitting the model's predictions simultaneously to three replicates of the same experiment, we used the variability among replicates as a criterion to evaluate the optimization result. The performance of the two algorithms was tested using 12 distinct settings for their operating parameters and running each of them 20 times. For the GA, the crossover fraction, crossover function and magnitude of mutation throughout the run of the algorithm were tested; for the PSO, we tested swarms with 3 different types of convergence behavior - convergent with and without oscillations and divergent - and also varied the relative weights of the local and global acceleration constants. The best objective function values were obtained when the PSO fell in the zone of convergence with oscillations or zigzagging, and had a local acceleration larger than the global acceleration.**

## I. Introduction

The mathematical modeling of microbial growth kinetics [1] is a subject of the uttermost importance, both as an exercise in basic microbiology and in terms of its potential industrial applications, as a method for improving the yields of biotechnological processes at large, for example in the context of biorefineries.

The large number of parameters and the nonlinearity of the differential equations describing these models pose serious challenges when it comes to fitting them to experimental data. The resulting nonlinear optimization problem must be solved iteratively, and the employment of classical nonlinear optimization techniques, such as the Levenberg-Marquardt algorithm [2], has limited success, especially when there are many parameters, the search space for each of them is vast, and there isn't a good initial estimate for their values. A much valuable alternative approach for solving this problem is to use heuristics or metaheuristics to perform the optimization, as opposed to fully deterministic algorithms [3][4].

Among the many metaheuristic methods available for solving complex optimization problems such as this, are the two evolutionary algorithms examined in this work: the genetic algorithm (GA) and particle swarm optimization (PSO). They are said to be population-based or ensemble search methods because they operate on a pool of candidate solutions for the problem, instead of on a single solution, and iteratively update it through the application of a number of heuristics.

In recent years, both the GA and the PSO, as well as hybrid algorithms combining the two, have been applied to parameter estimation problems, in the context of kinetic models of microorganisms and plants [5][6][7], metabolic models [8], models of chemical reactions [9][10], and models of biological reactors [11]. A fairly recent and exhaustive is that by Drager *et al* [8], in which the performance of eight different optimization strategies (including the GA and the PSO, as well as other metaheuristics) was compared when estimating the parameters of complex metabolic models for *C. glutamicum*. Mathematically, our model closely resembles some of the ones studied by Drager, highly non-linear and with a comparable number of parameters; however, the phenomena modeled in [8] refers to a well-characterized metabolic pathway, one for which all relevant chemical reactions are known. On the other hand, ours is more of a physiological model, in which we model the overall macroscopic effects deriving from the myriad of microscopic phenomena occuring inside cells. Such a model has the advantage of requiring only measurements that are relatively simple and cheap, as compared to the more sophisticated techniques used for single-cell measurements. Furthermore, the fact that there are three replicates of an experiment carried out under the same experimental conditions will allow us to:

1) optimize for a more representative behavior of the organism, thus avoiding over-fitting of the model to a very specific set of experimental data, and
2) compare the value of the objective function at the end of the optimization to the variability of the experimental data, which is a more natural criterion with which to evaluate the optimization result.

In summary, what we propose and examine in this work is a strategy for selecting the most appropriate metaheuristic, out of two widely used evolutionary algorithms, for estimating the parameters of a mathematically complex physiological model of the yeast *Debaryomyces hansenii* (*D. hansenii*). The remainder of the paper is organized as follows:

- In section II, Preliminaries, background information on the problem is provided, including the experimental data used, the mathematical formulation of the model, the definition of the optimization problem, and a brief summary of the optimization algorithms and the imple-

Dulce Calçada (email: dcalcada@laseeb.org) and Agostinho Rosa (email: acrosa@laseeb.org) are with Instituto de Sistemas e Robótica (LaSEEB-ISR), Instituto Superior Técnico, Av. Rovisco Pais 1, 1049-001 Lisboa, Portugal.

Luís C. Duarte (email: luis.duarte@lneg.pt) and Vítor V. Lopes (email: vitor.lopes@lneg.pt) are with Laboratório Nacional de Energia e Geologia (LNEG), Estrada do Paço do Lumiar 22, 1649-038 Lisboa, Portugal.

mentation used.

- In section III, Strategy for GA and PSO Performance, the methodology used to compare the performance of the two algorithms is described.
- In section IV, Results and Discussion, the results of the simulations are presented and discussed.
- Finally, section V, Conclusions, summarizes the main achievements of this work.

## II. Preliminaries

### A. Experimental Settings and Data Collected

The experimental data used in this work consists of fermentation profiles of the yeast *D. hansenii*, obtained by Duarte et al. [12]. The substrates provided to the yeast were glucose, xylose and arabinose, and the cultures were carried out aerobically in batch mode for a total of 168 h. Samples were taken at ten different time instants and the concentrations of the substrates, extracellular metabolic products (ethanol, glycerol, xylitol and arabitol) and biomass dry weight were measured. The data set we used is composed of three experimental replicates of the reference assay (that is, the experiments labeled B1E1, B2E1 and B2E2 in [12]).

### B. The Kinetic Model

It is beyond the scope of this work to discuss the adequacy of a given model for *D. hansenii* growth kinetics. Previous work, done by one of the authors on her Master's thesis [13], explored nine different models for *D. hansenii* physiology, all biologically reasonable, but each implementing a number of plausible approximations, seeking to reduce the complexity of the problem. We chose for this work the model from which all others were subsequently derived and show on Fig. 1 the network of reactions (arrows) and inhibitions (dash-dotted lines) that our model considers.

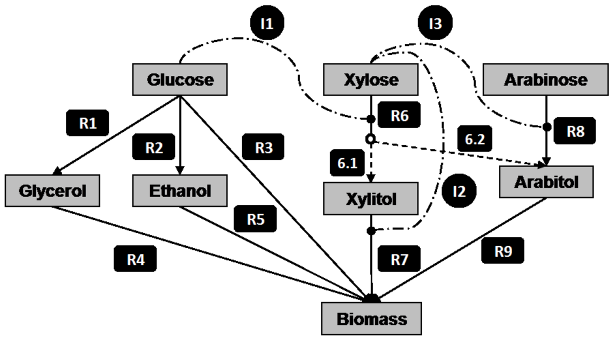

Fig. 1: Graphical representation of the kinetic model.

This network is mathematically formulated as a system of ordinary differential equations (ODEs), describing the rate of change of the concentration of all eight species in the model. Each rate is written as a sum of negative and positive terms, accounting for the reactions in which the species participates, respectively, as a substrate or a product. More specifically, for a species $S_m$ , $m = 1, \dots, 8$, we have the following ODE:

$$\frac{dS_m}{dt} = \left( -\sum_i \mu_i + \sum_j \sum_k Y_{j,k}\,\mu_j \right) X$$
$$i \in C, \quad j \in F, \quad k = 1, \dots, n$$

where $\mu_i = \mu_i(t)$ is the specific rate of reaction $i$ at time instant $t$, $X = X(t)$ is biomass concentration, $Y_{j,k}$ is the yield of branch $k$ of reaction $j$, $C$ is the set of all the reactions in which $S_m$ is consumed, and $F$ is the set of all the reactions in which $S_m$ is formed. If a branched reaction $j$ has $n$ branches, it means that $n$ products are formed from one substrate, all at the same specific rate $\mu_j$, but with different yields, $Y_{j,k}$ , $k = 1, \dots, n$. In our model this happens only in reaction 6, which branches into 6.1 and 6.2; reactions 1, 2 and 3, on the other hand, represent the formation of three products form the same substrate, but at different specific rates, $\mu_1$, $\mu_2$ and $\mu_3$, as well as yields. For non-branched reactions, i.e. if $n = 1$, $Y_{j,1}$ is simply written $Y_j$.

The specific rate $\mu_i$ of a reaction $i$ in which substrate $S$ is consumed has the hyperbolic form of the classic Monod equation:

$$\mu_i = \mu_{max,i} \frac{S}{S + K_S} \tag{1}$$

where $S = S(t)$ is substrate concentration, $\mu_{max,i}$ is the maximum specific rate of the reaction, and $K_S$ is its half-saturation constant, whose inverse is a measure of the affinity of the microorganism for substrate $S$. Given the network on Fig. 1 we built the system of ODEs for this model following the steps presented next:

i) For each of the reactions $R_j$ in the model, $j = 1, \dots, 8$, a specific rate, $\mu_j$, with the hyperbolic form of the Monod equation (1), was defined:

$$\mu_j = \mu_{max,j} \frac{S}{S + K_{S,j}}$$

ii) If $R_j$ had only one product (solid arrows in Fig. 1), then a single yield parameter, $Y_j$, corresponding to the ratio between the amount of product formed and the amount of substrate consumed, was defined:

$$Y_j = \left| \frac{dP}{dS} \right|_{R_j},$$

whereas if $R_j$ had $n > 1$ products (as *e.g.* reaction $R_5$ in Fig. 1), then $n$ yield parameters, $Y_{j,k}$ , $k = 1, \dots, n$, were defined:

$$Y_{j,k} = \left| \frac{dP_k}{dS} \right|_{R_j}$$

iii) If $R_j$ was inhibited by a particular species, $I$, then its specific rate became:

$$\mu_j = \mu_{max,j} \frac{S}{S + K_{S,j}} \frac{K_I}{K_I + I}$$

where $I$ is the concentration of the inhibitor and parameter $K_I$ quantifies the strength of the inhibition.

The resulting system of ODEs has a total of 8 equations and 31 parameters (9 maximum specific rates, 9 half-saturation constants, 10 yield coefficients and 3 inhibition constants), and describes the rate of change of the concentrations of each species throughout the duration of the experiment.

## C. Parameter Estimation as an Optimization Problem

The objective function for this problem, whose value is to be minimized by the optimization algorithms, is given by:

$$F_O(\hat{\theta}) = \sum_{i=1}^{10}\sum_{j=1}^{8}\sum_{k=1}^{3}\Big(y_{ijk} - f\left(t_i, y_{ijk}, \ldots, \hat{\theta}\right)\Big)^2 + \\ + \ \Phi \ + \ \Psi, \qquad \hat{\theta} \in D \tag{2}$$

where $y_{ijk}$ is the experimental concentration of species $k$ at time instant $j$, in replicate $i$, and $f$ is the corresponding model prediction, obtained via numerical integration of the models's system of ODEs, using estimate $\hat{\theta}$ for the vector of parameters $\theta$, and a set of initial conditions given by the initial concentrations in replicate $i$. $D$ is the range of allowed values for the parameters, establishing the bounds of the problem's search domain. All the parameters are required to be non-negative but, in order to avoid divisions by zeros we instead set their lower bounds to a value $10^{-7}$; the upper bounds were set to biological plausible values, by literature inspection [12][14] and, in the case of some of the yield coefficients, stoichiometric restrictions.

The terms $\Phi$ and $\Psi$ in equation (2) are penalizations for the violation of the problem's constraints. In particular, $\Phi$ penalizes the prediction of negative concentrations, according to:

$$\Phi = \sum_{i=1}^{10}\sum_{j=1}^{8}\sum_{k=1}^{3}\phi(y_{ijk})$$

$$\text{with} \quad \phi(y_{ijk}) = \begin{cases} 10^6 \times y_{ijk}^2 & \text{if } y_{ijk} < 0, \\ 0 & \text{if } y_{ijk} \geq 0. \end{cases}$$

In this way, the penalization for negative concentrations is proportional to the square of the value of the concentration, meaning that small negative concentrations are acceptable, since they could be the product of integration errors, but larger ones are not allowed. As for the $\Psi$ term, it penalizes the violation of the linear constraint enforcing that the sum of the yield coefficients of the two branches of reaction 6, $Y_{6,1}$ and $Y_{6,2}$, must not be higher than the maximum of the two upper bounds of $Y_{6,1}$ and $Y_{6,2}$. When this constraint is violated $\Psi = 10^6$, otherwise $\Psi = 0$. The factor of $10^6$ in the penalization terms was chosen so that a violation of either of the corresponding constraints would yield objective function values more than three orders of magnitude higher than those considered acceptable (see the discussion in the next section).

The integration of the system of ODEs was accomplished with a numerical solver employing a variable step size (`ode15s`[15], available in MATLAB®), so as to effectively deal with the stiffness of the problem.

## D. Analysis of Experimental Variability

The goal of performing an analysis of experimental variability is to establish a criterion against which to compare the result of the optimization, that is, the value of the objective function found at the last iteration by the optimization algorithms. Such a criterion is especially relevant in this problem due to the high variability characteristic of biological systems. Being a measure of experimental data variability, this criterion should also be comparable to the measure being minimized during the optimization, that is, to the output of equation (2). As such, we decided to make this criterion equal to the sum of distinct squared pairwise differences [16] between corresponding times and species in the $N = 3$ experimental replicates in the data set, henceforth designated $SSPD$:

$$SSPD = \sum_{i=1}^{10}\sum_{j=1}^{8}\sum_{k=1}^{N=3}\sum_{\substack{l=2 \\ l\neq k}}^{N=3}\Big(x_{ijk} - x_{ijl}\Big)^2 \tag{3}$$

where $x_{ijk}$ is the concentration measured, at time instant $i$, for species $j$, in replicate $k$.

The $SSPD$ provides an estimate of the variability of a data set and therefore constitutes a valuable measure of comparison of the optimization's results. However, it should be more informative to obtain a distribution for the $SSPD$, showing its own expected variability. This should be more accurate than evaluating objective function values in terms of their relative deviation from the single $SSPD$ value obtained using equation (3), since without variability information a small percentage deviation may correspond to a large absolute one, and vice-versa. Also, since both of the optimization algorithms are stochastic, and will thus arrive at different solutions in different runs, it would be possible to compare the overall variability of the objective function values with the varibility of the experimental $SSPD$.

Typically, in order to do so, one would have to obtain many comparable sets of replicates and compute their corresponding $SSPD$, obtaining its distribution. Since additional sets are not available, this distribution was computed, in a very simple and reasonably accurate fashion, by using a technique called bootstrapping [17].

Bootstrapping is a non-parametric, computer-intensive, resamplig procedure used for statistical inference. It treats the original data sample as if it were the population, and uses it to generate a number of bootstrap samples, resamplig with replacement. Since the $SSPD$ value of the original data set is the sum of all 240 distinct squared pairwise differences between replicates, the $SSPD$ value of the bootstrap samples was obtained by randomly picking, with replacement, 240 such differences from the pool of original differences, and summing them up. Repeating this procedure $10^5$ times yields $10^5$ bootstrap samples, from which a histogram may be built and a mean value and standard deviation computed. These results are discussed in section IV, where we compare them to the objective function values achieved by the GA and PSO.

### *E. The Optimization Algorithms*

*1) The Genetic Algorithm:* Genetic algorithms [18][19][20] were first proposed by John Holland in 1975 [21], and belong to a class of evolutionary algorithms which additionally includes evolutionary programming, evolution strategies and genetic programming.

In the GA, the pool of candidate solutions is a population of individuals, which the algorithm evolves over the course of its iterations (or generations). At each iteration, some of the individuals are selected for reproduction, according to their fitness. The simplest versions of the GA apply two heuristics to obtain the next offspring - crossover and mutation - which are inspired by the genetic phenomena with the same designation. In crossover, portions of two individuals are swapped between them, while in mutation, one or several of the genes of an individual are slightly altered.

The GA, like many other metaheuristics, possesses a number of operating parameters on which its performance greatly depends. These parameters include the number of individuals or size of the population, the crossover and mutation probabilities, the specific mutation and crossover operators used, and the method of selection [19]. The optimal values for these operating parameters are generally problem dependent and need to be tuned [22] in order to obtain maximum performance for the algorithm.

*2) Particle Swarm Optimization:* Particle swarm optimization [23][24][25] is also an evolutionary algorithm, firstly proposed by Kennedy and Eberhart in 1995 [26] and inspired by the natural phenomena of bird flocking. In the PSO, the pool of candidate solutions is a swarm of particles, with each particle being characterized by a *position* and a *velocity*. In the classic version of the PSO, the directions and magnitudes of these velocities are updated, at each iteration, according to equation ((4); the new positions are then computed using the updated velocities, according to equation (5):

$$v_i^{k+1} = \phi^k v_i^k + \alpha_1 \left[\gamma_{1i} \left(p_i - x_i^k\right)\right] + \alpha_2 \left[\gamma_{2i} \left(p_g - x_i^k\right)\right] \quad (4)$$

$$x_i^k + 1 = x_i^k + v_i^{k+1} \quad (5)$$

where $x_i^k$ and $v_i^k$ are the position and velocity of particle $i$ at iteration $k$, respectively, $\phi^k$ is the inertia weight at iteration $k$, $\alpha_1$ and $\alpha_2$ are acceleration constants, $\gamma_{1i}$ and $\gamma_{2i}$ are uniform random numbers on the interval $[0, 1]$ and $p_i$ and $p_g$ are particle's $i$ and the swarm's best positions, found up to iteration $k$, respectively.

The operating parameters of the PSO are the inertia weight $\phi^k$, the local acceleration constant $\alpha_1$ and the global acceleration constant $\alpha_2$. The values of these parameters need to be tuned relative to the optimization problem, since the convergence behavior of the algorithm is greatly affected by them.

## III. Strategy for GA and PSO Performance Comparison

The strategy followed for comparing the performance of the two algorithms comprised the tuning of their most relevant operating parameters, since the values of these parameters crucially affects their performance. We formed, for each algorithm, 12 different sets of operating parameters, and ran each tuning case 20 times, departing from the same initial pool of candidate solutions (one for the GA and another for the PSO). In regard to the number of tuning cases tested, it is important to notice that each run of an algorithm takes about 5.5 hours to complete and it would be impractical to test many more cases; instead, we focused on what we considered the most important parameters and chose the values more commonly reported in the literature for the others.

In order to allow direct comparison of each algorihm's tuning results, their stopping criterion was set at a fixed number of $10^5$ objective function evaluations, since the $F_O$ evaluation is the dominant component in the time complexity of the algorithms, due to numerical integration. Because the two algorithms use pools of candidate solutions with different sizes, this resulted in a different number of iterations for each of them. This was also the reason why we used different initial populations for each algorithm.

The tuning experiments were evaluated using three different criteria [27]:

- Best and mean $F_O$ value found at the last iteration,
- Proportion of runs reaching the error criterion (mean $SSPD$),
- Number of iterations to the criterion.

This performance comparison is therefore about the value of the objective function achieved by each algorithm and the corresponding convergence behavior, and *not* about the values found for the parameters of the model, which shall not be discussed in this work. In the next two sections we describe in more detail the settings of these experiments, separating the parameters that were kept fixed in all tuning cases from the ones that were actually tuned.

### *A. Fixed Settings*

Table I shows the settings common to both algorithm which were not subject to tuning: type of problem representation, number of individuals in the population, length of each individual, and number of iterations allowed to the algorithm.

TABLE I: Fixed Settings

| Setting | GA | PSO |
|---|---|---|
| Representation | Real | Real |
| Length of Individuals | 31 | 31 |
| No. of Individuals | 150 | 31 |
| No. of Iterations | 667 | 3226 |

TABLE II: Settings for the Tuning Experiments

(a) GA Tuning Settings

| Case | $\sigma$ Decrease Mode | Crossover Function | Crossover Fraction |
|---|---|---|---|
| 1 | $M_1$ | $SinglePoint$ | 0.3 |
| 2 | $M_1$ | $SinglePoint$ | 0.6 |
| 3 | $M_1$ | $SinglePoint$ | 0.9 |
| 4 | $M_1$ | $Heuristic$ | 0.3 |
| 5 | $M_1$ | $Heuristic$ | 0.6 |
| 6 | $M_1$ | $Heuristic$ | 0.9 |
| 7 | $M_2$ | $SinglePoint$ | 0.3 |
| 8 | $M_2$ | $SinglePoint$ | 0.6 |
| 9 | $M_2$ | $SinglePoint$ | 0.9 |
| 10 | $M_2$ | $Heuristic$ | 0.3 |
| 11 | $M_2$ | $Heuristic$ | 0.6 |
| 12 | $M_2$ | $Heuristic$ | 0.9 |

(b) PSO Tuning Settings

| Case | $\phi_{initial}$ | $\phi_{final}$ | Epoch of $\phi_{initial}$ | $\alpha$ | $\frac{\alpha_1}{\alpha}$ | $mv$ |
|---|---|---|---|---|---|---|
| 1 | 0.9 | 0.4 | $N/8$ | 4 | 1/4 | $10^4$ |
| 2 | 0.9 | 0.4 | $N/8$ | 4 | 3/4 | $10^4$ |
| 3 | 0.9 | 0.4 | $N/8$ | 4 | 1/2 | $10^4$ |
| 4 | 0.9 | 0.4 | $7N/8$ | 4 | 1/4 | $10^4$ |
| 5 | 0.9 | 0.4 | $7N/8$ | 4 | 3/4 | $10^4$ |
| 6 | 0.9 | 0.4 | $7N/8$ | 4 | 1/2 | $10^4$ |
| 7 | 0.2 | 0.2 | 1 | 0.2 | 3/4 | $10^4$ |
| 8 | 0.2 | 0.2 | 1 | 0.2 | 1/2 | $10^4$ |
| 9 | 1 | 1 | 1 | 4 | 3/4 | $10^4$ |
| 10 | 1 | 1 | 1 | 4 | 1/2 | $10^4$ |
| 11 | 1 | 1 | 1 | 4 | 3/4 | 0.4 |
| 12 | 1 | 1 | 1 | 4 | 1/2 | 0.4 |

For both algorithms, we chose to code each individual as a vector of real numbers, directly corresponding to vector $\theta$ in (2), as it was the most straightforward representation for a parameter estimation problem. This seems a rather natural in the case of the PSO, which was originally created with that representation in mind, but it's not as natural in the case of the GA, for which the classic approach is to represent individuals as bit strings; however, real-coded genetic algorithms have also been studied and are very commonly used in these kinds of problems [28][29][30].

Given the real representation used, the length of each individual in the population, i.e. the dimension of the problem, should necessarily be 31, the number of parameters to be estimated.

Given the $10^5$ allowed $F_O$ evaluations for each algorithm, and the size of their pool of candidate solutions, the allowed number of iterations was computed by dividing the former by the latter, yielding the values in Table I.

Additionally, both algorithms enforce the search domain bounds by creating an initial population uniformily distributed within those bounds, and including safeguards - the GA on the crossover and mutations operators and the PSO on the position update equation (5) -, which make any candidate solution that violates a bound to take the value of that bound.

The next two subsections examine the choice of population size as well as other particular aspects of each algorithm, including their implementation.

*1) Genetic Algorithm:* We used the genetic algorithm implementation available in the MATLAB® Genetic Algorithm and Direct Search Toolbox™ to run the simulations in this work.

For the selection process, rank-based fitness scaling (implemented by function `fitscalingrank`) was used, followed by roulette wheel sampling of the parents. Each new offspring of the GA is composed of one elite child (the best individual from the previous generation), plus a fraction $c_f$ of individuals formed by crossover, with the remaining being the product of mutation.

As for the population size, they usually range from 50 to 200 [31]. A reasonable rule for determining a lower bound for the population size is that there must be at least as many individuals as there are parameters. Since we have 31 parameters and larger populations are usually favored [32], we chose a population size of 150 individuals, about 5 times the number of parameters to be estimated.

*2) Particle Swarm Optimization:* The implementation of the PSO used in this work was that developed by Brain Birge [33] for MATLAB®, in the PSOt Toolbox, with minor alterations.

In what concerns the swarm size, it has been experimentally found that the performance of the PSO is practically insensitive to it, provided that it falls in the range of 20 to 160 particles [34]. The most popular empirical study, by Carlisle and Dozier [35], suggests that a swarm of around 30 particles results in the optimal tradeoff between algorithm performance and computational cost. Since the set of benchmark functions used to obtain this result included problems with a similar dimension, this suggestion was readily embraced and we made the swarm size equal to the dimension of the problem.

## B. Tuning

*1) Genetic Algorithm:* The tuning procedure explored the effect of two different crossover functions, three different crossover fractions, and two different schemes of decrease for the standard deviation $\sigma$ of the Gaussian mutation function [36]. The crossover functions tested were the classical single point crossover and Wright's heuristic crossover, as implemented, respectively, by MATLAB® functions `crossoversinglepoint` and `crossoverheuristic`. The latter function numerically combines two parents to produce a child that lies in the line defined by them, closer to the fittest parent and in the direction away from the parent with lowest fitness. The crossover fractions, $c_f$, tested were 0.3, 0.6 and 0.9, and the decrease of $\sigma$ throughout the run of the GA was, for the cases labeled $M_1$, given by equation (6), and for the cases labeled $M_2$ given by equation (7).

$$\sigma_k = \sigma_1 \left(1 - shrink\frac{k}{N}\right), \quad k = 1, \ldots, N \tag{6}$$

$$\sigma_k = \begin{cases} \sigma_1 \left( 1 - shrink \dfrac{k}{N} \right) & \text{if } 1 < k \leq \dfrac{N}{2}, \\ \sigma_{\frac{N}{2}} & \text{if } k > \dfrac{N}{2}. \end{cases} \tag{7}$$

with $k$ being the generation index, $N$ the total number of generations, and $shrink$ a parameter that was set to 1. In scheme $M_1$, $\sigma$ decreases all the way to zero at the end of the run, while in scheme $M_2$ it decreases only until halfway through the run, and remains constant thereafter. The GA tuning cases are summarized in Table II(a).

*2) Particle Swarm Optimization:* The tuning design was based on the analysis of the convergence behavior of the PSO algorithm made by Trelea [37]. In this analysis, three different types of behavior are predicted, depending on the values of the inertia weight and on the sum of the acceleration constants, $\alpha = \alpha_1 + \alpha_2$:

- convergence with oscillation or zigzagging,
- convergence without oscillation or zigzagging, and
- divergence.

Within some of these cases, the effects of different fractions of local acceleration, $\alpha_1/\alpha$, and of different schemes of inertia weight decrease were explored. For the divergent cases, it was tested if a lower $mv$ would help in containing the expected divergent behavior of the algorithm, while taking advantage of the enhanced exploration capacity of this type of behavior. The tuning cases are summarized in Table II(b).

## IV. Results and Discussion

### A. GA Tuning Experiments

The results for the tuning of the GA are shown in Table III(a) below. The numbering of the cases in this table is the same as in Table II(a).

Cases 7 to 12, for which $\sigma$ decrease was given by (7), performed, in general, much worse than cases 1 to 6, for which $\sigma$ decrease was given by (6). In fact, only when using single point crossover with a crossover fraction of 0.9 did 10% of the runs satisfy the error criterion (mean $SSPD$). It thus stands to reason that the increased interval of mutation, developed with the aim of obviating premature convergence, had, on the contrary, a negative effect on the algorithm's performance.

Cases 1, 4, 7 and 10, where the lowest crossover fraction of 0.3 was used, were, in general, the poorest performers, whereas cases 3, 6, 9 and 12, where a crossover fraction of 0.9 was used, were almost always the best performers. Therefore, for this particular problem, the role of the crossover operator is crucial for the performance of the GA, and a high crossover fraction of 0.6 to 0.9 is preferable to a lower one.

As for the crossover function, it was found that, for the instances with the highest crossover fractions, 0.6 and 0.9, single point crossover always outperformed heuristic crossover, in terms of best and mean objective function values, while giving equal or better performance in terms of percentage of runs reaching the criterion. For the lowest crossover fraction, 0.3, heuristic crossover did better than single point crossover, but none of the corresponding runs ever reached the criterion. This result may simply be an indication that the ability of heuristic crossover to use fitness information became an advantage over single point crossover when the population became highly heterogeneous due to the disruptive effect of the high mutation range.

The best case was case 3, using single point crossover with a crossover fraction of 0.9, and with $\sigma$ decreasing all the way to zero at the end of the run.

### B. PSO Tuning Experiments

Table III(b) below summarizes the results obtained for the tuning of the PSO, with case numbering corresponding to that defined in Table II(b).

Cases 1 to 6, 7 and 8, and 9 to 12, grouping three distinct swarm behaviors, clearly show three differentiated ranges of values for best and mean $F_O$ value. Indeed, when the operating parameters made the swarm behavior fall in either the divergent or the convergent without oscillation or zigzagging categories, none of the runs reached the error criterion. On the other hand, the convergent with oscillations or zigzagging zone always produced at least one run satisfying the criterion, and achieved much lower $F_O$ values.

The worst results were obtained for cases 7 and 8, with convergent behavior without oscillation or zigzagging. This was probably because the algorithm prematurely converged to a local minimum and was not able to escape that area of the search space, owing to its extremely low acceleration constants.

Among the divergent cases, the results were better for the cases where the maximum velocity was set lower (0.4 as compared to $10^4$), but they were still poor when compared to cases 1 to 6. While lowering the value of the maximum velocity might prove more effective in obtaining better performances in this divergent zone, it could also limit the swarm's ability to escape local minima [38].

In general, and for all swarm behaviors, the best results were found when the local acceleration $\alpha_1$ was larger than the global acceleration $\alpha_2$, while the worst corresponded to the opposite situation.

The best case was case 2, in which the algorithm has a convergent behavior with oscillation or zigzagging. In this case, the inertia weight reached its final lowest value earlier in the run, and $\alpha_1$ was three times larger than $\alpha_2$.

### C. Comparison of GA and PSO Results

The PSO results for cases 1 to 6 (see Table III(b)), referring to convergent behavior with oscillations or zigzagging, were, in general, much better than the best GA results (see first 4 columns of Table III(a)). The average number of iterations to the criterion was, in some cases, higher for the PSO than for the GA, but the amount of time the algorithm took to compute them was not, owing to its

TABLE III: Tuning Results

(a) GA Tuning Results

| Case | Best | Mean | % Reaching Criterion | Iterations to Criterion |
|---|---|---|---|---|
| 1 | 1940.8 | 3241.1 | 0 | – |
| 2 | 1190.3 | 1845.4 | 15 | 592.33 |
| 3 | 1176.9 | 1653.6 | 50 | 545.4 |
| 4 | 1920 | 2579.1 | 0 | – |
| 5 | 1653.6 | 2322.1 | 0 | – |
| 6 | 1534.3 | 2598.9 | 5 | 650 |
| 7 | 2372.4 | 3004.9 | 0 | – |
| 8 | 1586.7 | 2243.8 | 0 | – |
| 9 | 1191.9 | 2043.4 | 10 | 395 |
| 10 | 2017.5 | 2884 | 0 | – |
| 11 | 1775.7 | 2609.7 | 0 | – |
| 12 | 1630.6 | 2429.1 | 0 | – |

(b) PSO Tuning Results

| Case | Best | Mean | % Reaching Criterion | Iterations to Criterion |
|---|---|---|---|---|
| 1 | 1064 | 1831.9 | 15 | 272 |
| 2 | 841.66 | 1411.1 | 60 | 839.5 |
| 3 | 1002.5 | 1862.6 | 20 | 195.5 |
| 4 | 1186.9 | 2498.6 | 5 | 1444 |
| 5 | 922.38 | 1574.6 | 40 | 1721.3 |
| 6 | 1186.9 | 1966 | 15 | 886 |
| 7 | 5572.3 | 10102 | 0 | – |
| 8 | 6277.1 | 10282 | 0 | – |
| 9 | 2306.8 | 2725 | 0 | – |
| 10 | 1975.7 | 2971.4 | 0 | – |
| 11 | 2076.4 | 2882.6 | 0 | – |
| 12 | 1590.7 | 3367 | 0 | – |

smaller population size and hence fewest $F_O$ evaluations per iteration.

PSO tuning cases 7 and 8 were, on the other hand, much worse than the worst GA results, whereas divergent cases (9 to 12) were poor, but comparable to the GA's poorest results. This is what may be observed in Fig. 2, which shows the $SSPD$ histogram obtained via bootstrapping, with the best and worst results from each algorithm superimposed. For the sake of readability, we omitted the worst PSO case (case 8, in the convergent without oscillations zone) and instead included the worst case in the divergent zone (case 9). The mean value and standard deviation of the $SSPD$ were $\widehat{SSPD} = 1192.6$ and $\sigma_{SSPD} = 367.3$.

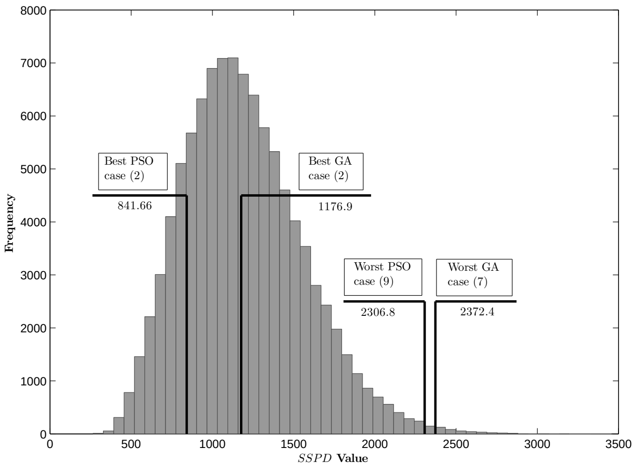

Fig. 2: Histogram of experimental $SSPD$ with representative optimization results from the GA and PSO superimposed.

The overall conclusion of the tuning experiments was that the PSO is more appropriate than the GA for solving this problem, since, according to the criteria examined, and for the best tuning settings, the former yielded significantly better results than the latter. This conclusion agrees with the results obtained by Drager *et al* [8], for the estimation of parameters of similar models (metabolic models described by systems of ODEs with a comparable number of parameters), as they also favor the use of the PSO (albeit in a slightly different variant) to the use of the GA (with settings similar to the present ones). We think that this may derive from the simpler, more straightforward manner in which the PSO is formulated in relation to this kind of problem. Indeed, not only the search domain of the problem is directly transposable to the search domain of the algorithm, but the operating parameters that control the balance of exploration and exploitation, or global search ability and local search ability are independent from each other, easy to interpret, and with more predicteable effects.

## V. Conclusions

The fitting of fermentation data to kinetic models of microbial growth is a complex problem for which the usual deterministic optimization algorithms, based on the derivatives of the objective function, yield poor results. This work tested two alternative parameter estimation strategies, using population-based stochastic optimization algorithms, namely particle swarm optimization (PSO) and a genetic algorithm (GA). It was concluded that the PSO performed much better than the GA. Furthermore, PSO tuning also revealed that the operating parameters of the algorithm should be such that its behavior is convergent with oscillations and/or zigzagging, and local search ability is preferred over global search ability.

## Acknowledgment

This work was supported by the FCT (ISR/IST plurianual funding) through the PIDDAC Program funds.

## References

[1] K. Kovarova-Kovar and T. Egli, "Growth kinetics of suspended microbial cells: from single-substrate-controlled growth to mixed-substrate kinetics," *Microbiol.Mol.Biol.Rev.*, vol. 62, no. 3, pp. 646–666, Sep. 1998. [Online]. Available: PM:9729604

[2] C. Kelley, *Iterative Methods for Optimization*, SIAM, Ed. SIAM, 1999, no. 18.

[3] Y. Mutsunori and I. Toshihide, "On metaheuristic algorithms for combinatorial optimization problems," *Systems and Computers in Japan*, vol. 32, no. 3, pp. 33–55, 2001.

[4] C. G. Moles, P. Mendes, and J. R. Banga, "Parameter estimation in biochemical pathways: A comparison of global optimization methods," *Genome Research*, vol. 13, no. 11, pp. 2467–2474, 2003. [Online]. Available: http://genome.cshlp.org/content/13/11/2467.abstract

[5] E. C. Rivera, A. C. Costa, D. I. Atala, F. Maugeri, M. R. Maciel, and R. M. Filho, "Evaluation of optimization techniques for parameter estimation: Application to ethanol fermentation considering the effect of temperature," *Process Biochemistry*, vol. 41, no. 7, pp. 1682–1687, Jul. 2006.

[6] I. Rocha and E. Ferreira, "Yield and kinetic parameters estimation and model reduction in a recombinant e. coli fermentation," in *ESCAPE : European Symposium on Computer-Aided Process Engineering*. Elsevier, 2004, pp. –.

[7] F. Espinoza-Quiñones, A. Módenes, L. Thomé, S. Palácio, D. Trigueros, A. Oliveira, and N. Szymanski, "Study of the bioaccumulation kinetic of lead by living aquatic macrophyte salvinia auriculata," *Chemical Engineering Journal*, vol. 150, no. 2-3, pp. 316 – 322, 2009.

[8] A. Drager, M. Kronfeld, M. Ziller, J. Supper, H. Planatscher, J. Magnus, M. Oldiges, O. Kohlbacher, and A. Zell, "Modeling metabolic networks in c. glutamicum: a comparison of rate laws in combination with various parameter optimization strategies," *BMC Systems Biology*, vol. 3, no. 1, pp. 5–, 2009. [Online]. Available: http://www.biomedcentral.com/1752-0509/3/5

[9] R. B. Boozarjomehry and M. Masoori, "Which method is better for the kinetic modeling: Decimal encoded or binary genetic algorithm?" *Chemical Engineering Journal*, vol. 130, no. 1, pp. 29–37, May 2007.

[10] M. Maeder, Y. M. Neuhold, and G. Puxty, "Application of a genetic algorithm: near optimal estimation of the rate and equilibrium constants of complex reaction mechanisms," *Chemometrics and Intelligent Laboratory Systems*, vol. 70, no. 2, pp. 193–203, Feb. 2004.

[11] R. Mendes, I. Rocha, E. Ferreira, and M. Rocha, "A comparison of algorithms for the optimization of fermentation processes," in *2006 IEEE Congress on Evolutionary Computation*, 2006, pp. 2018–2025.

[12] L. Duarte, F. Carvalheiro, J. Tadeu, and F. Girio, "The combined effects of acetic acid, formic acid, and hydroquinone on debaryomyces hansenii physiology," *Appl.Biochem.Biotechnol.*, vol. 129-132, pp. 461–475, 2006. [Online]. Available: PM:16915662

[13] D. Calçada, "Modeling of the physiology of d. hansenii using population-based search methods for parameter estimation," Master's thesis, Instituto Superior Técnico, November 2009.

[14] F. Gírio, C. Amaro, H. Azinheira, F. Pelica, and M. Amaral-Collaço, "Polyols production during single and mixed substrate fermentations in debaryomyces hansenii," *Bioresour.Technol.*, vol. 71, no. 3, pp. 245–251, 2000.

[15] L. Shampine and M. Reichelt, "The matlab ode suite," *Siam Journal on Scientific Computing*, vol. 18, no. 1, pp. 1–22, 1997. [Online]. Available: WOS:A1997WE98100002

[16] P. Heffernan, "New measures of spread and a simpler formula for the normal-distribution," *American Statistician*, vol. 42, no. 2, pp. 100–102, 1988. [Online]. Available: WOS:A1988N803200002

[17] B. Efron and R. J. Tibshirani, *An Introduction to the Bootstrap*. New York: Chapman & Hall, 1993.

[18] D. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Company, Inc., 1989.

[19] M. Mitchell, *An Introduction to Genetic Algorithms*. MIT Press, 1996.

[20] D. Whitley, "An overview of evolutionary algorithms: practical issues and common pitfalls," *Information and Software Technology*, vol. 43, no. 14, pp. 817–831, 2001. [Online]. Available: WOS:000172495300003

[21] J. H. Holland, *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. University of Michigan Press, 1975.

[22] J. Smith and T. Fogarty, "Operator and parameter adaptation in genetic algorithms," *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, vol. 1, no. 2, pp. 81–87, Jun. 1997. [Online]. Available: http://dx.doi.org/10.1007/s005000050009

[23] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization," *Swarm Intelligence*, vol. 1, no. 1, pp. 33–57, Jun. 2007. [Online]. Available: http://dx.doi.org/10.1007/s11721-007-0002-0

[24] M. Clerc and J. Kennedy, "The particle swarm - explosion, stability, and convergence in a multidimensional complex space," *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 1, pp. 58–73, 2002.

[25] A. Banks, J. Vincent, and C. Anyakoha, "A review of particle swarm optimization. part i: background and development," *Natural Computing*, vol. 6, no. 4, pp. 467–484, Dec. 2007. [Online]. Available: http://dx.doi.org/10.1007/s11047-007-9049-5

[26] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *1995 Ieee International Conference on Neural Networks Proceedings, Vols 1-6*, 1995, pp. 1942–1948. [Online]. Available: ISI:A1995BF46H00374

[27] R. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: simpler, maybe better," *Evolutionary Computation, IEEE Transactions on*, vol. 8, no. 3, pp. 204–210, 2004.

[28] A. Wright, "Genetic algorithms for real parameter optimization," in *Foundations of Genetic Algorithms*, G. Rawlins, Ed. Morgan Kaufmann Publishers, 1991, pp. 205–218.

[29] F. Herrera, M. Lozano, and J. Verdegay, "Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis," *Artificial Intelligence Review*, vol. 12, no. 4, pp. 265–319, 1998.

[30] I. G. Tsoulos, "Modifications of real code genetic algorithm for global optimization," *Applied Mathematics and Computation*, vol. 203, no. 2, pp. 598–607, 2008.

[31] G. Rawlins, *Foundations of Genetic Algorithms*. Morgan Kaufmann, 1991, no. 1.

[32] D. Goldberg, K. Deb, and J. Clark, "Genetic algorithms, noise, and the sizing of populations," *Complex Systems*, vol. 6, pp. 333–362, 1991.

[33] B. Birge, "Psot - a particle swarm optimization toolbox for use with matlab," in *Proceedings of the 2003 Ieee Swarm Intelligence Symposium (Sis 03)*, 2003, pp. 182–186. [Online]. Available: ISI:000183013200027

[34] Y. Shi and R. Eberhart, "Empirical study of particle swarm optimization," in *Proceedings of the 1999 Congress on Evolutionary Computation CEC99*, vol. 3, 1999, pp. –1950.

[35] A. Carlisle and G. Dozier, "An off-the-shelf pso," in *Proceedings of the 2001 Workshop on Particle Swarm Optimization Indianapolis*, 2001, pp. 1–6.

[36] R. Hinterding, "Gaussian mutation and self-adaption for numeric genetic algorithms," in *1995 Ieee International Conference on Evolutionary Computation, Vols 1 and 2*, 1995, pp. 384–389. [Online]. Available: WOS:A1995BF29M00071

[37] I. Trelea, "The particle swarm optimization algorithm: convergence analysis and parameter selection," *Information Processing Letters*, vol. 85, no. 6, pp. II–, 2003. [Online]. Available: WOS:000181133600006

[38] J. Kennedy, "The particle swarm: social adaptation of knowledge," in *IEEE International Conference on Evolutionary Computation*, 1997, pp. 303–308.