Evolutionary Algorithms for Static and Dynamic Optimization of Fed-batch Fermentation Processes [1]

Miguel Rocha[†], José Neves[†], Ana C.A. Veloso[⋆‡],
Eugénio C. Ferreira[⋆] and Isabel Rocha[⋆]
[†]Dep. Informática, University of Minho, Portugal
[⋆]Centro de Engenharia Biológica, University of Minho, Portugal
[‡]Esc. Superior Agrária de Bragança, Bragança, Portugal
{mrocha,jneves}@di.uminho.pt {anaveloso,ecferreira,irocha}@deb.uminho.pt

Abstract

In this work, *Evolutionary Algorithms (EAs)* are used to control a recombinant bacterial fed-batch fermentation process, that aims to produce a bio-pharmaceutical product. Initially, a novel *EA* was used to optimize the process, prior to its run, by simultaneously adjusting the feeding trajectory, the duration of the fermentation and the initial conditions of the process. Finally, dynamic optimization is proposed, where the *EA* is running simultaneously with the fermentation process, receiving information regarding the process, updating its internal model and reaching new solutions that will be used for online control.

Keywords: Evolutionary Computation, Fed-batch fermentation optimization, Online optimization, Variable size chromosomes, Real-valued representations

1 Introduction

Valuable products such as recombinant proteins, antibiotics and amino-acids are produced using fermentation techniques and thus there is an enormous economic incentive to optimize such processes. However, these are usually very complex, involving different transport phenomena, microbial components and biochemical reactions. Furthermore, the nonlinear behavior and time-varying properties make bioreactors difficult to control with traditional techniques. Thus, it is necessary to consider quantitative mathematical models, capable of describing the process dynamics and the interrelation among relevant variables. Additionally, robust optimization techniques must deal with the model's complexity, the environment constraints and the inherent noise of the experimental process.

Several optimization methods are described in literature. It has been shown that, for simple bioreactor systems, the problem can be solved analytically. Numerical methods, such as gradient algorithms based on the local sensitivities of the objective function for changes in the values of the control variables, have also been used [1]. One other popular method is dynamic programming which discretizes both time and control variables to a predefined number of values and uses a systematic backward search. However, those methodologies become too complex when the number of state variables increases. An alternative approach comes from *Evolutionary Algorithms (EAs)*, where previous work has reached interesting results in the optimization of feeding trajectories [2][3].

In this work, a fed-batch recombinant *E. coli* fermentation process to produce a bio-pharmaceutical product [4] was studied. In fed-batch fermentations there is an addition of certain nutrients along the process, in order to prevent the accumulation of toxic sub-products. The purpose is to apply real-valued representation based *EAs*, with variable sized chromosomes, in order to achieve both static and dynamic optimization. The former is conducted offline before the process starts and aims at setting some of the fermentation's variables, namely the substrate feeding trajectory, the initial conditions and also the duration of the fermentation. The latter is conducted in real time, interacting with the fermentation and optimizing the feeding trajectory by reacting to information about the real values of relevant state variables.

2 The fed-batch fermentation process

E. coli is the microorganism of choice for the production of bio-pharmaceuticals, usually grown under fed-batch mode due to the negative effect of acetic acid produced when glucose is present above certain concentrations. During this process the system states change considerably, from a low initial to a high biomass and product concentrations. This dynamic behavior motivates the development of optimization methods to find the optimal input feeding trajectories. The typical input is the sub-

[1] Ana Veloso is most grateful for the financial support provided by PRODEP.

strate inflow rate as a function of time. One way to evaluate the process performance is the productivity, defined as the units of product formed per unit of time, which is usually related with the final biomass obtained.

As previously presented [4], a white box mathematical model was developed, based on differential equations that represent the mass balances to the state variables, namely $F_{in,S}$ the substrate feeding rate (in kg/h), W the culture medium weight (in kg), S the glucose, X the biomass, O the dissolved oxygen, C the dissolved carbon dioxide and A the acetate. Real experiments were used as the basis for the model derivation, being conducted in a fermentation laboratory with a 5-L bioreactor and the experimental results were consistent with the model. This model will be used for the optimization described in the following, performing model simulation, by using the Euler numerical integration method, with a small step size d and a given duration for the process (T_f) measured in hours.

3 Evolutionary Algorithms for Static Optimization

3.1 Evolutionary Algorithms for Feeding Trajectory Optimization

The first approach to the problem was to develop an *EA* capable of optimizing the feeding trajectory, i.e., to determine the amount of substrate (glucose) to be fed into the bioreactor per time unit ($F_{in,S}$). Real-valued representations were used in order to encode the feeding amounts. Each gene will encode the amount of substrate to be introduced into the bioreactor in a given time unit and the genome will be given by the temporal sequence of such values.

In this case, the size of the genome is determined based on the final time of the process (T_f) and the discretization step (d) considered in the simulation, being given by the expression: $\frac{T_f}{d}$. However, this could produce a very large genome (a typical value would be 5000), which would difficult the *EA*'s convergence. Thus, feeding values are defined only at certain equally spaced points, and the remaining values are linearly interpolated. The size of the genome becomes $\frac{T_f}{dp} + 1$, where p stands for the number of points within each interpolation interval. The values used in the experiments are: $T_f = 25$, $d = 0.005$ and $p = 200$.

The amount of substrate that can be introduced per time unit is limited to a maximum of 0.4 kg/h due to the pump's capacity. Thus, there is the need to impose limits in the gene values that must be within the range $[0.0; 0.4]$. In the initial population, each individual is assigned, for each of its genes, a random value in this interval.

The evaluation process, for each individual in the population, measures the quality of the feeding trajectory in terms of the fermentation's productivity. This calculation is achieved by firstly running a simulation of the model, given as input the feeding values in the genome. In each simulation, the relevant state variables are initialized with the following values: $X_0 = 5$, $S_0 = 0$, $A_0 = 0$, $W_0 = 3$. The fitness value is then calculated from the final and initial values of the state variables X and W, by the expression

$$\frac{X_f * W_f - X_0 * W_0}{T_f}$$

Both mutation and crossover operators were taken into account. Two mutation operators were used, namely: *Random Mutation* (replaces one gene by a new randomly generated value, within the range $[min_i, max_i]$) and *Gaussian Mutation* (adds to a given gene a value taken from a Gaussian distribution, with a zero mean and a standard deviation given by $\frac{max_i - min_i}{4}$, where $[min_i; max_i]$ is the range of values allowed for gene i). In both cases, the operators were applied to a variable number of genes (a value that is randomly set between 1 and 10 in each application). On the other hand, the following crossover operators were chosen: *Two-Point*, *Uniform*, *Arithmetical* and *Sum* crossovers [5].

The population size was set to 200 and 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 were kept from the previous generation, and 50% were bred by the application of the genetic operators.

The implementation of the proposed *EA* was based on a general purpose package, developed by the authors in the *Java* programming language. All experiments reported were run on a *PC* with a *Pentium IV 2.4 GHz* processor.

A set of experiments was conducted in order to find the best set of genetic operators for this problem [4]. The best result was obtained using an alternative that contemplates the use of all genetic operators described above. In this case each crossover operator is responsible for breeding 12.5% of the offspring and each mutation operator 25%.

3.2 Optimization of Initial Conditions

The initial conditions of the experiments were set based on the practitioner's experience and wisdom. However, there is no guarantee that the initial values of the state variables are optimal. So, it was decided to incorporate the initial values of significant state variables in the optimization procedure.

Once each variable has different physical constraints it was necessary to define a genome where the limits are distinct in each position. The variables chosen to be optimized, additionally to the feeding trajectory, were the initial values of X, W, S and A, with their range of variability given by $X_0 \in [1; 5]$, $W_0 \in [2; 4]$, $S_0 \in [0; 5]$ and $A_0 \in [0; 5]$.

3.3 Optimization of the Final Time

The duration of the fermentation is not imposed by any theoretical result, yet is chosen by empirical knowledge, making it possible to optimize its value. In this section, an *EA* will be proposed for this task considering variable size chromosomes and new genetic operators.

The genetic operators defined in Section 3.1 were kept: the mutations were unchanged and the crossovers were updated to cope with parents of different sizes. In this case, each of the offspring keeps the size of one parent and for the genes where only one parent is defined (the one with greater length), their value is passed into the corresponding offspring. In the creation of the initial population the individuals are given chromosomes with distinct sizes, randomly selected in a range defined by two parameters: a minimum and a maximum size. Furthermore, two novel mutation operators were defined, in order to allow for the change of the size of individuals during the evolution process:

- *Grow*: consists in the introduction of a new gene into the genome, in a random position, being its value the average of the values of the two neighboring genes.

- *Shrink*: a randomly selected gene is removed from the genome.

Both operators are only applied when the maximum and minimum size constraints are obeyed. With the introduction of the new genetic operators, the probabilities used in the experiments are the following: each of the four crossover operators has a probability value of 10%, the random and gaussian mutation keep their probabilities of 25% and the new mutation operators have a probability of 5% each.

Two different experiments were conducted: in the first, only the final time and feeding trajectory are optimized, being the genome made out of the feeding trajectory; in the latter, the initial conditions are also considered a target of optimization, being the initial parameters encoded into the first group of four genes (fixed size), as before, and the remaining of the genome used to code the feeding trajectory (variable size). The minimum and maximum duration of the fermentation are set to 20 and 50 hours, respectively. The remaining parameters of the *EA* are kept unchanged.

Table 1. Comparison of the results obtained by the *EAs* for feeding trajectory (F), initial conditions (I) and duration (T) optimization.

Optim. aim	Mean and conf.interval	Best res.
F	8.98 $\pm$ 0.06	9.12
F+I	9.38 $\pm$ 0.06	9.46
F+T	9.16 $\pm$ 0.09	9.32
F+T+I	9.44 $\pm$ 0.05	9.50

3.4 Results

A set of experiments was conducted to test the previous approaches and the results obtained are displayed in Table 1. Each alternative was tested by 30 independent runs, and each run was stopped after 3000 generations. In the table, the first column indicates the purpose of the *EA*, where F stands for feeding trajectory, I for initial conditions and T for final time optimization. The results are given in terms of the defined fitness, being shown, in the second column, the mean of the runs and the confidence interval and in the third column the best result obtained over all the runs.

The optimization of both time, feeding trajectory and initial parameters had the best overall results, showing that the *EA* can simultaneously optimize all these aspects. An interpretation of the results led to the conclusion that the best results confirmed the findings of the practitioners and reached after years of practical experiments, although in some cases the results gave some interesting insights to the researchers.

4 Evolutionary Algorithms for Online Optimization

The offline optimization described previously makes use of a simulation model to evaluate each solution. Although this is a reliable model, validated by experimentation, in a real environment several sources of noise can contribute to changes in the observed values of the state values. This scenario has an impact on the experimental results that end up being worse than the ones predicted. During the fermentation process, some of the state variables can be measured, but its values tend to serve only for modeling purposes. However, it is possible to develop dynamic optimization algorithms that are capable of timely reacting to this new knowledge by updating its internal model and generating new solutions.

EAs make a promising approach to this real-time optimization task, since they keep a population of solutions that can be easily adapted to perform re-optimization. Indeed, the population can be evaluated under the new scenario and better adapted solutions created through the use of reproduction operators. 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 work.

In this work, online optimization based on *EAs* is proposed, working in two stages: the former is the static optimization, conducted before the fermentation process and described in the previous section; the latter is the dynamic optimization, where the fermentation monitoring software sends information about the values of state variables to the *EA*, that reacts by updating its internal model and reaching a new optimal solution, that is sent back to the monitoring software.

The *EA* used to perform online optimization is similar to the ones described in the previous section. When new information is received, the *EA* determines a starting point (in time) for its optimization, by adding the time label of the received data with its predicted running time. Then it takes the last available population and adapts it by removing from the genome of each individual the genes that encode feeding values for elapsed time periods. Each of these individuals is re-evaluated taking the new knowledge into consideration and the normal process of the *EA* proceeds for a given number of iterations. The best solution (feed) obtained is then sent to the fermentation process and can be used.

In order to validate this method a set of experiments was conducted, where the online measurements are simulated by adding noise to relevant state variables (X and W were selected since they have a direct impact on the fitness values). These variables were disturbed in regular intervals in time (of 1 hour) by the following rules: $X_i = X_i + X_i.U(-p, p)$ and $W_i = W_i + W_i.U(-p, p)$, where $U(a, b)$ is a value taken from an uniform distribution in the range $[a, b]$. In the experiments, both in static and dynamic optimization, only feed trajectory optimization was considered and the experimental setup was kept from the previous section. The offline *EA* is stopped after 2000 generations, while the online one runs for 500 generations in each time step.

Table 2 summarizes the results obtained with different values of p ranging from 0.01 to 0.1. In the columns the result from the static optimization in shown, followed by the result obtained by the optimal feed with the added noise and finally the result of the online optimization method.

It is possible to observe that the added noise is enough to make an huge impact on the process even when the disturbance is small (p=0.01), a result that shows that the process is quite sensitive to small changes in state variables (a conclusion shared by the researchers with practical experience), and emphasizes the need for dynamic optimization. The online optimization is capable of results near the ones initially expected, therefore compensating the noise, even though there is a decrease in performance when p raises, as would be expected.

Table 2. Results obtained by the *EAs* for online optimization.

p	Initial opt.	Init.+ noise	Online Opt.
0.01	8.84 ± 0.06	4.58 ± 0.21	8.68 ± 0.07
0.02	8.81 ± 0.06	4.49 ± 0.17	8.35 ± 0.10
0.05	8.82 ± 0.09	4.26 ± 0.14	7.67 ± 0.12
0.1	8.68 ± 0.09	4.17 ± 0.18	7.44 ± 0.13

5 Conclusions and further work

In this work *EAs*, based on real-valued representations and variable size chromosomes were proposed in order to optimize relevant parameters for a fermentation process, both in offline and online modes. The results, although based on simulations, show that the offline *EA* is capable of simultaneously optimizing the feeding trajectory, the initial conditions and the duration of a fermentation process. On the other hand, the online optimization deals well with significant changes (up to 10%) in the state variables relevant to the fitness function.

Future work includes the validation of the system by further testing, including both simulations and real fermentation processes. An integration of the time and initial parameter optimization in the dynamic procedure will also be undertaken.

References

[1] A.E. Bryson and Y.C. Ho. *Applied Optimal Control - Optimization, Estimation and Control*. Hemisphere Publication Company, New York, 1975.

[2] J.A. Roubos, G. van Straten, and A.J. van Boxtel. An Evolutionary Strategy for Fed-batch Bioreactor Optimization: Concepts and Performance. *Journal of Biotechnology*, 67:173–187, 1999.

[3] R. Marteijn, O. Jurrius, J. Dhont, C. de Gooijer, J. Tramper, and D. Martens. Optimization of a Feed Medium for Fed-Batch Culture of Insect Cells Using a Genetic Algorithm. *Biotechnol Bioeng*, 81:269–278, 2003.

[4] M. Rocha, J. Neves, I. Rocha, and E. Ferreira. Evolutionary algorithms for optimal control in fed-batch fermentation processes. In G.Raidl et al., editor, *Proceedings of the Workshop on Evolutionary Bioinformatics - EvoWorkshops 2004, LNCS 3005*, pages pp.84–93. Springer, 2004.

[5] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, USA, third edition, 1996.