Differential Evolution for the Offline and Online Optimization of Fed-Batch Fermentation Processes

Rui Mendes[1], Isabel Rocha[2], José P. Pinto[1], Eugénio C. Ferreira[2], and Miguel Rocha[1]

[1] Departament of Informatics / CCTC - University of Minho
Campus de Gualtar, 4710-057 Braga - Portugal
azuki@di.uminho.pt, mrocha@di.uminho.pt
[2] IBB - Institute for Biotechnology and Bioengineering
Centre of Biological Engineering - University of Minho
Campus de Gualtar, 4710-057 Braga - Portugal
irocha@deb.uminho.pt, ecferreira@deb.uminho.pt

Summary. The optimization of input variables (typically feeding trajectories over time) in fed-batch fermentations has gained special attention, given the economic impact and the complexity of the problem. Evolutionary Computation (EC) has been a source of algorithms that have shown good performance in this task. In this chapter, Differential Evolution (DE) is proposed to tackle this problem and quite promising results are shown. DE is tested in several real world case studies and compared with other EC algorihtms, such as Evolutionary Algorithms and Particle Swarms. Furthermore, DE is also proposed as an alternative to perform online optimization, where the input variables are adjusted while the real fermentation process is ongoing. In this case, a changing landscape is optimized, therefore making the task of the algorithms more difficult. However, that fact does not impair the performance of the DE and confirms its good behaviour.

1 Introduction

In recent years, many efforts have been devoted to the optimization of processes in biotechnology and bioengineering, since a number of valuable products such as recombinant proteins, antibiotics and amino-acids are produced using fermentation techniques.

A problem that has received major attention is the dynamic optimization of fed-batch bioreactors, which has traditionally been done on the substrate feed rates as key manipulated variables. The optimization problem is usually solved before the beginning of the fermentation process (open-loop optimal control) and may consist on finding an expression or a sequence of values for the feeding rate, that maximize a given objective function. This function will typically be a performance index that measures the process productivity, subject to the constraints represented by a dynamical model.

U.K. Chakraborty (Ed.): Advances in Differential Evolution, SCI 143, pp. 299–317, 2008.
springerlink.com

Several optimization methods have been applied to solve this class of problems. It has been shown that for relatively simple bioreactor systems, which are expressed as differential equation models, the optimization problem can be solved analytically from the Hamiltonian function, by applying the Minimum Principle of Pontryagin [3]. However, in the majority of the cases reported, determination of the optimal feed rate profile has a problem of singular control, because the control variable (feed rate) often appears linearly in the system of differential equations. Thus, this approach fails to provide a complete solution. Also, those methodologies become too complex when the number of state and control variables increase.

Numerical methods make a distinct approach to dynamic optimization. The gradient algorithms are used to adjust the control trajectories in order to iteratively improve the objective function [4]. In contrast, dynamic programming methods discretize both time and control variables to a predefined number of values. A systematic backward search method in combination with the simulation of the system model equations is then used to find the optimal path through the defined grid. However, in order to achieve a global minimum, the computational burden is very high [22].

An alternative approach comes from the use of algorithms from the Evolutionary Computation (EC) field, which have been used in the past to optimize nonlinear problems with a large number of variables. These techniques have been applied with success to the optimization of feeding or temperature trajectories [14][1], and, when compared with traditional methods, usually perform better [19][6].

In this chapter, the application of Differential Evolution (DE) to the optimization of input variables in fed-batch fermentation processes is proposed. The DE is implemented and tested in several distinct variants and compared to other algorithms from the EC field, such as Evolutionary Algorithms (EAs) and Particle Swarm Optimization (PSO) .

Three case studies were used to illustrate and validate the approach and to compare the performance of the different algorithms. 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 on the convergence speed.

This work also tackled the complementary issue of online optimization . In fact, in a real environment, even when the mathematical models used for open-loop optimization are reliable and validated by experimentation, several sources of noise can contribute to changes in the observed values of the state variables. These issues are of particular importance when dealing with recombinant high-cell density fermentations, as the process, besides the nonlinearities exhibited, tends to change dramatically upon some events, like induction. Also, it is likely that there exists a time-variance of both yield and kinetic parameters not contemplated in most process models. These scenarios have an important impact on the experimental results, that end up being worse than the ones predicted after running the offline optimization.

An alternative to cope with these model inaccuracies is the use of online optimization algorithms that periodically generate new solutions as the process is running, making use of the measurement of relevant state variables for update of the internal model. In this case, the optimization is performed simultaneously, taking into account values of the state variables measured by sensors within the fermentation process.

The performance of DE in this online optimization task was evaluated and compared to the results of a real-valued EA. The same case studies used in offline optimization, are now used in order to test the performance of both algorithms. These are firstly used to perform an offline optimization and then a simulation of a real-world fermentation is conducted. The relevant state variables are, in each case, disturbed by adding noise, at regular periods of time. The behavior of both algorithms is compared, as well as the degradation in performance when the initial offline solution is subjected to perturbations.

This chapter is organized as follows: firstly, the fed-batch fermentation case studies are presented; next, the optimization algorithms are described; the results of the application of the different algorithms to the case studies are presented, followed by a discussion of the results; online optimization is described, followed by the description of the experiments conducted and the discussion of the results; finally, the conclusions and further work are presented.

2 Case Studies: Fed-Batch Fermentation Processes

In fed-batch fermentations there is an addition of certain nutrients to the bioreactor along the time, in order to prevent the accumulation of toxic products, allowing the achievement of higher product concentrations.

During this process the system states change considerably, from a low initial to a very 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. The typical inputs in this process are the substrate inflow rates time profiles.

For the proper optimization 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.

2.1 Case Study I

In previous work by the authors, a fed-batch recombinant *Escherichia coli* fermentation process was optimized by *EAs* [16][17]. 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 microorganism can follow three main different metabolic pathways:

- Oxidative growth on glucose:

$$k_1 S + k_5 O \xrightarrow{\mu_1} X + k_8 C \tag{1}$$

- Fermentative growth on glucose:

$$k_2 S + k_6 O \xrightarrow{\mu_2} X + k_9 C + k_3 A \tag{2}$$

- Oxidative growth on acetic acid:

$$k_4 A + k_7 O \xrightarrow{\mu_3} X + k_{10} C \tag{3}$$

where S, O, X, C and 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 (in g/kg); μ_1 to μ_3 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:

$$\frac{dX}{dt} = (\mu_1 + \mu_2 + \mu_3)X - DX \tag{4}$$

$$\frac{dS}{dt} = (-k_1\mu_1 - k_2\mu_2)X + \frac{F_{in,S}S_{in}}{W} - DS \tag{5}$$

$$\frac{dA}{dt} = (k_3\mu_2 - k_4\mu_3)X - DA \tag{6}$$

$$\frac{dO}{dt} = (-k_5\mu_1 - k_6\mu_2 - k_7\mu_3)X + OTR - DO \tag{7}$$

$$\frac{dC}{dt} = (k_8\mu_1 + k_9\mu_2 + k_{10}\mu_3)X - CTR - DC \tag{8}$$

$$\frac{dW}{dt} \simeq F_{in,S} \tag{9}$$

being D the dilution rate, $F_{in,S}$ the substrate feeding rate (in kg/h), W the fermentation weight (in kg), OTR the oxygen transfer rate and CTR the carbon dioxide transfer rate.

The kinetic behavior, expressed in the rates μ_1 to μ_3, was given by specific functions of the state variables, whose description is out of the scope of the present work but can be found in [15].

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 related with the final biomass obtained, when the duration of the process is predefined. 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: $W(t) \leq 5$. The final time (T_f) is set to 25 hours.

2.2 Case Study II

This system is a fed-batch bioreactor for the production of ethanol by *Saccharomyces cerevisiae*, firstly studied by Chen and Huang [5]. The aim is to find the substrate feed rate profile that maximizes the final amount of ethanol.

The model equations are the following:

$$\frac{dx_1}{dt} = g_1 x_1 - u\frac{x_1}{x_4} \tag{11}$$

$$\frac{dx_2}{dt} = -10 g_1 x_1 + u\frac{150 - x_2}{x_4} \tag{12}$$

$$\frac{dx_3}{dt} = g_2 x_1 - u\frac{x_3}{x_4} \tag{13}$$

$$\frac{dx_4}{dt} = u \tag{14}$$

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 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: $PI = x_3(T_f)x_4(T_f)$.

The final time is set to $T_f = 54$ hours, 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: $0 \leq x_4(t) \leq 200$ for all time points and the feeding rate $0 \leq u(t) \leq 12$.

2.3 Case Study III

This case study handles a hybridoma reactor described by the equations [19]:

$$\frac{dX_v}{dt} = (\mu - k_d)X_v - \frac{F_1 + F_2}{V}X_v \tag{17}$$

$$\frac{dGlc}{dt} = \frac{F_1}{V}Glc_{in} - \frac{F_1 + F_2}{V}Glc - q_{Glc}X_v \tag{18}$$

$$\frac{dGln}{dt} = \frac{F_2}{V}Gln_{in} - \frac{F_1 + F_2}{V}Gln - q_{Gln}X_v \tag{19}$$

$$\frac{dLac}{dt} = q_{Lac}X_v - \frac{F_1 + F_2}{V}Lac \tag{20}$$

$$\frac{dAmm}{dt} = q_{Amm}X_v - \frac{F_1 + F_2}{V}Amm \tag{21}$$

$$\frac{dMab}{dt} = q_{Mab}X_v - \frac{F_1 + F_2}{V}Mab \tag{22}$$

$$\frac{dV}{dt} = (F_1 + F_2) \tag{23}$$

where the state variables X_v, Glc, Gln, Lac, Amm, Mab, V are the concentrations of viable cells, glucose, glutamine, lactate, ammonia, monoclonal antibodies and culture volume, respectively. The control variables F_1 and F_2 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 [19].

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:

$$PI = \int_0^{T_f} -q_{Mab}X_v(t)V(t) \tag{24}$$

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

3 Algorithms

3.1 Solution Representation and Evaluation

The optimization task addressed in this chapter is to find the best trajectory of some input variables (e.g. substrate feed), that yield the maximum performance index, defined in each specific case. A solution to the problem will consist in a real-valued vector, that encodes a temporal sequence of values, one per each time unit.

As mentioned above, the typical input variable in fed-batch fermentation processes is the feeding trajectory or trajectories, i.e. the amount of a given substrate to be introduced into the bioreactor, in a given time unit. Case studies I and II have only one input variable, given by the substrate feeding rate ; case study III has two feeding rates F_1 and F_2.

The size of the solution will be determined by the final time of the process (T_f), the discretization step (d) considered in the numerical simulation of the model and the number of input variables NV, given by the expression: $NV\frac{T_f}{d}$.

However, as the resulting genome would be very large (e.g. 5000 genes, for case study I), feeding values were defined only at certain equally spaced points, and the remaining values are linearly interpolated . The size of the genome (G) becomes:

$$G = NV(\frac{T_f}{dI} + 1) \tag{25}$$

where I stands for the number of points within each interpolation interval. The value of d used in the experiments was $d = 0.005$, for all case studies.

The evaluation of each solution is performed by running a numerical simulation of the defined model, given as input the feeding values. The numerical simulation is performed using *ODEToJava*, a package of ordinary differential equation solvers, using a linearly implicit implicit/explicit (IMEX) Runge-Kutta scheme used for stiff problems [2]. The fitness value is then calculated from the values of the state variables according to the *PI* defined for each case.

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 $f : \mathbb{R}^n \to \mathbb{R}$ to be minimised, a DE begins by randomly generating p n-dimensional vectors. These vectors (called individuals) form a population that will evolve over the course of the algorithm's run. The algorithm then proceeds to manipulate the population until a termination criterion is met. The termination condition can be that a fixed number of function evaluations have elapsed or no sufficient improvement is achieved.

The following is an outline of DE that uses a binomial crossover [21]. For clarity, the computation of the new trial vector has been shown separately from the crossover operation that selects only some of the dimensions of the trial vector.

1. Initialize the population;
2. Evaluate the population;
3. Generate a new population where each candidate individual i is generated in parallel according to:
   (i) Randomly select 3 distinct individuals r_1, r_2, r_3 from the population that are different from i;
   (ii) Generate a trial vector based on the scheme
   (iii) Perform crossover between this vector and the vector of the current individual, with probability CR, using at least one dimension of the trial vector.
   (iv) If the candidate is not valid, change its invalid coordinates by resetting them to the closest bound;
   (v) Evaluate the candidate;
   (vi) Use the candidate in the new generation if it is at least as good as the current individual;
   (vii) 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 DEs [20]. Each scheme varies according to the number of difference vectors used and to whether or not the current individual or the global best individual will be used as part of that computation. Four schemes are considered in this paper. These are shown below along with

the corresponding trial vector generation formula. The variables x_{r_j}, $2 \leq j \leq 5$ represent distinct randomly selected individuals that are different from the current individual x_i and x_{best} is the best individual. The parameter $F \in \mathbb{R}$ is the scale of the difference vectors and is usually set between 0 and 2 and CR is the crossover probability.

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

3.3 Real-Valued EA

A real-valued Evolutionary Algorithm (EA) was also considered, since it provided good results in previous work [17][18]. 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 criteria is not met:
   (i) Select from P_t a subset of individuals for reproduction.
   (ii) Apply the genetic operators to the individuals in order 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:

- *Random Mutation*, which replaces one gene by a new randomly generated value, within the range $[min_i, max_i]$ [13]; and
- *Gaussian Mutation*, which adds to a given gene 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 [13], applied in the traditional way;
- *Arithmetical crossover*, where each gene in the offspring will be a linear combination of the values in the ancestors' chromosomes [13];

where $[min_i; max_i]$ is the range of values allowed for gene i.

Both mutation operators are applied to a variable number of genes (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 [17]. 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, 50% of the individuals are kept from the previous generation, and 50% are bred by the application of the genetic operators.

3.4 Fully Informed Particle Swarm

A particle swarm optimizer (PSO) uses a population of particles that evolve over time by flying 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 [11] the Fully Informed Particle Swarm (FIPS) [12] clearly outperformed the canonical particle swarm in this class of problems, this method will be used in this study. 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 and Kennedy [7] 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 given by:

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

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. The values of φ and χ are given by Clerc's formula. In this study, the population is organized according to the *von Neumann* topology [10], where each particle is connected to four others, in a torus configuration.

4 Offline Optimization

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 [8] for two-sample comparisons was adopted. In order to improve the readibility of the analysis, a symbolic encoding of the p-values resulting from the t-tests was used. To enhance readability of the tables and allow a straighforward comparison between the approaches tested, different symbols are used to report whether the mean of approach $A1$ is greater than the mean of $A2$ or vice-versa. The encoding used is presented in Table 1.

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

p-value	condition	symbol
$p \leq 0.001$	$mean(A1) > mean(A2)$	+++
$p \leq 0.001$	$mean(A1) < mean(A2)$	- - -
$0.001 < p \leq 0.01$	$mean(A1) > mean(A2)$	++
$0.001 < p \leq 0.01$	$mean(A1) < mean(A2)$	- -
$0.01 < p \leq 0.05$	$mean(A1) > mean(A2)$	+
$0.01 < p \leq 0.05$	$mean(A1) < mean(A2)$	-
$p \geq 0.05$		O

Given that multiple pairwise comparisons were performed, the authors used the *Holm* correction for the p-values [9]. 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 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, these experiments take a long time (typically a few hours per run) and there are usually time constraints. Thus, it was decided to use standard configurations for each algorithm that were either validated by 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 200,000 function evaluations. In the case of FIPS the population size was 20 and the other parameters have the usual values given in the literature. The neighborhood topology selected was the *von Neumann* [10].

For all *DE* algorithms, the population size was set to 20, F was set to 0.5, CR to 0.6 and the schemes used were *DE/rand/1*, *DE/rand/2*, *DE/best/1* and *DE/best/2*. In terms of the real-valued EA, the population size was set to 200.

For each case study, 30 runs were performed with each algorithm. The value of I was determined, for each case study, based on preliminary results, and set in the following way: $I = 200$ in case studies 1 and 2, and $I = 100$ for the case study III.

Thus, the solution sizes are equal to 26, 55 and 21 for the three case studies.

4.3 Results

For all case studies, the results will be shown in two distinct tables. The first 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 50,000, 100,000 and 200,000 function evaluations 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, when 200 k FEs have elapsed, using the methodoloy aforementioned. This will show the statistically significant 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 and 3, 4 and 5 and finally 6 and 7, present the results obtained by each of the algorithms on the case studies I, II and III, respectively.

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

Algorithm	PI 50,000 FEs	PI 100,000 FEs	PI 200,000 FEs
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
DE/rand/2	O				
DE/best/1	- - -	- - -			
DE/best/2	O	+++	+++		
EA	- - -	- - -	+++	- - -	
FIPS	O	+++	+++	O	+++

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

Algorithm	PI 50,000 FEs	PI 100,000 FEs	PI 200,000 FEs
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
DE/rand/2	O				
DE/best/1	- - -	- - -			
DE/best/2	O	O	+++		
EA	- -	O	+++	O	
FIPS	- - -	- - -	O	- - -	- - -

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

Algorithm	PI 50,000 FEs	PI 100,000 FEs	PI 200,000 FEs
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

The first conclusion to be drawn from the results is a superiority of some of the DE schemes (specially DE/rand/1 and DE/rand/2) over the EA and FIPS as soon as 50,000 FEs. FIPS converges fast, but the quality of the solutions in cases II and III is not at the level of the results of DE and even the EA. The EA

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

is usually the algorithm with the slowest convergence, although it steadly improves over the entire run. The worst algorithm in all problems is the DE/best/1 scheme. DE/best/2 is much better showing that in some problems having a noisier setup with a greedier scheme can pay off. However, it is still a step behind the DE/rand alternatives.

In case study I, DE/rand/1 has already obtained good solutions with only 50,000 FEs, closely followed by FIPS and DE/best/2. When 100,000 FEs have elapsed, the quality of the solutions of the three approaches is very similar, with the EA still trailing far behind and DE/best/1 out of the competition. Finally, with 200,000 FEs the three approaches (DE/rand/1, FIPS and DE/best/2) maintain similar performance and the EA is still trailing behind, although steadily improving.

In case study II, there are some changes. FIPS is not as good as the EA and presents results similar to the ones of DE/best/1. DE/rand/1 and DE/rand/2 have similar results with 50,000 FEs, slowly improving with a larger number of FEs. The EA is the algorithm that shows the steadier improvement but still exhibits a somewhat lower performance when compared to DE/rand at 200,000 FE.

Case study III shows a similar performance for DE/rand/1 and DE/rand/2, with DE/rand/2 having somewhat better performance for 100,000 and 200,000 FEs. In this case, the EA is again the third best alternative and the worst performer is still DE/best/1.

5 Online Optimization

5.1 Description

During the fermentation process, some of the state variables can be measured, but its values are scarcely used for closed-loop optimization purposes, and are rather employed to evaluate qualitatively the performance of the process. However, it is possible to develop dynamic optimization algorithms capable of timely reacting to this new knowledge generated by updating the corresponding internal model and generating new solutions.

EC is a promising approach to this real-time optimization task, since the algorithms keep a population of solutions that can be easily adapted to perform re-optimization. Indeed, a population of solutions previously obtained can be evaluated under the new scenario and better adapted solutions can be created

through the use of evolution. The fact that a set of solutions is kept, and not only the best solution, makes a faster adaptation to new conditions possible, while taking advantage of previous optimisation efforts.

In this work, two online optimization strategies based on *EAs* and *DE* are proposed, working in two stages: before the fermentation process starts, an offline optimization is conducted as it is described in the previous sections. After this preliminary step, online optimization algorithms use information gathered by measuring the value of relevant state variables in certain points in time during the real fermentation. These algorithms react by updating their internal model and reaching an improved solution, that is available to be sent back to the fermentation monitoring software.

The version of the *EA*/*DE* used to perform online optimization is similar to the ones described for the offline problem. The DE scheme selected was DE/rand/1 due to the fact that it is the simplest to implement and the one that usually gave the best results. When new information regarding the state variables is received, the following steps are followed by both *EA* and *DE*:

1. a starting point (in time) is determined for the re-optimized solution, by adding the time label of the received data with the predicted time necessary to compute a new solution (since it is impossible to reach and therefore apply a solution before that time);
2. the last available population is adapted by removing from the genome of each individual the genes that encode feeding values for elapsed time periods;
3. half of the individuals in the population are replaced by new randomly generated solutions (these individuals are chosen randomly, although the best individual is always kept). This helps in maintaining genetic diversity, a useful feature for the optimization in changing landscapes;
4. the internal model of the fermentation used by the *EA*/*DE* is updated with the new information available from the real process and each of the individuals is re-evaluated taking this new knowledge into consideration;
5. the normal process of the *DE* or *EA* proceeds for a given number of iterations;
6. the best solution obtained is sent to the fermentation software and can be used in the real process.

5.2 Experimental Setup

In this study, and given time and physical constraints, real fermentations were not conducted and instead these were replaced by simulating the fermentation process and adding noise to the value of the state variables. This process is implemented by considering two interacting components: an *optimizer*, that implements the optimization algorithm (*DE* or *EA*), and a *noise simulator (NS)*, that simulates the real fermentation process adding noise to the state variables.

This is performed by considering that there is a deviation between the model prediction and the behaviour of the process due to several reasons (e.g. model innacuracies or parameters changing over time). Therefore, for each sampling time, the state variables that represent the real process are obtained from the

simulated variables by adding noise. These new values of the state variables would originate a deviation of the process from its optimal behavior, which had been defined during offline optimization. To compensate for this deviation, the new values of the state variables will be used by the optimization algorithm to reach a new feeding profile.

The following sequence of events takes place:

1. an offline optimization is performed by the *optimizer* and its results are passed on to the *NS*, used to compute the predicted values of the state variables. The *optimizer* stops and waits for new information.
2. the variable t, which stores the simulated time in the *NS* is set to $t = 0$.
3. while $t < T_f$ (where T_f denotes the final time of the fermentation process) the following steps are executed:
   (i) the values of all state variables at time t are disturbed by the *NS* by adding/ subtracting noise, given by the original value multiplied by a value taken from an uniform distribution with range $[0, U]$. The new values of the state variables are sent to the *optimizer*.
   (ii) the *optimizer* receives this information and runs the steps for online optimization listed in the previous section. The best solution reached is sent to the *NS* that updates its model accordingly.
   (iii) the *NS* updates $t = t + \Delta t$.

Each run for the initial optimization is stopped after $200,000$ function evaluations and the re-optimization process is allowed $20,000$ function evaluations. The parameters of the *DE* and *EA* keep the values of the offline optimization given in the previous sections The value of Δt was set to 1 (h.) in case study I, 2 (h.) in II and 0.5 (d.) in III.

5.3 Results

The results will be presented in terms of the mean of the *PI* values obtained in 30 runs, as well as 95% confidence intervals. The Tables 8, 9 and 10 show the results of the algorithms obtained on case studies I, II and III, respectively. In every case, the first column represents the parameter U used to generate noise (an increase in this parameter implies noisier setups). The next two columns show the results for the *DE* and the *EA* during offline optimization; columns 4 and 5 show the results obtained for the same algorithms, but applying the noise disturbances without changing the solutions of offline optimization (simulating the case where there are discrepancies between model predictions and real processes but without intervention of online optimizers) and, finally, the last two columns show the results obtained by the online optimization.

The first conclusion to draw from the results is that, in every case study, even a low level of noise is enough to clearly disturb the results, although that effect is clearly more visible in case study I.

The levels of noise studied are certainly within the range of the differences observed between model predictions and experimental results in biotechnological processes. However, the consequences in terms of process performance when an

Table 8. Results obtained by the *DE* and *EAs* in case study I

U	Initial optim.		Initial+noise		Online opt.	
	DE	**EA**	**DE**	**EA**	**DE**	**EA**
0.01	9.47 ± 0.00	8.85 ± 0.04	4.67 ± 0.70	4.79 ± 0.73	9.11 ± 0.14	8.72 ± 0.14
0.02	9.47 ± 0.00	8.83 ± 0.05	4.41 ± 0.75	4.69 ± 0.78	8.80 ± 0.24	8.53 ± 0.25
0.03	9.47 ± 0.00	8.81 ± 0.05	4.20 ± 0.76	4.35 ± 0.81	8.47 ± 0.34	8.17 ± 0.35

Table 9. Results obtained by the *DE* and *EAs* in case study II

U	Initial optim.		Initial+noise		Online opt.	
	DE	**EA**	**DE**	**EA**	**DE**	**EA**
0.01	20405 ± 4	20374 ± 9	20097 ± 133	20236 ± 108	20421 ± 115	20408 ± 119
0.02	20407 ± 3	20379 ± 7	19832 ± 305	19986 ± 244	20404 ± 243	20392 ± 242
0.03	20405 ± 5	20376 ± 9	19711 ± 357	19938 ± 393	20282 ± 317	20236 ± 335

Table 10. Results obtained by the *DE* and *EAs* in case study III

U	Initial optim.		Initial+noise		Online opt.	
	DE	**EA**	**DE**	**EA**	**DE**	**EA**
0.01	394.7 ± 0.2	386.3 ± 0.8	371.7 ± 8.5	367.9 ± 7.1	386.2 ± 4.8	379.8 ± 3.8
0.02	394.7 ± 0.2	385.2 ± 0.7	353.9 ± 14.9	351.2 ± 12.3	374.1 ± 9.2	371.8 ± 8.3
0.03	394.7 ± 0.2	386.1 ± 0.9	330.0 ± 23.5	343.0 ± 15.4	364.5 ± 13.0	367.6 ± 11.0

open-loop fermentation (without online optimization) is performed are quite extreme, implying that in many cases the utility of even relatively good models for process optimization with current state-of-the-art optimization techniques (mostly offline approaches) is quite low.

Therefore, the results obtained with online optimization strategies indicate that the reward obtained in terms of process productivity is probably more than enough to justify its implementation and the corresponding costs. In fact, he results obtained for all 3 case studies are quite close to the ones predicted by offline optimization without added noise, thus implying that the optimization scheme is robust to the levels of noise studied in this work. Furthermore, the degradation of the results that is caused by the increase of U is quite graceful, as an increase in U does not cause dramatic effects in the PI.

A comparison of the results obtained by both optimization algorithms show that *DE* seems to be more effective than the *EAs*. The difference is very clear when offline optimization is performed, but decreases when the level of noise increases. In fact, the differences for $U = 0.02$ and 0.03 are not significant from a statistical perspective and in case study III, the *EA* displays a better mean than *DE* for $U = 0.03$. Nevertheless, if an alternative has to be chosen the *DE* still has an advantage, since it shows the best results (mean) in almost all scenarios.

6 Conclusions and Further Work

This chapter compares FIPS, a real-valued EA (*EA*) and four distinct schemes of *Differential Evolution (DE)* in three case studies of optimizing the feeding trajectory in fed-batch fermentation processes. The best overall algorithms in these tasks were the *DE/rand/1* and *DE/rand/2*, that consistently obtained good results in all the case studies and furthermore had a good convergence speed. If a single configuration was to be chosen, the DE/rand/1 scheme would be the selected one, since it represents the simpler alternative to implement and obtains good results.

Fips was a good contender in one of the cases where it found good results and was as fast as DE. However, it got stuck on local optima on the other ones. EA was slower to converge but reliable. If one can afford the computational time needed, it always finds good solutions. However, in some problems (specially case study I) it requires a large number of function evaluations to achieve a good result. Given that the computational time needed for these problems is quite large, it is a good reason to choose *DE* instead.

In this work, the task of optimizing feed profiles for fed-batch fermentation problems was also approached by proposing optimization algorithms, such as *EAs* and *DE*, that are able to implement online optimization strategies, i.e., to perform the optimization simultaneously with the real process. The proposed approach was validated by conducting a number of experiments that used a noise simulator to emulate the differences between the values predicted by the mathematical model and the real values in the fermentation process. The results of the experiments show that even small differences lead to important disruptions in the behavior that was predicted by offline optimization.

The proposed approach to online optimization deals well with the noise and exhibits properties of graceful degradation. When comparing the optimization algorithms, the *DE* seems the best alternative, but its superiority seems to decrease when noisier settings are considered.

In future work, the priority is to validate these results by implementing the approach to online optimization with a real fed-batch fermentation process. Furthermore, other case studies will be tested and distinct optimization algorithms will be taken into account.

Previous work by the authors [18] 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.

Acknowledgments

This work was supported in part by the Portuguese Foundation for Science and Technology under project POSC/EIA/59899/2004. The authors wish to thank Project SeARCH (Services and Advanced Research Computing with HTC/HPC clusters), funded by FCT under contract CONC-REEQ/443/2001, for the computational resources made available.

References

1. Angelov, P., Guthke, R.: A Genetic-Algorithm-based Approach to Optimization of Bioprocesses Described by Fuzzy Rules. Bioprocess Engin. 16, 299–303 (1997)
2. Ascher, Ruuth, Spiteri: Implicit-explicit runge-kutta methods for time-dependent partial differential equations. Applied Numerical Mathematics 25, 151–167 (1997)
3. Banga, J.R., Moles, C., Alonso, A.: Global Optimization of Bioprocesses using Stochastic and Hybrid Methods. In: Floudas, C.A., Pardalos, P.M. (eds.) Frontiers in Global Optimization - Nonconvex Optimization and its Applications, vol. 74, pp. 45–70. Kluwer Academic Publishers, Dordrecht (2003)
4. Bryson, A.E., Ho, Y.C.: Applied Optimal Control - Optimization, Estimation and Control. Hemisphere Publication Company, New York (1975)
5. Chen, C.T., Hwang, C.: Optimal Control Computation for Differential-algebraic Process Systems with General Constraints. Chemical Engineering Communications 97, 9–26 (1990)
6. Chiou, J.P., Wang, F.S.: 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 (1999)
7. Clerc, M., Kennedy, J.: The particle swarm - explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation 6(1), 58–73 (2002)
8. Goulden, C.H.: Methods of Statistical Analysis, 2nd edn. John Wiley & Sons Ltd., Chichester (1956)
9. Holm, S.: A simple sequentially rejective multiple test procedure. Scandinavian Journal of Statistics 6, 65–70 (1979)
10. Kennedy, J., Mendes, R.: Topological structure and particle swarm performance. In: Fogel, D.B., Yao, X., Greenwood, G., Iba, H., Marrow, P., Shackleton, M. (eds.) Proceedings of the Fourth Congress on Evolutionary Computation (CEC 2002), Honolulu, Hawaii, May 2002. IEEE Computer Society, Los Alamitos (2002)
11. Mendes, R., Rocha, I., Ferreira, E., Rocha, M.: A comparison of algorithms for the optimization of fermentation processes. In: 2006 IEEE Congress on Evolutionary Computation, Vancouver, BC, Canada, July 2006, pp. 7371–7378 (2006)
12. Mendes, R., Kennedy, J., Neves, J.: The fully informed particle swarm: Simple, maybe better. IEEE Transactions on Evolutionary Computation 8(3), 204–210 (2004)
13. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs, 3rd edn. Springer, USA (1996)
14. Moriyama, H., Shimizu, K.: On-line Optimization of Culture Temperature for Ethanol Fermentation Using a Genetic Algorithm. Journal Chemical Technology Biotechnology 66, 217–222 (1996)
15. Rocha, I.: Model-based strategies for computer-aided operation of recombinant E. coli fermentation. PhD thesis, Universidade do Minho (2003)
16. Rocha, I., Ferreira, E.C.: 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 (2002)
17. Rocha, M., Neves, J., Rocha, I., Ferreira, E.: Evolutionary algorithms for optimal control in fed-batch fermentation processes. In: Raidl, G.R., Cagnoni, S., Branke, J., Corne, D.W., Drechsler, R., Jin, Y., Johnson, C.G., Machado, P., Marchiori, E., Rothlauf, F., Smith, G.D., Squillero, G. (eds.) EvoWorkshops 2004. LNCS, vol. 3005, pp. 84–93. Springer, Heidelberg (2004)

18. Rocha, M., Rocha, I., Ferreira, E.: 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, Los Alamitos (2005)
19. Roubos, J.A., van Straten, G., van Boxtel, A.J.: An Evolutionary Strategy for Fed-batch Bioreactor Optimization: Concepts and Performance. Journal of Biotechnology 67, 173–187 (1999)
20. Storn, R.: 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, Los Alamitos (1996)
21. Storn, R., Price, K.: Minimizing the real functions of the icec'96 contest by differential evolution. In: IEEE International Conference on Evolutionary Computation, pp. 842–844. IEEE, Los Alamitos (1996)
22. Tholudur, A., Ramirez, W.F.: Optimization of Fed-batch Bioreactors Using Neural Network Parameters. Biotechnology Progress 12, 302–309 (1996)