Reliable Optimal Control of a Fed-Batch Bio-Reactor Using Ant Colony Optimization and Bootstrap Aggregated Neural Networks

Mahmood Hilal Al-Mahrouqi and Jie Zhang

School of Chemical Engineering and Advanced Materials, Newcastle University, Newcastle upon Tyne NE1 7RU, U.K. (Tel: +44-191-2227240; e-mail: jie.zhang@newcastle.ac.uk)

Abstract: Optimal control of a fed-batch bio-reactor using ant colony optimisation and bootstrap aggregated neural network models is presented in this paper. In order to overcome the difficulties in developing detailed mechanistic models and to improve the reliability of data based empirical models, bootstrap aggregated neural networks were used to model a fed-batch bio-reactor using process operational data. Bootstrap aggregated neural networks can not only improve model prediction accuracy but also provide prediction confidence bounds. In order to overcome the problem of local minima in the optimisation, ant colony optimisation (ACO) is used. A modified ACO algorithm is proposed for continuous variable optimisation. In the proposed technique, model prediction confidence bounds are incorporated in the optimisation objective function so as to enhance the reliability of the calculated “optimal” control actions.

1. INTRODUCTION

Batch processes are the dominant mode of operation in a wide range of industries, including food, pharmaceutical and bio-materials industries. Batch and fed-batch processes are routinely used for the manufacturing of high value added products, such as specialty polymers and fine chemicals (Bonvin, 1998). Optimal control of batch processes is very important because, in the face of growing competition and stringent environmental regulations, it represents a natural way for reducing production costs and improving product quality.

Mechanistic models have been utilized for many years for optimal control studies (Park and Ramirez, 1988; Luus, 1991). However, developing full phenomenological models for complex processes is usually very difficult and time consuming if feasible at all. The time and effort needed to develop mechanistic models has tended to limit the applications of mechanistic model based optimal control strategies. To circumvent these difficulties, black-box data-based empirical model have been widely used. The most popular data-based nonlinear modelling technique is artificial neural networks. Neural network models gain their attraction from speed and ease of implementation, wide applicability and abundant knowledge and research that they have been receiving. Neural networks have been widely used in process modelling and control (Morris et al., 1994; Zhang et al., 1998a; Zhang, 2005).

However, the use of neural network model based optimal control strategy is faced with two major challenges. The first challenge is the non-robust performance of neural networks when they are applied to unseen data and the second challenge is the need for powerful global optimization method that can effectively overcome the conventional problem of falling into local minima. Neural network models are highly non-linear and thus are rich in sub-optimal traps that can lock in the traditional gradient-based optimization methods. Therefore, population-based optimization methods such as genetic algorithms (GA) and ant colony optimization (ACO) should be used to overcome this problem.

2. A FED-BATCH BIO-REACTOR

The process under study is a fed-batch reactor for the production of secreted protein using Baker’s yeast as the host organism and is taken from Park and Ramirez (1988). They studied the secretion of foreign protein bioreactor and developed a mechanistic model that describes its dynamics. The process dynamic behavior is described by the following set of differential equations:

\[ \dot{P}_M = A(S)(P_T - P_M) - \frac{q}{V} P_M \]  
\[ \dot{P}_T = B(S)X - \frac{q}{V} P_T \]  
\[ \dot{X} = C(S)X - \frac{q}{V} X \]  
\[ \dot{S} = -Y C(S)X + \frac{q}{V} (m - S) \]  
\[ \dot{V} = q \]

with

\[ A(S) = \Phi(\mu_s) = \frac{4.75C(S)}{0.12 + C(S)} \]
\[
B(S) = f_p(S) = \frac{Se^{-0.5S}}{0.1 + S} \\
C(S) = \mu_x(S) = \frac{21.78S}{(S + 0.4)(S + 62.5)}
\]

In the above equations, \( P_p \) is the amount of secreted protein on a unit culture volume basis, \( P_f \) is the total protein amount on a unit culture volume basis, \( X \) is the culture cell density, \( S \) is the culture glucose concentration, \( V \) is the culture volume, \( q \) is the feed flow rate which is used as the control variable, \( m \) is the glucose concentration of the feed stream, \( Y \) is the yield of glucose per cell mass, \( \phi \) is the protein secretion rate of the host cell, and \( \mu_x \) is the specific growth rate of the host cell.

The initial conditions of the state variables are \( P_p(t_0) = 0.0 \), \( P_f(t_0) = 0.0 \), \( X(t_0) = 1.0 \) g/L, \( S(t_0) = 0.5 \) g/L, \( V(t_0) = 1.0 \) L. The parameters of the model are \( m = 20.0 \) g/L and \( Y = 7.3 \). The feed rate \( q \) is restricted in the range \((0.0, 10.0)\) L/h and the final time is fixed at \( t_f = 15 \) h.

The goal of the optimal control policy is to maximize the total secreted protein at the end of the batch by changing the feeding rate. In this study the control policy (feed rate profile) consists of \( n \) discrete control actions lasting for equal time intervals, i.e. \( U = \{q_1, q_2, \ldots, q_n\} \). The optimization problem can be formulated as:

\[
\max_C J = P_f(t_f) \quad \text{s.t. } 0 \leq q_i \leq 10, \quad i = 1, 2, \ldots, n.
\]

The process was studied by a number of researchers (Park and Ramirez 1988; Tian et al., 2002).

3. ANT COLONY OPTIMISATION

3.1 Ant Colony Optimization

Ant colony optimization can be regarded as part of swarm intelligence (SI). SI systems are made up from a group of simple individuals that, through simple local interactions, have collective behavior and self-organization, without any form of central control over the swarm members (Bonabeau et al., 1999). SI optimization algorithms can be considered as a member of a larger family of metaheuristic or metaphorical algorithms. These algorithms are inspired by physical phenomena like autocatalytic processes in chemistry (such as simulated annealing) or evolutionary biological systems (such as GA). These optimization algorithms share some common features:

- They draw inspiration from nature.
- They are population-based.
- They are stochastic in nature.
- They do not require gradient information to perform their search.

ACO is inspired by the behavior of real ants that are capable of finding the shortest path from a food source to the nest. It is well known that the medium of communication between ants is pheromone trails. A moving ant lays pheromone on the ground as it moves. The other ants can detect the previously laid trails and choose probabilistically either to follow it (reinforcing the existing solution) or to go to a new direction (exploration for a new solution). How can almost blind ants find the shortest path to the food source? Following the illustrations in Fig. 1 (Dorigo and Gambardella, 1997), the following can be observed:

- A. Real ants follow a route between the nest and the food source.
- B. The route is obstructed by an obstacle: ants choose whether to turn left or right with equal probability.
- C. Pheromone is deposited more quickly on the shorter route.
- D. All ants chose the shorter path.

Fig.1. The behaviour of real ants when an obstacle appears in the path (Dorigo and Gambardella, 1997)

The basic mechanisms in ACO that took inspiration from real ant colonies are:

- **Pheromone Update**, whereby the shortest routes are awarded more pheromone.
- **Pheromone Evaporation**, whereby all the routes are decreased in pheromone amount in a process analogous to pheromone evaporation after food exhaustion.
- **Probabilistic decisions**: ants will not always follow the routes rich in pheromone trail but will make a probabilistic decision whether to do so or not. This is important to avoid early convergence to local minima.

3.2 ACO for Continuous and Mixed-Variable Domain

The main idea of ACO for continuous and mixed-variable domain (ACOCM) (Socha, 2004) is that each ant generates a random number according to a certain probability density function (PDF), \( P^i(X) \), for each dimension \( i \). The PDF are constructed from a weighted sum of several normal PDFs. The PDF of a normal (Gaussian) distribution is of the form:

\[
g(x, \mu, \sigma) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}
\]

8408
where $\sigma$ is the standard deviation and $\mu$ is the mean of the distribution. A weighted sum of a mixture of PDFs is denoted as

$$P(x) = G(x, w, \mu, \sigma) = \sum_{j=1}^{k} w_j \cdot g(x, \mu_j, \sigma_j)$$

where $w_j$ is the weight of the PDF kernel $j$. The vectors $w$, $\mu$ and $\sigma$ define the mixture of normal kernel PDFs $P(x')$ (hereinafter referred as the mixture) that is associated with one dimension. A mixture of normal PDFs might look like the one depicted in Fig. 2.

![Diagram of normal kernels PDFs](image)

**Fig. 2.** Example of five normal PDFs and their sum (the resulting mixture of normal kernels) in the interval (-5, 5)

A solution is constructed by stochastically choosing a value for each dimension $i$ from its mixture $P(x')$. The process of generating a random number according to a mixture is done in two stages. First, an ant selects a single normal kernel from the mixture with a probability $p_j$, proportional to $w_j$:

$$p_j = \frac{w_j}{\sum_{i=1}^{k} w_i}$$

Then the ant generates a number according to the selected normal kernel $P_j$ using, for example, a normal random number generator. The idea of constructing the solution makes general layout of ACOCM algorithm very similar to the regular ACO (i.e. ACO for combinatorial problem). The structure of ACOCM is outlined below:

1. **Initialization**: Create the mixtures of normal PDFs.
2. **Solution Construction**: Each ant constructs one solution.
3. **Positive Update**: Reward the distributions that contributed to good solutions.
4. **Negative Update**: Analogous to pheromone evaporation.
5. **Looping**: Repeat steps 2 through 4 until the termination criterion is met.

There are various issues associated with each step of the algorithm. The description given below is only a brief of what is fully documented by Socha (2004).

**Initialization**

The mixtures are initiated with uniformly distributed means as follows:

$$P'(x') = \sum_{j=1}^{k} \frac{1}{k} g\left(x', a + (2j-1) \frac{b-a}{2k}, \frac{b-a}{2k} \right)$$

(13)

**Positive Update**

One way of accomplishing the positive update is by adding a PDF $P(x')$ into the mixture $P(x')$. The mean $\mu'_i$ of the newly created distribution should be equal to the solution component $x'$ used for the update (the best found solution from last iteration). The values of $w'_i$ and $\sigma'_i$ should be chosen based on the quality of the solution used for the update. One suggested option is updating $\sigma'_i$ according to the equation:

$$\sigma'_i = \max \left( \frac{\max(x_{1..m}) - \min(x_{1..m})}{\sqrt{c}}, \epsilon \right)$$

(14)

where $c$ is the current iteration and $\epsilon$ is the targeted accuracy (or tolerance) of the solution.

**Negative Update**

Negative update can be achieved by many ways. The intuitive way is to do the opposite of the positive update, i.e., to remove an existing ‘bad’ kernel from each dimension. This method might be combined by a pheromone evaporation process, whereby the vector of weights $w'$ for each dimension $i$ is reduced (evaporated). A third option to accomplish the negative update might be through increasing the standard deviation $\sigma'_i$ of normal PDF $P(x')$ for each dimension $i$.

The parameters of ACOCM are the number of normal PDF, $k$, and the number of ants $m$. There are no specific rules for choosing the values of these parameters and they are only based on trial. However, a good heuristic guide to start with is to have $m = 3n$ and $k = 2n$, where $n$ is the number of dimensions, and increment until satisfaction.

The algorithm was tested on some benchmark optimization problems (Mathur et al., 2000). The algorithm was run for 50 times for each bench mark problem to consistently and statistically evaluate its performance. The results showed that the success rate for some problems is about 60-80%, especially for Rosenbrock problem, which implies the tendency of algorithm to fall into local minima. On investigating this problem, it was observed that at least for one variable, the mixture of kernels gets unified into one shape, i.e. all kernel PDFs converge into the same mean and standard deviation. When this happens, the algorithm keeps narrowing the search space for that particular dimension because the newly created kernels by Eq(14) are made with less standard deviation $\sigma$. If the optimal solution lies far from this region, then the algorithm is trapped in a local minimum.

A modification to the original algorithm is introduced in this paper to avoid this premature convergence. The modification is made by monitoring the mean standard deviations of the kernels of each dimension. When their spans start to fall below a critical value, the newly created kernel is randomized and not created in the normal procedure expressed by Eq(14). This process is analogous to mutation in GA. With this new modification to the original algorithm, the success rates are
found to be 100% for all the test problems. Tables 1 and 2 give, respectively, the performance of continuous ant colony optimization (CACO) and the modified ACOCM on some benchmark optimization problems (Mathur et al., 2000). In Tables 1 and 2, $m$ is the number of ants, $k$ is the number of normal PDFs for each $n$ dimensions and $\varepsilon$ is the required accuracy. The number of function evaluations is averaged over 50 runs of each test problem to allow for greater consistency in the results. The performance of the modified ACOCM (indicated by the average number of function evaluations) is highly promising and surpasses that of the CACO.

**Table 1. CACO performance on the benchmark problems**

<table>
<thead>
<tr>
<th>Test function</th>
<th>$n$</th>
<th>$\varepsilon$</th>
<th>Number of function evolutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sphere Model</td>
<td>6</td>
<td>$10^{-2}$</td>
<td>5880</td>
</tr>
<tr>
<td>Goldstein and Price</td>
<td>2</td>
<td>$10^{-4}$</td>
<td>4880</td>
</tr>
<tr>
<td>Rosenbrock</td>
<td>4</td>
<td>$10^{-2}$</td>
<td>10140</td>
</tr>
<tr>
<td>Zakharov</td>
<td>2</td>
<td>$10^{-4}$</td>
<td>5220</td>
</tr>
<tr>
<td>Hartmann</td>
<td>3</td>
<td>$10^{-4}$</td>
<td>4760</td>
</tr>
</tbody>
</table>

**Table 2. ACOCM parameters and results on the benchmark problems**

<table>
<thead>
<tr>
<th>Test function</th>
<th>$n$</th>
<th>$m$</th>
<th>$k$</th>
<th>$\varepsilon$</th>
<th>Number of function evaluations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sphere Model</td>
<td>6</td>
<td>20</td>
<td>12</td>
<td>$10^{-2}$</td>
<td>3638</td>
</tr>
<tr>
<td>Goldstein and Price</td>
<td>2</td>
<td>6</td>
<td>4</td>
<td>$10^{-4}$</td>
<td>1795</td>
</tr>
<tr>
<td>Rosenbrock</td>
<td>4</td>
<td>15</td>
<td>8</td>
<td>$10^{-2}$</td>
<td>4982</td>
</tr>
<tr>
<td>Zakharov</td>
<td>2</td>
<td>8</td>
<td>4</td>
<td>$10^{-4}$</td>
<td>1107</td>
</tr>
<tr>
<td>Hartmann</td>
<td>3</td>
<td>9</td>
<td>6</td>
<td>$10^{-4}$</td>
<td>1120</td>
</tr>
</tbody>
</table>

**4. MODELLING THE FED-BATCH PROCESS USING BOOTSTRAP AGGREGATED NEURAL NETWORKS**

Fig. 3 shows a bootstrap aggregated neural network where several networks are developed to model the same relationship and are combined together. Earlier studies show that an advantage of stacked neural networks is that they can not only give better generalisation performance than single neural networks, but also provide model prediction confidence measures (Zhang et al., 1998b).

Since the ultimate interest is in the total secreted protein at the end of the batch (hereinafter termed as process output), it is intuitive to try to relate the control actions to the process output. Therefore the neural network model will predict the output ($P_{sd}(t_j)V(t_j)$) using the set of $n$ control actions as model inputs. Mathematically, $Y = f(U)$, where $Y = P_{sd}(t_j)V(t_j)$ and $U = [q_1, q_2, \ldots, q_n]$. For ease of implementation, we take the number of regions that constitutes the feeding profile to be $n = 10$. Restricting the control profile to 10 equal stages was found not to degrade the maximum achievable output.

**Table 3. MSE of an aggregated network and a single network**

<table>
<thead>
<tr>
<th></th>
<th>Single network</th>
<th>Aggregated network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training and validation data</td>
<td>0.6570</td>
<td>0.5990</td>
</tr>
<tr>
<td>Unseen testing data</td>
<td>14.5509</td>
<td>2.9952</td>
</tr>
</tbody>
</table>

**5. RELIABLE OPTIMAL CONTROL BASED ON BOOTSTRAP AGGREGATED NEURAL NETWORKS**

It is known that data-based modeling technique work well for the range of data that it is trained with. Neural network is not any better and it only learns from the examples that it is trained with and it might perform very poorly if it is used to
predict out of its “familiarity” regime. We only trained our network in the region around the nominal optimal control profile. Thus when it is used to predict far from this region, it will highly probably give misleading results or, put in another way, there will probably be a high degree of model-plant mismatch. Ant Colony Metaheuristic used for the optimization of our model finds the global optimum by searching the entire problem domain including regions where the neural network model is not very reliable. The end result might be obtaining a feeding policy that is “optimal on the model” but not “optimal on the process”. This problem can be solved using bootstrap aggregated neural networks by incorporating model prediction confidence in the optimization objective (Zhang, 2004).

6. RESULTS AND DISCUSSIONS

Now we can use ACO to maximize the output of the neural network. Although we based our control strategy on aggregated neural networks, we will start by using a single neural network and demonstrate its limitations. It, hence, becomes clearer why using bootstrap aggregated networks is significantly more advantageous.

Using a single neural network to model the fed-batch reactor and using ACOCM to maximize the output, the following results were obtained:

\[ q = (1.9382 \quad 0.0133 \quad 0.3444 \quad 2.8155 \quad 0.0705 \quad 2.8851) \]

\[ P_{d(t)}V(t) = 30.0246 \]

However, when this control policy (i.e. \( q \)) was implemented on the mechanistic model, the resulting amount of product is \( P_{d(t)}V(t) = 4.8823 \), which is very different from the neural network prediction. This confirms the earlier statement that neural networks perform badly outside its training regime. It can be concluded that using a single feed forward neural network to model the process is very far from being acceptable. In general, implementing a single neural network in an optimal control study may well degrade the obtained optimal control policy and might cause serious consequences if applied in practice.

Much better results are obtained when using the control strategy based on:

- Aggregated network that intrinsically has better generalization ability, i.e. more accurate predictions of unseen data.
- Incorporating the standard error of prediction \( \sigma \) in the objective function to ensure that the obtained control policy is obtained with high reliability.

It is important to choose a proper value for \( \lambda \). Too large value might bias the search towards minimizing the standard error with minor consideration to whether that will maximize the output or not. Too small value will suppress the penalization of ‘bad’ predictions and might result in a control policy that is “optimal on the model” but not “optimal on the process”.

In order to put forward a value for \( \lambda \), we make use of the relative scale of the obtained outputs and the standard prediction error. The distribution of output of 60 batches (Fig. 5a) has a mean of around 27 while the distribution of \( \sigma \) over these batches (Fig. 5b) has mean of approximately 1.4. The ratio of the two means is about 20. This suggests using a weighting of \( \lambda = 20 \) in order to give them equal importance in the objective function. Therefore \( \lambda \) is selected as 20.

The optimization problem given by Eq(17) is solved using the modified ACOCM. The results of optimization are:

\[ q = (0.3925 \quad 0.2897 \quad 0.3834 \quad 0.5946 \quad 0.6557 \]

\[ 2.5098 \quad 0.2942 \quad 0.5923 \quad 0.8095 \quad 0.9089 \]

\[ P_{d(t)}V(t) = 29.5634 \]

Using this control policy (i.e. \( q \)) on the mechanistic model we get \( P_{d(t)}V(t) = 30.7590 \), which is very close to the optimum
output using the mechanistic model, 31.7379. This result proves the effectiveness and reliability of using the control strategy based on aggregated neural networks. The two control policies, the one obtained by neural network model and that obtained using the mechanistic model, are shown in Fig. 6. It can be seen that they are quite close.

![Distribution of the output and standard prediction error of 60 batches](image)

**Fig. 5.** Distribution of the output and standard prediction error of 60 batches

![Comparison between the optimal control policy using mechanistic model and neural network model](image)

**Fig. 6.** Comparison between the optimal control policy using mechanistic model and neural network model

### 7. CONCLUSIONS

A reliable on-line re-optimisation control strategy for batch processes based on bootstrap aggregated neural network models is proposed. In addition to process operation objectives, model prediction reliability offered by bootstrap aggregated neural networks is incorporated as additional optimisation objectives. The proposed batch process optimal control approach based on bootstrap aggregated neural networks and ant colony optimization is very effective and reliable. This was demonstrated on protein secretion reactor, which is a typical batch process. The maximum obtained output matches that obtained using a strategy based on the mechanistic model with an error of only 1%. The main motive for this study is the practical difficulties in building mechanistic models for complex processes, which prohibits the development of optimal control strategies. The presented strategy provides an effective and fast solution, which is most useful when a full mechanistic model is very costly or infeasible. Two versions of ACO for continuous domain were studied: ACOCM and CACO. The performance of ACOCM was found to be faster than that of CACO. The original ACOCM was found to fall into local minima when applied to high dimensional problems. This problem was successfully remedied by introducing a mechanism similar to mutation in GAs.

### REFERENCES


