Expert Systems with Applications 41 (2014) 2186–2195

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Optimization of fed-batch fermentation processes with bio-inspired algorithms

Miguel Rocha [a,1], Rui Mendes [a,*], Orlando Rocha [a,1], Isabel Rocha [b], Eugénio C. Ferreira [b]

[a] CCTC, School of Engineering, University of Minho, Campus Gualtar, 4710-057 Braga, Portugal
[b] IBB–Institute Biotechnology and Bioengineering, Centre for Biological Engineering, University of Minho, Campus Gualtar, 4710-057 Braga, Portugal

ARTICLE INFO

Keywords:
Fed-batch fermentation
Differential Evolution
Evolutionary algorithms
Particle Swarm Optimization
Feeding trajectory optimization

ABSTRACT

The optimization of the feeding trajectories in fed-batch fermentation processes is a complex problem that has gained attention given its significant economical impact. A number of bio-inspired algorithms have approached this task with considerable success, but systematic and statistically significant comparisons of the different alternatives are still lacking. In this paper, the performance of different meta-heuristics, such as Evolutionary Algorithms (EAs), Differential Evolution (DE) and Particle Swarm Optimization (PSO) is compared, resorting to several case studies taken from literature and conducting a thorough statistical validation of the results. DE obtains the best overall performance, showing a consistent ability to find good solutions and presenting a good convergence speed, with the DE/rand variants being the ones with the best performance. A freely available computational application, OptFerm, is described that provides an interface allowing users to apply the proposed methods to their own models and data.

1. Introduction

In addition to the considerable number of valuable products that have long been produced using fermentation techniques, such as recombinant proteins and antibiotics, Biotechnology has been replacing traditional manufacturing processes in many areas that were until recently dominated by the chemical industry. When compared with conventional production methods, biotechnological processes have relatively low energy requirements and environmental costs, as well as decreased waste generation associated with the possibility of creating biodegradable products and of using renewable raw materials.

Industrial or White Biotechnology can be defined as the use of cells or enzymes for the production of commodity and specialty chemicals. The share of biotechnological processes in the production of various chemical processes is currently raising and this trend is expected to continue. In this new reality, rational fermentation optimization approaches gain a novel fundamental importance since, when compared with pharmaceutical processes, the margins in industrial biotechnological processes are much lower and competitiveness depends more on product price than on patent protection. In fact, for such processes, an increase in productivity or efficiency can have a direct and strong impact in profitability.

Under this scope, several optimization strategies can be delineated, depending on the defined purposes. The process engineer point of view aims at optimizing the fermentation process by exploiting the maximum capabilities of an already selected microorganism and by manipulating environmental and operational variables, therefore using potentially similar tools, but with a different perspective when compared with optimization strategies that envisage the design of improved strains (Patil, Rocha, Forster, & Nielsen, 2005). The challenge consists in controlling a bioprocess at its optimal state in order to reach its maximum productivity with the lowest possible cost.

In recent years, the efforts devoted to the application of process engineering approaches to optimize biotechnological production processes have focused on the dynamic optimization (open-loop optimal control) of fed-batch bioreactors. This optimization has traditionally been done on the substrate feed rate as key manipulated variable.

For that purpose, the optimization strategies described in the literature include both analytical and numerical methods. It has been shown that for relatively simple bioreactor systems, which are expressed by differential equations models, the optimization problem can be solved analytically by applying the Pontryagin's Minimum Principle (PMP). However, in the majority of the cases

* Corresponding author. Tel.: +351 253604456; fax: +351 253 604471.
E-mail addresses: mrocha@di.uminho.pt (M. Rocha), azuki@di.uminho.pt (R. Mendes), ornrocha@di.uminho.pt (O. Rocha), irocha@deb.uminho.pt (I. Rocha), ecferreira@deb.uminho.pt (E.C. Ferreira).
[1] Tel.: +351 253604456; fax: +351 253 604471.

0957-4174/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2013.09.017

reported, determination of the optimal feed rate profile is a problem of singular control, because the control variable (feed rate) often appears linearly in the system of differential equations (Shin & Lim, 2006).

Thus, this approach fails to provide a complete solution (Shukla & Pushpavanam, 1998; Tartakovsky, Ulitzur, & Sheintuch, 1995). Nevertheless, in several studies, singular control theory was used to solve the analytical control problem (Lee & Ramirez, 1994; Levisauskas et al., 2003; Modak, Lim, & Tayeb, 1986; Park & Ramirez, 1988), while an unified methodology was proposed by Smets, Claes, November, Bastin, and Impe (2004), the so-called optimal adaptive control strategy, where a theoretically realizable optimum is obtained using the PMP and an adaptive controller is derived based on those results to correct on-line deviations caused by model inaccuracies. However, the analytical approach becomes too complex when the number of state and control variables increases. Moreover, those methods can only be applied to differential equation models and do not satisfy for alternative bioreactor modeling methods such as neural networks or fuzzy models.

Here, numerical methods are required. These can be divided into deterministic (usually local) and stochastic (usually global) methods (Banga, Moles, & Alonso, 2003). Popular techniques that follow the first approach are gradient-based local methods like Sequential Quadratic Programming that are applied to the non-linear programming problem obtained after Control Vector Parameterization (Bapat, Sohoni, Moses, & Wangikar, 2006). The main disadvantages of these methods are the computational cost, as well as premature convergence for local optima, especially if the optimization is started far away from the global solution.

Because of those limitations, some promising approaches regarding methods for bioprocess optimization rely on the use of stochastic metaheuristic algorithms, namely those belonging to the class of Evolutionary Computation (EC). With these methods, although global optimality cannot be guaranteed, good solutions are normally obtained in relatively modest computational times. Popular tools like Genetic and Evolutionary Algorithms (EAs), together with other approaches like Evolution Strategies and Differential Evolution (DE) belong to this class.

Several applications of EC algorithms to bioprocesses can be found in the literature. For example, Roubos, van Straten, and van Boxtel (1999) compared the performance of a class of EAs with first order gradient algorithms and with dynamic programming. The results show a good and often superior performance of EAs in comparison with other methods. These approaches were applied to hybridoma cells and to recombinant protein production with *E. coli*. A very similar work (for hybridoma cells) can be found in Nguang, Chen, and Chen (2001).

Integrated methods based on EAs for the maximization of cell mass production in recombinant *S. cerevisiae* and *Aureobasidium pullulans* fed-batch cultures are described in Na, Chang, Chung, and Lim (2002) and Ronen, Shabtai, and Guterman (2002), respectively. In Sarkar and Modak (2003) and Sarkar and Modak (2004), the authors use EAs for the determination of the correct switching structure, as well as the feed rate in the singular intervals obtained by applying the optimal control theory in several case-studies.

Artificial Neural Networks (ANNs), as an alternative modeling scheme, have been used by Zuo and Wu (2000) and Chen, Nguang, Chen, and Li (2004) to represent bioprocesses that are optimized with EAs. A different approach was used by Franco-Lara and Weuster-Botz (2005), in which the feeding profile is represented by means of ANNs that are optimized by EAs. Following a similar approach a hybrid combination of ANNs and DE was presented in Dragoi, Curteanu, Galaction, and Cascaval (2013) with the aim of optimizing the oxygen transfer coefficient in aerobic fermentation processes.

Other stochastic nature-inspired methods have also been applied in the optimization of fed-batch fermentations. In Kookos (2004), the authors used Simulated Annealing to optimize the feeding profile in ethanol and penicillin production, while Jayaraman, Kulkarni, Gupta, Rajesh, and Kusumaker (2001) used the Ant Algorithm for the optimization of the first case study and of protein production in *E. coli*.

Some authors claim that, when compared with EAs, DE is a more appropriate solution for those applications (Banga et al., 2003). In Ros et al. (2013), the authors have compared several stochastic optimization methods in the estimation of the kinetic parameters of an alcoholic fermentation model, and concluded that DE had the best performance, even suggesting their integration in hybrid search procedures.

Among the few works reported, DE has been applied to the optimization of the fed-batch fermentation of *Zymomous mobilis* (Chiou & Wang, 1999), bacteriocin production with lactic acid bacteria (Moonchai, Madlhoo, Jariyachavalit, & Shimizu, 2005), ethanol production in *Saccharomyces cerevisiae* (Wang & Cheng, 1999) and multiple bioprocesses (Kapadi & Gudi, 2004). However, the performance of the different EC and other nature-inspired algorithms has scarcely been compared in an effective way. Moreover, most of the studies described in the literature are based on only one run of the algorithms which, for stochastic methods, is not appropriate.

Therefore, in this work, a more thorough comparison is proposed, that involves several bio-inspired algorithms (from the classes of EAs, DE and Particle Swarm Optimization – PSO) and that goes through an appropriate process of statistical validation of the results. Four case studies were used to compare the performance of the different algorithms, two of them encompassing two distinct feed variables. Each algorithm was allowed to run for a given number of function evaluations and the comparison among the methods was based on their final result and also on the convergence speed of the algorithms, i.e. on the computational time required to obtain a high-quality solution. For every algorithm, 30 runs were conducted for each case study to achieve statistical significance and appropriate statistical tests were performed. The results obtained are clearly favorable to the DE/rand variants, that outperform both EAs and PSO.

As a complement to this work, a computational application with an user friendly interface was developed and is made available for the community. This allows users to validate the proposed algorithms and apply them to their own models. The framework is also fully modular allowing interested researchers to develop their own algorithms or to improve the application with other functionalities.

2. Case studies: fed-batch fermentation processes

In fed-batch fermentations there is an addition of certain nutrients along the process, allowing the achievement of higher product concentrations. During this process the system's states change considerably, from initially low to high biomass and product concentrations. This dynamic behavior motivates the development of optimization methods to find the optimal input feeding trajectories in order to improve the process performance. For the proper simulation of the process, a white box mathematical model is typically developed, based on differential equations that represent the mass balances of the relevant state variables.

The selected four case studies cover a wide range of industrial applications and organisms. These are detailed in the following sections.

2.1. Case study I

In previous work by the authors, a fed-batch recombinant *E. coli* fermentation process was optimized by EAs (Rocha & Ferreira, 2002; Rocha, Neves, Rocha, & Ferreira, 2004). This was considered as the first case study in this work. During the aerobic growth of the bacterium, with glucose as the only added substrate, the micro-organism can follow three main different metabolic pathways:

- Oxidative growth on glucose:

$$k_1S + k_5O \xrightarrow{\mu_1} X + k_8C \tag{1}$$

- Fermentative growth on glucose:

$$k_2S + k_6O \xrightarrow{\mu_2} X + k_9C + k_3A \tag{2}$$

- Oxidative growth on acetic acid:

$$k_4A + k_7O \xrightarrow{\mu_3} X + k_{10}C \tag{3}$$

where S, O, X, C, A represent glucose, dissolved oxygen, biomass, dissolved carbon dioxide and acetate components, respectively. In the sequel, the same symbols are used to represent the state variables' concentrations (g/kg); μ_1 to μ_3 (h^{-1}) are time variant specific growth rates that nonlinearly depend on the state variables, and k_i are constant yield coefficients.

The associated dynamical model can be described by the following equations:

$$\frac{dX}{dt} = (\mu_1 + \mu_2 + \mu_3)X - DX \tag{4}$$
$$\frac{dS}{dt} = (-k_1\mu_1 - k_2\mu_2)X + \frac{F_{in,S}S_{in}}{W} - DS \tag{5}$$
$$\frac{dA}{dt} = (k_3\mu_2 - k_4\mu_3)X - DA \tag{6}$$
$$\frac{dO}{dt} = (-k_5\mu_1 - k_6\mu_2 - k_7\mu_3)X + OTR - DO \tag{7}$$
$$\frac{dC}{dt} = (k_8\mu_1 + k_9\mu_2 + k_{10}\mu_3)X - CTR - DC \tag{8}$$
$$\frac{dW}{dt} \simeq F_{in,S} \tag{9}$$

where $D = F_{in,S}/W$ is the dilution rate (h^{-1}), $F_{in,S}$ the substrate feeding rate (kg/h), W the fermentation weight (kg), S_{in} is the substrate concentration in the feeding solution, OTR the oxygen transfer rate and CTR the carbon dioxide transfer rate (g/kg h).

The kinetic behavior, expressed in the rates μ_1 to μ_3, was given by specific functions of the state variables, that are out of the scope of the present work but can be found in Rocha (2003). The purpose of the optimization is to determine the feeding rate profile ($F_{in,S}(t)$) that maximizes the productivity of the process, defined as the units of product (recombinant protein) formed per unit of time. In this case, this is usually related with the final biomass obtained, when the duration of the process is pre-defined. Thus, a *performance index (PI)* is defined by the following expression:

$$PI = \frac{X(t_f)W(t_f) - X(0)W(0)}{t_f} \tag{10}$$

The relevant state variables are initialized with the following values: $X(0) = 5$, $S(0) = 0$, $A(0) = 0$, $W(0) = 3$. Due to limitations in the feeding pump capacity, the value of $F_{in,S}(t)$ must be in the range $[0.0; 0.4]$. Furthermore, the following constraint is defined over the value of W:$W(t) \leqslant 5$. The final time (t_f) is set to 25 (h).

2.2. Case study II

This process is a fed-batch bioreactor for the production of ethanol by *Saccharomyces cerevisiae*, firstly studied by Chen and Hwang (1990). The aim is to find the substrate feed rate profile that maximizes the final amount of ethanol. The model equations are the following:

$$\frac{dx_1}{dt} = g_1x_1 - u\frac{x_1}{x_4} \tag{11}$$
$$\frac{dx_2}{dt} = -10g_1x_1 + u\frac{150 - x_2}{x_4} \tag{12}$$
$$\frac{dx_3}{dt} = g_2x_1 - u\frac{x_3}{x_4} \tag{13}$$
$$\frac{dx_4}{dt} = u \tag{14}$$

where x_1, x_2 and x_3 are the cell mass, substrate and ethanol concentrations (g/L), x_4 the volume of the reactor (L) and u the feeding rate (L/h).

On the other hand, the kinetic variables g_1 and g_2 (h^{-1}) are given by:

$$g_1 = \frac{0.408}{(1 + \frac{x_3}{16})}\frac{x_2}{(0.22 + x_2)} \tag{15}$$
$$g_2 = \frac{1}{(1 + \frac{x_3}{71.5})}\frac{x_2}{(0.44 + x_2)} \tag{16}$$

The *performance index (PI)* is given by:

$$PI = x_3(t_f)x_4(t_f) \tag{17}$$

The final time is set to $t_f = 54$ (h), and the initial values for the state variables are the following: $x_1(0) = 1$, $x_2(0) = 150$, $x_3(0) = 0$ and $x_4(0) = 10$. Additionally, there are physical constraints over the variables, namely: $0 \leqslant x_4(t) \leqslant 200$ and $0 \leqslant u(t) \leqslant 12$.

2.3. Case study III

This case study consists in a hybridoma reactor for the production of monoclonal antibodies, described by the equations (Roubos et al., 1999):

$$\frac{dX_v}{dt} = (\mu - k_d)X_v - \frac{F_1 + F_2}{V}X_v \tag{18}$$
$$\frac{dGlc}{dt} = \frac{F_1}{V}Glc_{in} - \frac{F_1 + F_2}{V}Glc - q_{Glc}X_v \tag{19}$$
$$\frac{dGln}{dt} = \frac{F_2}{V}Gln_{in} - \frac{F_1 + F_2}{V}Gln - q_{Gln}X_v \tag{20}$$
$$\frac{dLac}{dt} = q_{Lac}X_v - \frac{F_1 + F_2}{V}Lac \tag{21}$$
$$\frac{dAmm}{dt} = q_{Amm}X_v - \frac{F_1 + F_2}{V}Amm \tag{22}$$
$$\frac{dMab}{dt} = q_{Mab}X_v - \frac{F_1 + F_2}{V}Mab \tag{23}$$
$$\frac{dV}{dt} = (F_1 + F_2) \tag{24}$$

where the state variables X_v, Glc, Gln, Lac, Amm, Mab are the concentrations of viable cells (cells/L) and of glucose, glutamine, lactate, ammonia and monoclonal antibodies (g/L), respectively; Glc_{in} and Gln_{in} are glucose and glutamine concentrations in the feeding solutions, and V is the culture volume (L). The control variables F_1 and F_2 (L/h) are the volumetric feed rates. The complete kinetic expressions for μ, k_d, q_{Glc}, q_{Gln}, q_{Lac}, q_{Amm} and q_{Mab} are given in Roubos et al. (1999).

The target of the optimization process, in this case, is to increase the total amount of monoclonal antibodies produced. So, the *PI* is given by:

$$PI = \int_0^{t_f} q_{Mab}X_v(t)V(t) \tag{26}$$

Initialization values for the state variables are the following: $X_v = 2.0 \times 10^8$, $Glc = 25$, $Gln = 4$, $Lac = 0$, $Amm = 0$, $Mab = 0$, $V = 0.8$. t_f is 10 (days) and the value of $V(t)$ is constrained by $V(t) \leqslant V_{max}$.

2.4. Case study IV

This process was proposed by Lee and Ramirez (1994) integrated in the development of optimal control policies for induced foreign protein production in bacteria.

$$\frac{dx_1}{dt} = u_1 + u_2 \tag{27}$$
$$\frac{dx_2}{dt} = \mu x_2 - \frac{u_1 + u_2}{x_1} x_2 \tag{28}$$
$$\frac{dx_3}{dt} = \frac{u_1}{x_1} C_{nf} - \frac{u_1 + u_2}{x_1} x_3 - \mu Y^{-1} x_2 \tag{29}$$
$$\frac{dx_4}{dt} = R_{fp} x_2 - \frac{u_1 + u_2}{x_1} x_4 \tag{30}$$
$$\frac{dx_5}{dt} = \frac{u_2}{x_1 C_{if}} - \frac{u_1 + u_2}{x_1} x_5 \tag{31}$$
$$\frac{dx_6}{dt} = -k_1 x_6 \tag{32}$$
$$\frac{dx_7}{dt} = k_2(1 - x_7) \tag{33}$$

where x_1 to x_7 are the state variables representing the reactor volume (L), cell, substrate, foreign protein and inducer concentrations (g/L), inducer shock and recovery factors on the cell growth rate, respectively. The two control variables (u_1 and u_2) are the glucose and inducer feed rates (L/h). The additional variables Y, C_{nf} and C_{if} represent the growth yield coefficient, nutrient concentration and inducer concentration on the corresponding feeds (the values of 0.51, 100 and 4.0 were used).

The process kinetics is given by:

$$R_R = \frac{0.22}{0.22 + x_5} \tag{35}$$
$$\mu = \frac{0.407 x_3}{0.108 + x_3 + \frac{x_3^2}{14815.8}} (x_6 + x_7 R_R) \tag{36}$$
$$R_{fp} = \frac{0.095 x_3}{0.0108 + x_3 + \frac{x_3^2}{14815.8}} \frac{0.0005 + x_5}{0.022 + x_5} \tag{37}$$
$$k_1 = k_2 = \frac{0.09 x_5}{0.034 + x_5} \tag{38}$$

The *PI* is defined as:

$$PI = x_4(t_f) x_1(t_f) - Q \int_0^{t_f} u_2(t) dt \tag{39}$$

where Q is the ratio of the cost of the inducer to the value of the protein product. The initial conditions for x_1 to x_7 are defined as: 1,0.1,40,0,0,1,0, $Q = 5$ and $t_f = 15$ (h).

3. Algorithms

In this section, the algorithms used for the optimization are described. First, some general comments about solution representation and evaluation are given. Then, the different algorithms used are described in detail.

3.1. Solution representation and evaluation

The optimization task addressed in this work aims at finding the best trajectory of some input variables (feeding), that yield the maximum performance index, defined in each specific case. Case studies I and II have only one input variable, while in case studies III and IV there are two variables to optimize.

A solution to the problem will consist of a set of V real-valued vectors of equal length $L + 1$, where V is the number of input variables. Each vector encodes an input variable as a temporal sequence of values, defined as a piecewise linear function, with L segments. Feeding values are provided only at $L + 1$ equally spaced points while the remaining values are linearly interpolated. The size of a solution will therefore be given by $V \times (L + 1)$, since the V vectors are joined sequentially to create a solution. The value of parameter L needs to be set before the optimization process. Larger values of L increase the search space, thus making the problem harder, while augmenting the flexibility of the possible feeding profiles.

The evaluation process, for each solution, is achieved by running a numerical simulation of the defined model, given as input the feeding values. The numerical simulation is performed using *ODEToJava*, a package implementing ordinary differential equation (ODE) solvers. In this work, a linearly implicit/explicit (IMEX) Runge–Kutta scheme is used, that is suitable for stiff problems (Ascher, Ruuth, & Spiteri, 1997). For any solution, the fitness value is then calculated after the simulation, taking the calculated values of the state variables, according to the PI defined for each case. The overall process of decoding and evaluating a solution is exemplified in Fig. 1.

3.2. Differential Evolution

Differential Evolution (DE) is a population-based approach to function optimization that generates trial individuals by calculating vector differences between other randomly selected members of the population. Given a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ to be minimised, a DE algorithm starts by randomly generating a population of p n-dimensional vectors. These vectors (or individuals) will evolve over the course of the algorithm's run until a termination criterion is met. The simplest termination condition, used in this work, is defined by setting a fixed maximum number of function evaluations.

There are several variants of DE that vary according to the scheme used in creating new candidate solutions. The following is an outline of the structure of the DE algorithm used in this work (Storn & Price, 1996) that is common to all variants. The scheme used for the computation of the candidate solutions, that defines each variant, will been shown separately below.

1. Initialize the population.
2. Evaluate the population.
3. Generate a new population where for each individual $\vec{x}_i$, $i = 1 \ldots p$ in parallel the following steps occur:
   (i) Generate a trial vector $\vec{t}_i$ according to one of the possible schemes (see below).
   (ii) Generate a candidate individual by performing crossover between $\vec{t}_i$ and $\vec{x}_i$, with probability CR at each position (at least one position of the trial vector must be used).
   (iii) Evaluate the candidate.
   (iv) Replace the current individual by the candidate if the candidate is at least as good.
4. Loop to 3 unless the termination criterion is met.

Various schemes are currently in use for creating candidate solutions in DE (Storn, 1996). Four schemes are considered in this paper. These are shown below along with the corresponding trial vector generation formula:

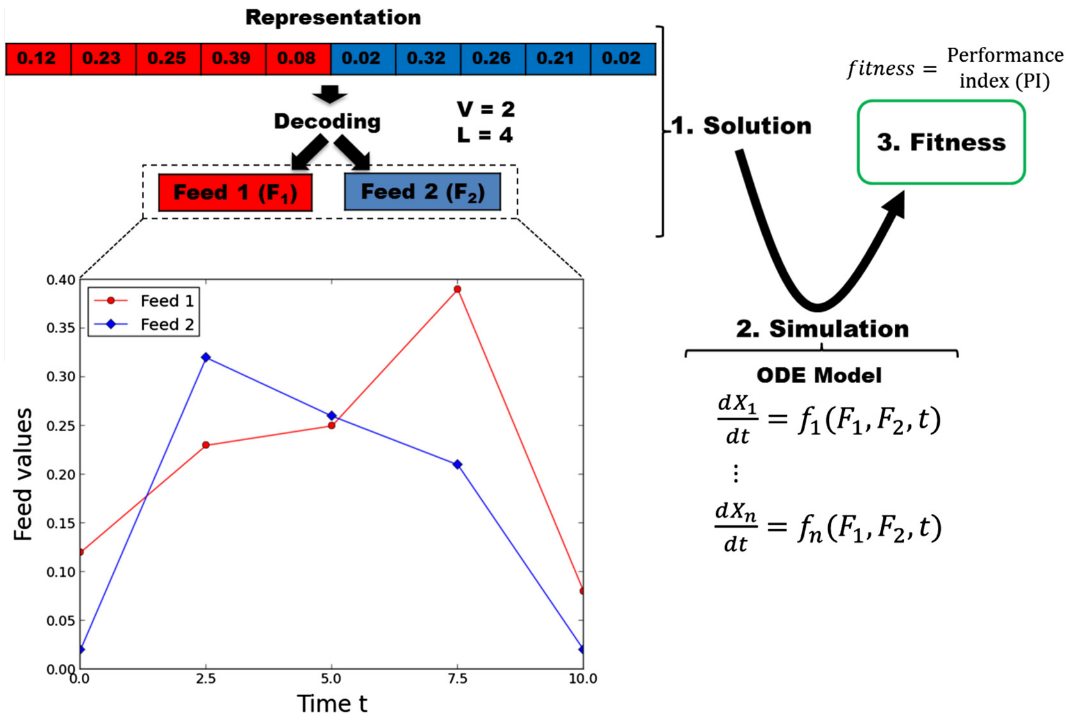

Fig. 1. An illustration of the process of decoding and evaluating a solution (1) where the number of input variables is $V = 2$ (i.e. there are two feeds), each curve has length $L = 4$ (i.e., it has 4 line segments) and the final time is $t_f = 10$. The first $L + 1$ values of the solution correspond to the first input variable (Feed 1) while the remaining ones correspond to the second input variable (Feed 2). The feed values are used by a numerical simulation (2) and its corresponding performance index is used to evaluate the solution's fitness (3).

DE/rand/1 $\vec{t}_i = \vec{x}_{r_1} + Z(\vec{x}_{r_2} - \vec{x}_{r_3})$
DE/rand/2 $\vec{t}_i = \vec{x}_{r_1} + Z(\vec{x}_{r_2} + \vec{x}_{r_3} - \vec{x}_{r_4} - \vec{x}_{r_5})$
DE/best/1 $\vec{t}_i = \vec{x}_{best} + Z(\vec{x}_{r_2} - \vec{x}_{r_3})$
DE/best/2 $\vec{t}_i = \vec{x}_{best} + Z(\vec{x}_{r_2} + \vec{x}_{r_3} - \vec{x}_{r_4} - \vec{x}_{r_5})$

where the variables $\vec{x}_{r_j}$, $1 \leqslant j \leqslant 5$ represent distinct randomly selected individuals that are different from the current individual $\vec{x}_i$ and $\vec{x}_{best}$ is the best individual in the population. The parameter Z is usually set between 0 and 2 and denotes the scale of the difference vectors – a difference vector is the difference between two individuals (e.g. $\vec{x}_{r_2} - \vec{x}_{r_3}$). The schemes vary in the anchor – the solution to which the difference vectors are added – that can be a random individual or the best individual in the population. The number of difference vectors is another parameter.

DE is by nature a greedy algorithm, since a new solution is only accepted if it is at least as good as the one it replaces in the population. The DE/best schemes are even greedier since they use the best individual in the population as the anchor, while the DE/rand schemes use a randomly selected individual in the population. The number of difference vectors used is related to the randomness of the scheme: a larger value means the degree of randomness is increased.

3.3. Real-valued evolutionary algorithm

Evolutionary Algorithms (EAs) (Michalewicz, 1996) are also population based algorithms that evolve solutions to a given problem encoded in an individual. A real-valued EA was considered in this work, since it provided good results in previous work in this task (Rocha et al., 2004; Rocha, Rocha, & Ferreira, 2005). In this case, the chromosomes are vectors of real numbers as in DE.

The overall structure of the EA is given by:

1. Initialize time ($t = 0$), generate and evaluate the initial population (P_0).
2. While the termination criterion is not met:
   (i) Select from P_t individuals for reproduction.
   (ii) Apply the genetic operators to breed the offspring and evaluate them.
   (iii) Insert the offspring into the next population (P_{t+1}).
   (iv) Select the survivors from P_t to be kept in P_{t+1}.
   (v) Increase current time ($t = t + 1$).

Regarding the reproduction step, this EA uses the following mutation and crossover operators to create new solutions (Michalewicz, 1996):

- *Random Mutation*, which replaces one value at a given random position i by a new randomly generated value within $[min_i, max_i]$, the range of values allowed for position i;
- *Gaussian Mutation*, which adds to a given value at a random position i a value taken from a Gaussian distribution, with a zero mean and a standard deviation given by $\frac{max_i - min_i}{4}$ (i.e., small perturbations will be preferred over larger ones);
- *Two-Point crossover*, a standard Genetic Algorithm operator;
- *Arithmetical crossover*, where each value in the offspring will be a linear combination of the values in the ancestors' chromosomes.

Both mutation operators are applied to a variable number of positions in the vector (a value that is randomly set between 1 and 10 in each application). In previous work, the best result was obtained using an alternative that contemplates the use of all genetic operators described above (Rocha et al., 2004). All operators are used with equal probabilities to breed the offspring.

The selection procedure is done by converting the fitness value into a linear ranking in the population, and then applying a roulette wheel scheme. In each generation, half of the individuals are kept from the previous generation, and the remaining are bred by the application of the reproduction operators.

The EAs use a much lower selective pressure than DE or FIPS (described in the next section) as new solutions can have a worse quality than the ones that are being replaced. This feature makes them more suitable to explore the search space, but implies they are slower to converge.

3.4. Fully Informed Particle Swarm (FIPS)

A particle swarm optimizer (PSO) uses a population of particles, whose coordinates in the space encode the solutions to the problem, and that evolve over time by moving through the search space. Particles imitate their neighbors by approaching their best positions. In the canonical particle swarm, the two sources of imitation are the individual's previous best position and the best position found by the most successful neighbor.

Due to the fact that in previous studies the FIPS (Mendes, Kennedy, & Neves, 2004) clearly outperformed the canonical particle swarm, this method will be used. In this case, each particle is defined by: $P_t^{(i)} = \langle x_t, v_t, p_t, e_t \rangle$, where $x_t \in \mathbb{R}^d$ is the current position in the search space; $p_t \in \mathbb{R}^d$ is the position visited by the particle in the past that had the best function evaluation; $v_t \in \mathbb{R}^d$ is a vector that represents the direction in which the particle is moving, it is called the 'velocity'; e_t is the evaluation of p_t under the function $f : \mathbb{R}^d \to \mathbb{R}$ being optimized, i.e. $e_t = f(p_t)$. Particles are connected to others in the population via a predefined topology. This can be represented by the adjacency matrix of a directed graph $M = (m_{ij})$, where $m_{ij} = 1$ if there is an edge from particle i to particle j and $m_{ij} = 0$ otherwise.

In FIPS, each particle moves in the direction of the stochastic barycenter of the previous best position of all the neighboring particles (excluding the particle itself). As in the canonical particle swarm, the neighbors of a particle are the ones that share a vertex in the graph that represents the topology. The following is an outline of a generic PSO:

1. Set the iteration counter, $t = 0$.
2. Initialize each $x_0^{(i)}$ and $v_0^{(i)}$ randomly. Set $p_0^{(i)} = x_0^{(i)}$.
3. Evaluate each particle and set $e_0^{(i)} = f(p_0^{(i)})$.
4. Let $t = t + 1$ and generate a new population, where each particle i is moved to a new position in the search space according to:

   (i) $v_t^{(i)} = velocity\_update(v_{t-1}^{(i)})$.
   (ii) $x_t^{(i)} = x_{t-1}^{(i)} + v_t^{(i)}$.
   (iii) Evaluate the new position, $e = f(x_t^{(i)})$.
   (iv) If the new position is better than the previous best, update the particle's previous best position. i.e if $e < e_{t-1}^{(i)}$ then let $p_t^{(i)} = x_t^{(i)}$ and $e_t^{(i)} = e$ else let $p_t^{(i)} = p_{t-1}^{(i)}$ and $e_t^{(i)} = e_{t-1}^{(i)}$.
   (v) Loop to 4 until the termination criterion is met.

Clerc et al. (2002) introduced the use of a factor called the 'constriction factor', symbolized by χ, into the velocity update equation. The velocity update equation for FIPS is then given by:

$$velocity\_update(v_t^{(i)}) = \chi\left(v_{t-1}^{(i)} + \sum_{j \in N(i)} U(0,1) \cdot \frac{\varphi}{|N(i)|} \cdot (p_{t-1}^{(j)} - x_{t-1}^{(i)})\right) \tag{40}$$

where U is the generator of pseudo-random numbers following the uniform distribution, $\varphi = 4.1$, $\chi = 0.729$, $N(i)$ is the neighborhood (the set of the particles) of particle i. In this study, the population is organized according to the *von Neumann* topology (Kennedy & Mendes, 2002), where each particle is connected to four others, in a torus configuration.

4. Experiments and results

In this section, a number of experiments and their respective results are presented, aiming at the evaluation of the different algorithms, in the selected case studies. These results will be discussed in detail.

4.1. Methodology

The results reported in this text are the means of 30 runs and are presented with 95% confidence intervals. Additionally, the use of t-tests Goulden, 1956 for two-sample comparisons was adopted. Given that multiple pairwise comparisons were performed, the authors used the *Holm* correction for the p-values (Holm, 1979).

In order to improve the readability of the analysis, a symbolic encoding of the p-values resulting from the t-tests was used. To allow a straightforward comparison between the algorithms, different symbols are used to report whether the mean of algorithm $A1$ is greater than the mean of $A2$ or vice versa. The encoding used is presented in Table 1.

Sometimes statistical tests cannot find a significant difference between two algorithms (e.g., because the confidence interval of one of them is too wide). Nonetheless, we are interested in a *reliable* method: one that consistently yields good results. Thus, an algorithm with a good average and a narrow confidence interval is preferred in these cases.

4.2. Parameter settings and test conditions

When solving a real world optimization problem, the main concern is to have a tool that may be applied to the problem with as few fine-tuning as possible. The main focus of this work will be in the results and not in a thorough study about the parameterization of the algorithms involved. It was not an aim of this work to go through the cumbersome task of testing the valuation of all the parameters of these algorithms until a suitable setting for the problem at hand could be found. Furthermore, this would be quite difficult to achieve in practice given the computational effort required. Thus, it was decided to use standard configurations for each algorithm that were either validated by preliminary experimental results or suggested by previous studies.

Due to the previous experience of the authors with the real-valued EA, each run was stopped after 200 kFEs (thousands of function evaluations). In the case of FIPS, the population size was 20 and the other parameters have the usual values given in the literature. For all DE variants, the population size was set to 20, F was set to 0.5 and CR to 0.6. In terms of the real-valued EA, the population size was set to 200. The value of L was determined, for each

Table 1
Encoding used in the presentation of p-values of the pairwise t-tests comparing results from algorithms $A1$ and $A2$.

p-Value	Condition	Symbol
$p \leqslant 0.001$	$mean(A1) > mean(A2)$	+++
$p \leqslant 0.001$	$mean(A1) < mean(A2)$	- - -
$0.001 < p \leqslant 0.01$	$mean(A1) > mean(A2)$	++
$0.001 < p \leqslant 0.01$	$mean(A1) < mean(A2)$	- -
$0.01 < p \leqslant 0.05$	$mean(A1) > mean(A2)$	+
$0.01 < p \leqslant 0.05$	$mean(A1) < mean(A2)$	-
$p \geqslant 0.05$		O

Table 2
Results for case study I: mean and confidence intervals of the PI.

Algorithm	PI 50 kFEs	PI 100 kFEs	PI 200 kFEs
DE/rand/1	9.4726 ± 0.0005	9.4727 ± 0.0005	9.4727 ± 0.0003
DE/rand/2	9.0669 ± 0.0390	9.4074 ± 0.0102	9.4728 ± 0.0001
DE/best/1	5.1580 ± 0.4795	5.2274 ± 0.4470	5.2315 ± 0.4443
DE/best/2	9.4423 ± 0.0626	9.4729 ± 0.0000	9.4729 ± 0.0000
EA	8.4762 ± 0.0731	8.7891 ± 0.0613	9.0037 ± 0.0497
FIPS	9.4716 ± 0.0014	9.4729 ± 0.0000	9.4729 ± 0.0000

Table 3
Pairwise t-test with the Holm *p*-value adjustment for the algorithms of case study I.

	DE/rand/1	DE/rand/2	DE/best/1	DE/best/2	EA	FIPS
DE/rand/1		O	+++	O	+++	O
DE/rand/2	O		+++	- - -	+++	- - -
DE/best/1	- - -	- - -		- - -	- - -	- - -
DE/best/2	O	+++	+++		+++	O
EA	- - -	- - -	+++	- - -		- - -
FIPS	O	+++	+++	O	+++	

Table 4
Results for case study II: mean and confidence intervals of the PI.

Algorithm	PI 50 kFEs	PI 100 kFEs	PI 200 kFEs
DE/rand/1	20386 ± 7	20400 ± 7	20409 ± 6
DE/rand/2	20348 ± 8	20366 ± 6	20382 ± 6
DE/best/1	19702 ± 128	19723 ± 128	19751 ± 134
DE/best/2	20229 ± 86	20263 ± 80	20281 ± 84
EA	20119 ± 48	20280 ± 35	20373 ± 17
FIPS	19821 ± 120	19822 ± 120	19822 ± 120

Table 5
Pairwise t-test with the Holm *p*-value adjustment for the algorithms of case study II.

	DE/rand/1	DE/rand/2	DE/best/1	DE/best/2	EA	FIPS
DE/rand/1		+++	+++	O	++	+++
DE/rand/2	- - -		+++	O	O	+++
DE/best/1	- - -	- - -		- - -	- - -	O
DE/best/2	O	O	+++		O	+++
EA	- -	O	+++	O		+++
FIPS	- - -	- - -	O	- - -	- - -	

Table 6
Results for case study III: mean and confidence intervals of the PI.

Algorithm	PI 50 kFEs	PI 100 kFEs	PI 200 kFEs
DE/rand/1	392.81 ± 3.81	393.93 ± 3.20	394.99 ± 3.13
DE/rand/2	391.66 ± 0.48	394.18 ± 0.33	395.73 ± 0.20
DE/best/1	276.40 ± 10.74	283.50 ± 12.25	289.37 ± 12.82
DE/best/2	372.90 ± 12.44	375.08 ± 12.60	378.67 ± 11.86
EA	374.83 ± 1.67	382.49 ± 0.86	387.62 ± 0.52
FIPS	362.45 ± 15.10	370.66 ± 13.68	375.69 ± 10.79

Table 7
Pairwise t-test with the Holm *p*-value adjustment for the algorithms of case study III.

	DE/rand/1	DE/rand/2	DE/best/1	DE/best/2	EA	FIPS
DE/rand/1		O	+++	O	++	+
DE/rand/2	O		+++	O	+++	++
DE/best/1	- - -	- - -		- -	- - -	- - -
DE/best/2	O	O	++		O	O
EA	- -	- - -	+++	O		O
FIPS	-	- -	+++	O	O	

Table 8
Results for case study IV: mean and confidence intervals of the PI.

Algorithm	PI 50 kFEs	PI 100 kFEs	PI 200 kFEs
DE/rand/1	0.8143 ± 0.0036	0.8147 ± 0.0035	0.8147 ± 0.0035
DE/rand/2	0.8165 ± 0.0000	0.8165 ± 0.0000	0.8165 ± 0.0000
DE/best/1	0.3988 ± 0.1746	0.3988 ± 0.1746	0.3988 ± 0.1746
DE/best/2	0.8101 ± 0.0062	0.8101 ± 0.0062	0.8101 ± 0.0062
EA	0.8158 ± 0.0002	0.8162 ± 0.0001	0.8164 ± 0.0000
FIPS	0.8165 ± 0.0000	0.8165 ± 0.0000	0.8165 ± 0.0000

case study, based on preliminary results, and was set to the value of t_f in case studies I and II and to the value of $2t_f$ in case studies III and IV.

4.3. Results

For all case studies, the results will be shown in two distinct tables. The first one will present the results obtained by each of the algorithms showing the mean and the 95% confidence intervals for the PI. These will be shown for three distinct steps of the optimization process: when 50, 100 and 200 kFEs were performed by each algorithm. It was decided to probe PI at these time-steps to estimate the possibility of terminating the runs earlier whilst still maintaining good quality solutions.

The second set of tables will help to further validate the results, showing the pairwise t-test results (at 200 kFEs), using the methodology aforementioned. This will help to clarify the statistical differences among the algorithms. In these tables, the algorithm that appears on each row will correspond to A1 on Table 1 and the algorithm given by the column to A2.

Tables 2–7, finally, 8 and 9, present the results obtained by each of the algorithms on the case studies I–IV, respectively.

4.4. Discussion

Overall, the DE/rand variants are able to obtain the best results in all case studies, presenting the highest means and showing quite narrow confidence intervals, alternating as the best algorithm (in some cases there are other algorithms that show differences that are not statistically significant).

The DE/best variants do not show such a good behaviour. Indeed, DE/best/1 is always outperformed by all the other contenders, being stuck in local optima at a very premature stage. On the other hand, the DE/best/2 alternative behaves much better than DE/best/1, but still does not compare well with the DE/rand variants (the exception is case study I).

FIPS has a good behavior in two out of the four case studies (I and IV), where it is at the level of DE/rand variants. In the remaining ones, the performance is worse, since it gets stuck prematurely in local optima. Therefore, it does not present a good reliability for this task, when compared to DE/rand.

Finally, the real-valued EA behaves relatively well in most problems, although it is consistently outperformed by DE/rand. Nevertheless, it clearly outperforms FIPS and DE/best/2 in two cases (II and III), that seem to be the most challenging. Case study I seems to favor the use of greedier algorithms, such as DE/rand and FIPS and the EA shows here its worst result.

The previous analysis only took into consideration the final results (at 200 kFEs). Regarding the intermediate results, the DE variants and FIPS both get good results before 50 kFEs and do not improve significantly in the remaining time. On the other hand, the EA converges slower but uses all the available time to improve the solutions. It is not hard to believe that if more time is available the solutions could still improve. However, since the problems are quite demanding in terms of computational resources, methods

Table 9
Pairwise t-test with the Holm *p*-value adjustment for the algorithms of case study IV.

	DE/rand/1	DE/rand/2	DE/best/1	DE/best/2	EA	
De/rand/1		O	+++	O	O	O
DE/rand/2	O		+++	O	+++	O
DE/best/1	- - -	- - -		- - -	- - -	- - -
DE/best/2	O	O	+++		O	O
EA	O	- - -	+++	O		- - -
FIPS	O	O	+++	O	+++	

that converge fast are more interesting and the DE confirms its merits also in this regard.

To further explore this issue, the average of the best fitness in the population for each algorithm was plotted against the number of function evaluations. Fig. 2 presents the typical convergence curve of the algorithms (in case study III). Similar graphs for the other case studies are given as additional material.

A global analysis of the results shows that algorithms that are too greedy (i.e. DE/best/1) have serious problems in this task. The reason why DE/best/2 is not as easily misled by local optima lies in the higher degree of randomness. The trade-off between randomness and greediness seems to be near optimal for this task in the DE/rand variants, that show a good balance between the exploration and exploitation of the search space. The fact that the EA favors a higher degree of exploration, due to the decreased selective pressure, limits its convergence speed and impairs its performance.

5. OptFerm, an optimization supporting tool for fermentation engineering

In this section, the OptFerm application will be presented, whose main aim is to make available to the community a set of computational tools that will allow its users to take advantage of the previously described algorithms for fed-batch fermentation optimization, together with complementary features at the level of the fermentation process simulation and parameter fitting. Also, as an important driver behind the development of this tool, there is the lack of free computational tools for fermentation engineering.

The OptFerm platform was developed using the Java programming language, with the aim of being a user-friendly, open-source,

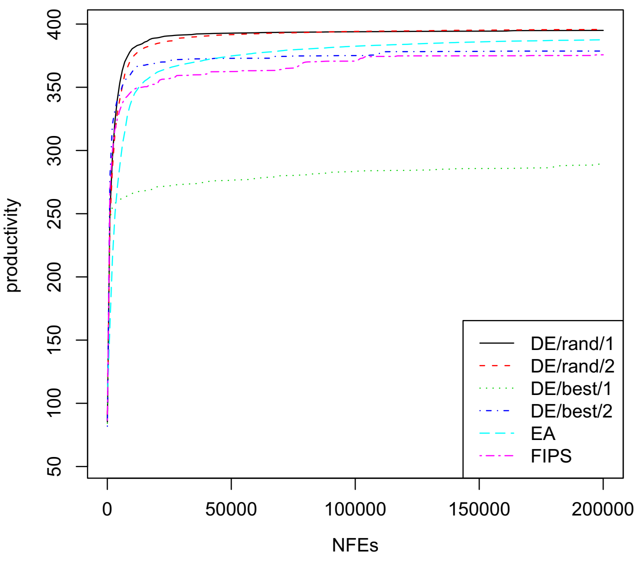

Fig. 2. Convergence of the algorithms for case study III.

extensible and platform-independent tool. It was designed to allow its users to evaluate and compare several different methods for the tasks of simulation, optimization and parameter estimation, in the context of fermentation processes. The goal is to allow users to improve process productivity, achieving better results in reduced times. The major features of this tool are listed in the following topics:

Model representation: The model is given by a set of ODEs and a set of kinetic reactions representing the problem dynamics. All information related to these dynamic and kinetic equations such as names, initial values, upper and lower limits of the state variables and kinetic parameters must be described in the model file. Moreover, it is necessary to define one or more objective functions to optimize. After importing the model, users may create new configuration sets where they specify values for the initial state variables and model parameters whose values may differ from those that were initially defined in the model. These new configuration datasets will be added automatically to the OptFerm clipboard, where they are kept for future use.

Simulation: Users may test several combinations of initial values for state variables, parameters and experimental or hypothetical feeding trajectories. In addition, the results obtained in optimization and parameter estimation tasks are immediately accessible in the simulation datasets panel, allowing the user to perform their analysis in future simulations. The simulation results are presented in a graph, in which, the user can visualize all (simultaneously) or each (separately) state variables or kinetic rates. Moreover, it is possible to perform the comparison between simulated data and experimental datasets.

Optimization: Three different types of operations can be performed:

1. the optimization of a feeding trajectory, where the ideal amount of substrate to be fed into the reactor per time unit is determined;
2. the optimization of feeding trajectory plus final time, where besides the determination of the best feeding profile an optimal duration of the fermentation is also provided;
3. the optimization of feeding trajectory plus initial conditions, which allows to simultaneously determine the feeding trajectory and the best initial concentrations for each selected state variable. The visualization of the optimization results is accessible through a graphical interface, that shows the feed profile along time in a graph and the obtained vector of feed values vs time-points. The optimization algorithms available include the ones with best results, namely DE/rand and EAs.

Parameter estimation: It is possible to the estimate model parameter values by fitting the simulated data to experimental data. This operation is performed by minimizing a total cost function that represents the prediction error between experimental and simulated data:

$$TotalCost = \sum_{i=1}^{n}\left(\frac{1}{N_p}\sum_{j=1}^{p}\left(\frac{\xi_{ij}^{(sim)} - \xi_{ij}^{(exp)}}{\xi_{ij}^{(exp)}}\right)^2\right) \tag{41}$$

where $\xi_{ij}^{(sim)}$ represents simulated data and $\xi_{ij}^{(exp)}$ the experimental data for the state variable ξ (n is the number of state variables) for every point (p is the total of data points). An user may choose which model parameters to estimate by providing a predefined interval for some of the parameters, while the remaining parameters remain fixed.

The implementation of OptFerm was performed following a modular approach, using AIBench (Glez-Peña et al., 2010) (a Model-View-Control based Java application) as the basis platform.

Several functions were implemented to provide users with the aforementioned operations through graphical interfaces. A module containing specific optimization routines was created for feed optimization and related tasks. This module uses JECoLi (Evangelista, Maia, & Rocha, 2009) that contains generic optimization routines based on metaheuristic search algorithms. Some adaptations had to be made in order to use these algorithms to support feed optimizations (Rocha et al., 2004; Mendes, Rocha, Ferreira, & Rocha, 2006). To perform numerical simulation the *ODEToJava* package (Ascher et al., 1997) was integrated into the system. This package contains several methods such as the explicit Runge–Kutta scheme and the linearly implicit-explicit Runge–Kutta scheme in order to address non-stiff and stiff problems for ODEs. The OptFerm framework is freely available at http://darwin.di.uminho.pt/optferm.

6. Conclusions and further work

This paper proposes the use of several bio-inspired algorithms, namely Evolutionary Algorithms (EAs), Differential Evolution (DE) and Particle Swarm Optimization (PSO) applied to the optimization of the feeding trajectory in fed-batch fermentation processes. The main contribution of the work relies on conducting a thorough and statistically validated comparison between these approaches. Four distinct case studies are used to compare the algorithms, using the means and confidence intervals of the objective functions over 30 runs and statistical tests to validate the results. Furthermore, the convergence speed of each algorithm was also evaluated.

The best overall algorithms in these tasks were the DE/rand variants, that consistently obtained good results in all the case studies and presented a good convergence speed. If an algorithm is chosen, the DE/rand/1 would be the one, since it represents the simpler alternative to implement and obtains good results. FIPS was a good contender in two of the cases where it found good results and was as fast as DE/rand. However, it got stuck on local optima in the other two case studies. EA was slower to converge, but reliable. If you can afford the computational time needed, it finds good solutions. An explanation of the results can be offered by analysing the selective pressure of the approaches. The good balance between exploration and exploitation of the search space shown by DE/rand algorithms enables them to achieve consistent high quality results. Greedier schemes are frequently stuck in local optima, while the more conservative approach of the EA slows the convergence, increasing the computational time needed to reach good solutions.

A complementary contribution of this work relied in the development of OptFerm, a computational framework enabling the practical application of the methods proposed. This user friendly open-source software allows its users to use a number of tools for the simulation, optimization and parameter estimation of fermentation processes, thus bridging a gap existent between optimization algorithms and its application by Biotechnology researchers that find in OptFerm a valuable tool for their daily work.

Previous work by the authors (Rocha et al., 2005) developed a new representation in EAs in order to allow the optimization of a time trajectory with automatic interpolation. It would be interesting to develop a similar approach within DE and implement this into OptFerm. Another area of future research is the consideration of online adaptation, where the model of the process is updated during the fermentation process. In this case, the good computational performance of DE is a benefit, if there is the need to re-optimize the feeding trajectories in real-time given values for the state variables that are measured online from the real fermentation process.

Acknowledgments

The work is partially funded by ERDF – European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT (Portuguese Foundation for Science and Technology) within projects Ref. COMPETE FCOMP-01-0124-FEDER-015079 and PEst-OE/EEI/UI0752/2011.

References

Ascher Ruuth & Spiteri (1997). Implicit–explicit Runge–Kutta methods for time-dependent partial differential equations. *Applied Numerical Mathematics, 25*, 151–167.

Banga, J., Moles, C., & Alonso, A. (2003). Global optimization of bioprocesses using stochastic and hybrid methods. *Frontiers in global optimization – nonconvex optimization and its applications* (Vol. 74, pp. 45–70). Kluwer Academic Publishers.

Bapat, P., Sohoni, S., Moses, T., & Wangikar, P. (2006). Model-based optimization of feeding recipe for rifamycin fermentation. *AIChE Journal, 52*, 4248–4257.

Chen, C., & Hwang, C. (1990). Optimal control computation for differential-algebraic process systems with general constraints. *Chemical Engineering Communications, 97*, 9–26.

Chen, L., Nguang, S., Chen, X., & Li, X. (2004). Modelling and optimization of fed-batch fermentation processes using dynamic neural networks and genetic algorithms. *Biochemical Engineering Journal, 22*, 51–61.

Chiou, J., & Wang, F. (1999). Hybrid method of evolutionary algorithms for static and dynamic optimization problems with application to a fed-batch fermentation process. *Computers & Chemical Engineering, 23*, 1277–1291.

Clerc, M., & Kennedy, J. (2002). The particle swarm - explosion, stability and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation, 6*(1), 58–73.

Dragoi, E. N., Curteanu, S., Galaction, A. I., & Cascaval, D. (2013). Optimization methodology based on neural networks and self-adaptive differential evolution algorithm applied to an aerobic fermentation process. *Applied Soft Computing, 13*, 222–238.

Evangelista, P., Maia, P., & Rocha, M. (2009). Implementing metaheuristic optimization algorithms with JE. coli. In *Ninth international conference on intelligent systems design and applications* (pp. 505–510). IEEE.

Franco-Lara, E., & Weuster-Botz, D. (2005). Estimation of optimal feeding strategies for fed-batch bioprocesses. *Bioprocess and Biosystems Engineering, 27*, 255–262.

Glez-Peña, D., Reboiro-Jato, M., Maia, P., Rocha, M., Díaz, F., Fdez-Riverola, F., et al. . *Computer Methods and Programs in Biomedicine, 98*(2), 191–203.

Goulden, C. H. (1956). *Methods of statistical analysis* (2nd ed.). John Wiley & Sons Ltd.,.

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics, 6*, 65–70.

Jayaraman, V., Kulkarni, B., Gupta, K., Rajesh, J., & Kusumaker, H. (2001). Dynamic optimization of fed-batch bioreactors using the ant algorithm. *Biotechnology Progress, 17*, 81–88.

Kapadi, M., & Gudi, R. (2004). Optimal control of fed-batch fermentation involving multiple feeds using differential evolution. *Process Biochemistry, 39*, 1709–1721.

Kennedy, J., & Mendes, R. (2002). Population structure and particle swarm performance. In *Proceedings of the fourth congress on evolutionary computation (CEC-2002)* (pp. 1671–1676). Honolulu, Hawaii: IEEE Computer Society.

Kookos, I. (2004). Dynamic optimization of fed-batch bioreactors using the ant algorithm. *Biotechnology Progress, 20*, 1285–1288.

Lee, J., & Ramirez, W. (1994). Optimal fed-batch control of induced foreign protein production by recombinant bacteria. *AIChE Journal, 40*, 899–907.

Levisauskas, D., Galvanauskas, V., Henrich, S., Wilhelm, K., Volk, N., & Lübbert, A. (2003). Model-based optimization of viral capsid protein production in fed-batch culture of recombinant escherichia coli. *Bioprocess and Biosystems Engineering, 25*, 255–262.

Mendes, R., Rocha, I., Ferreira, E. C., & Rocha, M. (2006). A comparison of algorithms for the optimization of fermentation processes. In *IEEE Conference on Evolutionary Computation* (pp. 7371–7378).

Mendes, R., Kennedy, J., & Neves, J. (2004). The fully informed particle swarm: Simple, maybe better. *IEEE Transactions on Evolutionary Computation, 8*(3), 204–210.

Michalewicz, Z. (1996). *Genetic algorithms + data structures = evolution programs* (3rd ed.). USA: Springer-Verlag.

Modak, J., Lim, H. C., & Tayeb, Y. (1986). General characteristics of optimal feed rate profiles for various fed-batch fermentation processes. *Biotechnology and Bioengineering, 28*, 1396–1407.

Moonchai, S., Madlhoo, W., Jariyachavalit, K., & Shimizu, H. (2005). Application of a mathematical model and differential evolution algorithm approach to optimization of bacteriocin production by *Lactococcus lactis* c7. *Bioprocess and Biosystems Engineering, 28*, 15–26.

Na, J., Chang, Y., Chung, B., & Lim, H. (2002). Adaptive optimization of fed-batch culture of yeast by using genetic algorithms. *Bioprocess and Biosystems Engineering, 24*, 299–308.

Nguang, S., Chen, L., & Chen, X. (2001). Optimisation of fed-batch culture of hybridoma cells using genetic algorithms. *ISA Transactions, 40*, 381–389.

Park, S., & Ramirez, W. (1988). Optimal production of secreted protein in fed-batch reactors. *AIChE Journal, 34*(9), 1550–1558.

Patil, K., Rocha, I., Forster, J., & Nielsen, J. (2005). Evolutionary programming as a platform for in silico metabolic engineering. *BMC Bioinformatics, 6*(308), 1–12.

Rocha, I. (2003). Model-based strategies for computer-aided operation of recombinant e. coli fermentation. Ph.D. thesis, Universidade do Minho. URL <http://hdl.handle.net/1822/1269>.

Rocha, I., & Ferreira, E. (2002). On-line simultaneous monitoring of glucose and acetate with fia during high cell density fermentation of recombinant *E. coli*. *Analytica Chimica Acta, 462*(2), 293–304.

Rocha, M., Neves, J., Rocha, I., & Ferreira, E. (2004). Evolutionary algorithms for optimal control in fed-batch fermentation processes. *Lecture Notes in Computer Science, 3005*, 84–93.

Rocha, M., Rocha, I., & Ferreira, E. (2005). A new representation in evolutionary algorithms for the optimization of bioprocesses. In *Proceedings of the IEEE congress on evolutionary computation* (pp. 484–490). IEEE Press.

Ronen, M., Shabtai, Y., & Guterman, H. (2002). Optimization of feeding profile for a fed-batch bioreactor by an evolutionary algorithm. *Journal of Biotechnology, 97*, 253–263.

Ros, S. D., Colusso, G., Weschenfelder, T. A., de Marsillac Terra, L., de Castilhos, F., Corazza, M. L., et al. (2013). A comparison among stochastic optimization algorithms for parameter estimation of biochemical kinetic models. *Applied Soft Computing, 13*, 2205–2214.

Roubos, J., van Straten, G., & van Boxtel, A. (1999). An evolutionary strategy for fed-batch bioreactor optimization: Concepts and performance. *Journal of Biotechnology, 67*, 173–187.

Sarkar, D., & Modak, J. (2003). Optimisation of fed-batch bioreactors using genetic algorithms. *Chemical Engineering Science, 58*, 2283–2296.

Sarkar, D., & Modak, J. (2004). Optimization of fed-batch bioreactors using genetic algorithm: Multiple control variables. *Computers & Chemical Engineering, 28*, 789–798.

Shin, H., & Lim, H. (2006). Cell-mass maximization in fed-batch cultures – sufficient conditions for singular arc and optimal feed rate profiles. *Bioprocess and Biosystems Engineering, 29*, 335–347.

Shukla, P., & Pushpavanam, S. (1998). Optimisation of biochemical reactors – an analysis of different approximations of fed-batch operation. *Chemical Engineering Science, 53*, 341–352.

Smets, I., Claes, J., November, E., Bastin, G., & Impe, J. V. (2004). Optimal adaptive control of (bio)chemical reactors: Past, present and future. *Journal of Process Control, 14*, 795–805.

Storn, R. (1996). On the usage of differential evolution for function optimization. In *1996 Biennial conference of the North American fuzzy information processing society (NAFIPS 1996)* (pp. 519–523). IEEE.

Storn, R., & Price, K. (1996). Minimizing the real functions of the icec'96 contest by differential evolution. In *IEEE international conference on evolutionary computation* (pp. 842–844). IEEE.

Tartakovsky, B., Ulitzur, S., & Sheintuch, M. (1995). Optimal control of fed-batch fermentation with autoinduction of metabolite production. *Biotechnology Progress, 11*, 80–87.

Wang, F., & Cheng, W. (1999). Simultaneous optimization of feeding rate and operation parameters for fed-batch fermentation processes. *Biotechnology Progress, 15*, 949–952.

Zuo, K., & Wu, W. (2000). Semi-realtime optimization and control of a fed-batch fermentation system. *Computers and Chemical Engineering, 24*, 1105–1109.