ISSN 2694-5614

Modelling and Prediction of Compressive Strength of Hydraulic Concrete Structure in Multiproduct Batch Plant Design for Protein Production using Extreme Gradient Boosting Regressor with Grid Search support

Journal of Pharmaceutical Research and Development

Research Article

Youness El Hamzaoui* and Juan Antonio Álvarez Arellano

Facultad de Ingeniería. Calle 56 No. 4 Esq. Avenida Concordia Col. Benito Juárez C.P. 24180 Cd. del Carmen, Campeche, México

Correspondence author

Youness El Hamzaoui
Facultad de Ingeniería. Calle 56 No. 4 Esq. Avenida Concordia Col. Benito Juárez C.P. 24180 Cd. del Carmen, Campeche, México

Submitted : 25 May 2021 ; Published : 10 Jun 2021

Abstract

This work deals with the problem of modeling and prediction of compressive strength of concrete structure in multiproduct batch plant design of protein production found in a chemical engineering process with uncertain demand. Modeling the strength of concrete for this process is very complex. However, it can be solved by minimizing the investment cost. Therefore, the aim of this work is to minimize the investment cost and find out the number and size of parallel equipment units in each stage. For this purpose, it is proposed to solve the problem by using extreme gradient boosting regressor with grid search support (XGBoost), could be interpreted as an optimization algorithm on a suitable cost function, which take into account, the uncertainty on the demand using gaussian process modeling. The results about number and size of equipment's, investment cost, production time, process time and idle times in plant obtained by light gradient boosted trees regressor are the best.

This methodology can help the decision makers and constitutes a very promising framework for finding a set of "good solutions".

Keywords: Strength of concrete; Investment Cost; Extreme gradient boosting regressor with grid search support; Uncertain; Batch Process; Optimal Design.

Introduction

Concrete is a composite material composed of fine and coarse aggregate bonded together with a fluid cement (cement paste) that hardens (cures) over time. In the past, lime based cement binders, such as lime putty, were often used but sometimes with other hydraulic cements, such as a calcium aluminate cement or with Portland cement to form Portland cement concrete (named for its visual resemblance to Portland stone). Many other non-cementitious types of concrete exist with other methods of binding aggregate together, including asphalt concrete with a bitumen binder, which is frequently used for road surfaces, and polymer concretes that use polymers as a binder. Concrete is distinct from mortar. Whereas concrete is itself a building material, mortar is a bonding agent that typically holds bricks, tiles and other masonry units together.

When aggregate is mixed with dry Portland cement and water, the mixture forms a fluid slurry that is easily poured and molded into shape. The cement reacts with the water and other ingredients to form a hard matrix that binds the materials together into a durable stone-like material that has many uses. Often, additives (such as pozzolans or superplasticizers) are included in the mixture to improve the physical properties of the wet mix or the finished material. Most concrete is poured with reinforcing materials (such as rebar) embedded to provide tensile strength, yielding reinforced concrete.

Concrete is one of the most frequently used building materials. Its usage worldwide, ton for ton, is twice that of steel, wood, plastics, and aluminum combined. Globally, the ready-mix concrete industry, the largest segment of the concrete market, is projected to exceed $600 billion in revenue by 2025. This widespread use results in a number of environmental impacts. Most notably, the production process for cement produces large volumes of greenhouse gas emissions, leading to net 8% of global emissions. Significant research and development is being done to try to reduce the emissions or make concrete a source of carbon sequestration. Other environmental concerns include widespread illegal sand mining, impacts on the surrounding environment such as increased surface runoff or urban heat island effect, and potential public health implications from toxic ingredients. Concrete is also used to mitigate the pollution of other industries, capturing wastes such as coal fly ash or bauxite tailings and residue.

Nonetheless, in chemical engineering, there has been an increased interest in the development of systematic method for the design of batch process in specialty chemicals, food products, and pharmaceutical industries (Reklaitis, 1992) [1]. Most processes in the modern biotechnology industry correspond to batch plants and with the rapid development of new products (i.e, both therapeutic and non therapeutic proteins) (Crougham et al. 1997) [2].

The main host for recombinant proteins for many years has been Escherichia coli. However, the developments with yeast cells have grown at a very rapid pace, which has resulted in several important commercial products such as insulin, hepatitis B vaccine, and also more recently, chymosin and protease. The fact that many recombinant proteins made in yeast can be made to be secreted out of the cell and that yeast allows for at least partial glycosilation is an added bonus for this host (Montatgna et al. 2000) [3], therefore, in the optimal design of a multiproduct batch chemical process, the production requirement of each product and the total production time available for all products are specified. The number and size of parallel equipment units in each stage as well as the location and size of intermediate storage are to be determined in order to minimize the investment cost.

The common approach used by previous research in solving the design problem of batch plant has been to formulate it as a mixed integer nonlinear programming (MINLP) problem and then employ optimization techniques to solve it. Robinson and Loonkar (1972) [4] studied the problem of designing multiproduct plants operating in single product campaign mode and with a single unit in each processing stage and they extended the nonlinear programming model to include both the design of discrete equipment size and the selection of the parallel units number, by solving it through the use of heuristics and branch and bound. The same problem was further formulated by Grossmann and Sargent (1979) as a (MINLP) model [5]. Knopf et al. (1981) and Yeh and Reklaitis (1987) accounted for the presence of semicontinuous units [6, 7]. Voudouris and Grossmann (1992) proposed reformulations of the previous design models where discrete size are explicitly accounted for [8].

Many works in the literature on batch process design are based on expressions that relate the batch sizes linearly with the equipment sizes. Also, the processing times are usually expressed as nonlinear functions of the batch size. Given certain restrictions on these mathematical expressions, the models can be referred to as posynomials, which possess a unique optimum (Grossmann and Sargent. 1979) [5]. Salomone and Iribarren (1992) proposed posynomial models in which the constants are obtained as a result of the optimization of the process decision variables with simplified models [9]. Salomone et al. (1994) generalized the approach by allowing the process parameters to be generated from either experimental data and/or dynamic simulation [10]. Because of the NP-hard nature of the design problem of batch plant, unbearable long computational time will be induced by the use of Mathematical Programming (MP) when the design problem is somewhat complicated. Severe initial values for the optimization variables are also necessary. Moreover, with the increasing size of the design problem, MP will be futile. Heuristics needs less computational time, and severe initial values for optimization variables are not necessary, but it may end up with a local optimum due to its greedy nature. Also, it is not a general method with respect to the fact that special heuristic rules will be needed for a special problem.

In economics, demand is the desire to own something and the ability to pay for it (Henning et al. 1988) [11]. The term demand is also defined elsewhere as a measure of preferences that is weighted by income, but the market demand for such products is usually changeable, and at the stage of design of a batch plant, it is almost impossible to get the precise information on the future product demand over the lifetime of the plant. However, decisions must be made about the plant capacity. This capacity should be able to balance the product demand satisfaction. In the conventional optimal design of a multiproduct batch chemical plant (Hasebe, 1979) [12], a designer specifies the production requirements for each product and total production time for all products (Floudas, 2005) [13]. The number required of volume and size of parallel equipment units in each stage is to be determined in order to minimize the investment cost.

Basically, batch plants are composed of items operating in a discontinuous way. Each batch then visits a fixed number of equipment items, as required by a given synthesis sequence (so-called production recipe) (Ponsich et al. 2007) [14]. For instance, the design of a multiproduct batch chemical plant is not only to minimize the investment cost, but also to minimize: the operation cost, total production time, and to maximize: the revenue, flexibility index, simultaneously (Aguilar et al. 2005) [15].

On the other hand, the key point in the Design of Multiproduct Batch Plants (DMBP) under uncertain demand. The market demand for products resulting from the batch industry is usually changeable, and at the stage of conceptual design of a batch plant, it is almost impossible to obtain the precise information on the future product demand over the plant lifetime. Nevertheless, decisions must be made about the plant capacity. This capacity should be able to balance the product demand satisfaction and extra-capacity in order to reduce the loss on the excessive investment cost or than on market share due to the varying product demands.

The most recent common approaches treated in the dedicated literature represent the demand uncertainty using fuzzy concepts with trapezoidal fuzzy number which can be represented by a membership function (Bautista et al. 2007) [16]. Yet, this assumption does not seem to be always a reliable representation of reality, because in practice we can't get whole linguistics parameters about the uncertainty demand, such as perceptions, seasons and offers. For this reason an alternative treatment of the imprecision is constituted by using gaussian process modeling that represents the "more or less possible values".

In this work, we will only consider multiproduct batch plants, which means that all the products follow the same operating steps (Cao et al. 2002) [17], the structure of the variables are the equipment sizes and number of each unit operation that generally take discrete values.

The aim of this work is to solve the DMBP under uncertain demand using gradient bosting algorithms. The model presented is general, it takes into account all the available options to increase the efficiency of the batch plant design: unit duplication in-phase

and out-phase and intermediate storage tanks.

The paper is organized as follows, section 2 is devoted to the methodology. In section 3 we formulate the problem formulation, including process description. Then in section 4 we report results and discussion with comparative results. Finally the conclusions on this work are drawn.

Methodology

In the 1960s and 1970s witnessed a tremendous development in the size and complexity of industrial organizations. The administrative decision-making has become very complex and involves large numbers of workers, materials and equipment. A decision is a recommendation for the best design or operation in a given system or process engineering, so as to minimize the costs or maximize the gains (Salvendy, 1982) [18]. Using the term "best" implies that there is a choice or set of alternative strategies of action to make decisions. The term optimal is usually used to denote the maximum or minimum of the objective function, and the overall process of maximizing or minimizing is called optimization. The optimization problems are not only in the design of industrial systems and services, but also apply in the manufacturing and operation of these systems once they are designed. Including various methods of optimization, we can mention: MINLP, Monte Carlo Method and Evolutionary algorithms.

Extreme Gradient Boosting Regressor with Grid Search support

The extreme gradient boosting regressor with grid search support is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. When a decision tree is the weak learner, the resulting algorithm is called gradient boosted trees, which usually outperforms random forest. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

The idea of gradient boosting originated in the observation by Leo Breiman that boosting can be interpreted as an optimization algorithm on a suitable cost function [19]. Explicit regression gradient boosting algorithms were subsequently developed by Jerome H. Friedman [20], simultaneously with the more general functional gradient boosting perspective of Llew Mason, Jonathan Baxter, Peter Bartlett and Marcus Frean. The latter two papers introduced the view of boosting algorithms as iterative functional gradient descent algorithms. That is, algorithms that optimize a cost function over function space by iteratively choosing a function as a weak hypothesis that points in the negative gradient direction. This functional gradient view of boosting has led to the development of boosting algorithms in many areas of machine learning and statistics beyond regression and classification.

Like other boosting methods, gradient boosting combines weak "learners" into a single strong learner in an iterative fashion. It is easiest to explain in the least-squares regression setting, where the goal is to "teach" a model F to predict values of the form.

$$\hat{y} = f(x)$$

by minimizing the mean squared error

$$\frac{1}{n}\sum_i(\hat{y}_i - y_i)^2$$

Where i indexes over some training set of size n of actual values of the output variable y :

- $\hat{y}_i$ = the predicted value F(x)
- y_i = the observed value
- n the number of samples in y

Now, let us consider a gradient boosting algorithm with M stages. At each stage m ($1 \leq m \leq M$) of gradient boosting, suppose some imperfect model Fm for low m, this model may simply return $\hat{y}=\bar{y}$ where the $\bar{y}$ is the arithmetic mean of y.

In order to improve Fm the algorithm should add some new estimator, $h_m(x)$. Thus

$$F_{m+1}(x) = F_m(x) + h_m(x) = y$$

Or, equivalently,

$$h_m(x) = y\text{-}F_m(x)$$

Therefore, gradient boosting will fit h to do residual $y\text{-}F_m(x)$. As in other boosting variants, each $F_{(n+1)}$ attempts to correct errors of its predecessor F_m

A generalization of this idea to loss functions other than squared error, and to classification and ranking problems, follows from the observation that residuals $h_m(x)$ for a given model are the negative gradients of the mean squared error (MSE) loss function (with respect to $F(x)$):

$$L_{MSE} = \frac{1}{2}(y - F(x))^2$$

$$h_{m(x)} = -\frac{\partial L_{MSE}}{\partial F} = y - F(x)$$

So, light gradient boosted trees regressor could be specialized to a gradient boosting machine, and generalizing it entails "plugging in" a different loss and its gradient.

Algorithm

Gradient Boosting Machines (or Generalized Boosted Models, depending on who you ask to explain the acronym 'GBMs') are an advanced algorithm for fitting extremely accurate predictive models. GBMs have won a number of recent predictive modeling competitions and are considered by many data scientists to be the most versatile and useful predictive modeling algorithm. GBMs require very little preprocessing, elegantly handle missing data, strike a good balance between bias and variance, and are typically able to find complicated interaction terms, making them a useful "Swiss army knife" of predictive models.

GBMs are a generalization of Freund and Schapire's adaboost algorithm (1995) that handles arbitrary loss functions [21]. They are very similar in concept to random forests, in that they fit individual decision trees to random re-samples of input data, where each tree sees a bootstrap sample of the rows of the dataset and N arbitrarily chosen columns, where N is a configurable

parameter of the model. GBMs differ from random forests in a single major aspect: rather than fitting the trees independently, the GBM fits each successive tree to the residual errors from all the previous trees combined. This is advantageous, as the model focuses each iteration on the examples that are most difficult to predict (and therefore most useful to get correct).

Due to their iterative nature, GBMs are almost guaranteed to overfit the training data, given enough iterations. Therefore, the 2 critical parameters of the algorithm are the learning rate (or how fast the model fits the data) and the number of trees the model is allowed to fit. It is critical to tune one of these 2 parameters, and when done correctly, GBMs are capable of finding the exact point in the training data where overfitting begins, and halt one iteration prior to that point. In this manner GBMs are usually capable of squeezing every last bit of information out of the training set and producing a model with the highest possible accuracy without overfitting.

Extreme Gradient Boosting (XGBoost) is a very efficient, parallel version of GBM that has won a large number of Kaggle competitions. The base algorithm is very similar to GBM in R or in Python, but it has been heavily optimized and tweaked for faster runtimes and higher predictive accuracy.

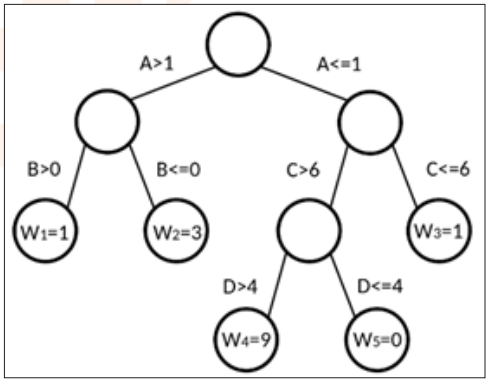

Figure 1: Demonstration of a Decision Tree.

However, the algorithms of Tree Split Finding with Missing Value is described as follow:

Input: I, instance set of current node
Input: $I_k = \{i \in I | x_{ik} \neq \text{missing}\}$p
Input: d, feature dimension
$gain \leftarrow 0$
$G \leftarrow \sum_{i \in I} g_i, H \leftarrow \sum_{i \in I} h_i$
for $k = 1$ ***to*** m **do**
  enumerate missing value goto right
  $G_L \leftarrow 0,\ H_L \leftarrow 0$
  for j *in sorted*$(I_k,$ *ascent order by* $\mathbf{x}_{jk})$ **do**
    $G_L \leftarrow G_L + g_j,\ H_L \leftarrow H_L + h_j$
    $G_R \leftarrow G - G_L,\ H_R \leftarrow H - H_L$
    $gain \leftarrow \max(gain, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda})$
  end
  enumerate missing value goto left
  $G_R \leftarrow 0,\ H_R \leftarrow 0$
  for j *in sorted*$(I_k,$ *descent order by* $\mathbf{x}_{jk})$ **do**
    $G_R \leftarrow G_R + g_j,\ H_R \leftarrow H_R + h_j$
    $G_L \leftarrow G - G_R,\ H_L \leftarrow H - H_R$
    $gain \leftarrow \max(gain, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda})$
  end
end
Output: Split and default direction with max gain

Reduction by introducing the split is calculated by

$$gain = \frac{1}{2}\left[\frac{(\sum_{i\in I_L} g_i)^2}{\sum_{i\in I_L} h_i + \lambda} + \frac{(\sum_{i\in I_R} g_i)^2}{\sum_{i\in I_R} h_i + \lambda} - \frac{(\sum_{i\in I} g_i)^2}{\sum_{i\in I} h_i + \lambda}\right] - \gamma$$

Loss Function

The XGBoost regressor uses least-squares loss by default, but can also use: tweedie loss for zero-inflated positive distributions, poisson loss for count problems, and gamma loss for right skewed positive distributions.

Early Stopping Support

Early stopping is a method for determining the number of trees to use for a boosted trees model. The training data is split into a training set and a test set, and at each iteration the model is scored on the test set. If test set performance decreases for 200 iterations, the training procedure stops and the model returns the fit from the best tree seen so far. The approach saves time by not continuing past the point where it is clear that the model is over fitting and further trees will not result in more accuracy.

Note that the early stopping test set uses a 90/10 train/test split within the training data for a given model. For example, a 64% model on the Leader board will internally use 57.6% of the data for training, and 6.4% of the data for early stopping. A 100% model on the Leader board will internally use 90% of the data for training and 10% of the data for early stopping. Since the early stopping test set was used for early stopping, it cannot be used for training. This limitation also applies to grid search: within the grid search train/test split, the model will use a 90/10 train/test split for early stopping.

Grid Search Support

Grid search is supported in this task. During training, grid search is run to estimate the optimal model parameter values that yield the best performance (evaluated by the configured loss function). The grid search runs on a 70/30 train/test split within the training data; the estimated score uses 30% of the training data split. After the grid search completes and the best tuning parameters are found, the final model is retrained on 100% of training data. Validation scores of the final model are different from the validation scores of the grid search.

Grid search is run on the task parameter with one of the following types: 'intgrid', 'floatgrid', 'listgrid(int)', 'listgrid(float)', 'selectgrid', or 'multi'. Refer to the Parameters section for details of task parameter definitions.

For each grid search parameter, the search space is defined by the parameter values. Refer to the Parameters section for details of task parameter definitions as showed as follow:

1 $F_0(\mathbf{x}) = \arg\min_\rho \sum_{i=1}^N L(y_i, \rho)$
2 For $m = 1$ to M do:
3 $\quad \tilde{y}_i = -\left[\frac{\partial L(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}\right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})},\ i = 1, N$
4 $\quad \mathbf{a}_m = \arg\min_{\mathbf{a},\beta} \sum_{i=1}^N [\tilde{y}_i - \beta h(\mathbf{x}_i; \mathbf{a})]^2$
5 $\quad \rho_m = \arg\min_\rho \sum_{i=1}^N L(y_i, F_{m-1}(\mathbf{x}_i) + \rho h(\mathbf{x}_i; \mathbf{a}_m))$
6 $\quad F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \rho_m h(\mathbf{x}; \mathbf{a}_m)$
7 endFor
end Algorithm

Parameters tuning of Extreme Gradient Boosting

- *learning_rate (lr): floatgrid (default='0.3')*
- *n_estimators (n) : int (default='100')*
- *max_depth (md): intgrid (default='3')*
- *min_child_weight(mcw):floatgrid (default=1.0)*
- *colsample_bytree(cbt):floatgrid (default='1.0')*
- *colsample_bylevel(cbl):floatgrid (default='1.0')*
- *min_split_loss(msl):floatgrid (default='0.01')*
- *num_parallel_tree(npt):intgrid (default='1')*
- *scale_pos_weight(spw): float (default='1')*
- *max_delta_step(mds):floatgrid (default='0.0')*
- *missing_value (mv): float (default='0.0')*
- *base_margin_initialize (base_init): select (default='False')*
- *tree_method (tm): select (default='auto')*
- *max_bin (mb): int (default='256')*
- *mono_up(mono_up):string (default=None)*
- *mono_down(mono_down):string (default=None)*
- *random_state(rs): intgrid (default='1234')*
- *reg_alpha (ra): multi (default='0')*
- *subsample: floatgrid (default=1)*
- *reg_lambda (rl): multi (default='0')*
- *loss (l): select (default='ls')*
- *tweedie_p (p): float (default='1.5')*
- Problem formulation

Assumptions

The model formulation for DMBP's problem approach adopted in this section is based on (Karimi, 1989) [22]. It considers not only treatment in batch stages, which usually appears in all types of formulation, but also represents semi-continuous units that are part of the whole process (pumps, heat exchangers, others).

A semi-continuous unit is defined as a continuous unit alternating idle times and normal activity periods. Besides, this formulation takes into account mid-term intermediate storage tanks, the obligatory mass balance at the intermediate storage stage, which is one of the most efficient strategies to decouple bottlenecks in batch plant design. They are just used to divide the whole process into sub-processes in order to store an amount of materials corresponding to the difference of each sub-process productivity.

This representation mode confers on the plant better flexibility for numerical resolution: It prevents the whole production process from being paralyzed by one limiting stage. So, a batch plant is finally represented as a series of batch stages (B), semi-continuous stages (SC) and storage tanks (T).

The model is based on the following assumptions:

1. The processes operate in the way of overlay.
2. Production is achieved through a series of single product campaigns.
3. Units of the same batch or semi-continuous stage have the same type and size.
4. The devices in the same production line cannot be reused by the same product.
5. The long campaign and the single product campaign are considered.
6. The type and size of parallel items in-or out-of-phase are the same in one batch stage.
7. All intermediate tanks are finite.
8. The operation between stages can be of zero wait or no intermediate tank when there is no storage.
9. There is no limitation for utility.
10. The cleaning time of the batch item can be neglected or included in processing time.
11. The size of the devices can change continuously in its own range.

Model

The model considers the synthesis of (I) products treated in (J) batch stages and (K) semi-continuous stages. Each batch stage consists of (m_j) out-of-phase parallel items of the same size (V_j). Each semi-continuous stage consists of (n_k) out-of-phase parallel items with the same processing rate (R_k) (i.e. treatment capacity, measured in volume unit per time unit). The item sizes (continuous variables) and equipment numbers per stage (discrete variables) are bounded. The (S-1) storage tanks, with size (V_s^*), divide the whole process into (S) sub-processes.

Following the above mentioned notation, DMBP's problem can be formulated to minimize the investment cost for all items:

The investment cost (Cost) is written as an exponential function of the unit size, is formulated in terms of the optimization variables, which represent the plant configuration:

$$Min(Cost) = \sum_{j=1}^{J}(m_j a_j V_j^{\alpha_j}) + \sum_{k=1}^{K}(n_k b_k R_k^{\beta_k}) + \sum_{s=1}^{S}(c_s V_s^{\gamma_s}) \qquad (1)$$

Where aj and α_j, bk and β_k, Cs and γs are classical cost coefficients. Equation (1) shows that there is no fixed cost coefficient for any item. This may be unrealistic and will not tend towards minimization of the equipment number per stage. Nevertheless, this information was kept unchanged in order to compare our results with those found in the literature (Karimi, 1989) [22].

The constraints of the problem

Variable bounding

$$\forall j \in \{1,..,\ j\} \qquad V_{min} \le V_j \le V_{max} \qquad (2)$$

$$\forall k \in \{1,..,k\} \qquad R_{min} \le R_k \le R_{max} \qquad (3)$$

Volume V_j of the items of each batch stage j and treatment capacity R_k of each semi-continuous stage k. However, these variables are not continuous anymore and were discretized with an interval of 50 units between two possible values. This working mode was adopted in a view of realism. Indeed, since equipment manufacturers propose the items following defined size ranges, the design of operation unit equipment does not require a level of accuracy such as real number. Note, however, that the initial bounds on these size variables were kept unchanged, being for batch and semi-continuous, respectively: V_{min} and V_{max}, and R_{min} and R_{max}.

Item number m_j in batch stage j and item number n_k in semi-continuous stage k. These variables cannot exceed 3 items per stage ($m_j \geq 1$, $n_k \leq 3$).

Time constraint

the total production time for all products must be lower than a given time horizon H :

$$H \geq \sum_{i=1}^{I} H_i = \sum_{i=1}^{I} \frac{Q_i}{\mathrm{Prod}_i} \qquad (4)$$

Where is the demand for product i.

Constraint on productivities

the global productivity for product i (of the whole process) is equal to the lowest local productivity (of each sub-process s).

$$\forall i \in \{1,..I\} \quad \mathrm{Prod}_i = Min_{s \in S}\left[\mathrm{Prodlocis}\right] \qquad (5)$$

These local productivities are calculated from the following equations:

- Local productivities for product in sub-process s:

$$\forall i \in \{1,..I\}, \forall s \in \{1,..,S\} \quad \mathrm{Prodlocis} = \frac{B_{is}}{T_{is}^{L}} \qquad (6)$$

- Limiting cycle time for product in sub-process s:

$$\forall i \in \{1,..I\}, \forall s \in \{1,..S\} \quad T_{is}^{L} = Max\left[T_{ij}, \Theta_{it}\right] \qquad (7)$$

- where *Js* and *Ks* are, respectively, the sets of batch and semi-continuous stages in sub-process s.
- Cycle time for product in batch stage j:

$$\forall i \in \{1,..I\}, \forall j \in \{1,..J\} \quad T_{ij} = \frac{\Theta_{i,t} + \Theta_{i(t+1)} + p_{ij}}{m_j} \qquad (8)$$

- Where k and $k+1$ represent the semi-continuous stages before and after batch stage j.
- Processing time of product i in batch stage j:

$$\forall i \in \{1..I\}, \forall j \in \{1..J\} \quad \forall s \in \{1..S\} \quad p_{ij} = p_{ij}^{0} + g_{ij} B_{is}^{\alpha_{ij}} \qquad (9)$$

- Operating time for product in semi-continuous stage k :

$$\forall i \in \{1,..I\}, \forall k \in \{1,..Ks\}, \forall s \in \{1,..S\} \quad \theta_{ik} = \frac{B_{is} D_{ik}}{R_k n_k} \qquad (10)$$

- Batch size of product in sub-process s :

$$\forall i \in \{1,..I\}, \forall s \in \{1..S\} \quad B_{is} = Min\left[\frac{V_j}{S_{ij}}\right] \qquad (11)$$

- Finally, the size of intermediate storage tanks is estimated as the greatest size difference between the batches treated in two successive sub-processes:

$$\forall s \in \{1..S-1\} \quad V_s = Max\left[\mathrm{Prod}S *_{is} (T_{is}^{L} + T_{i(s+1)}^{L} - \Theta - \Theta_{i(s+1)})\right] \qquad (12)$$

Process description

The case study is a multiproduct batch plant for the production of proteins taken from the literature (Montagna et al. 2000) [3]. This example is used as a test bench since short-cut models describing the unit operations involved in the process. The batch plant involves eight stages for producing four recombinant proteins, on one hand, two therapeutic proteins, human insulin (A) and vaccine for hepatitis (B) and, on the other hand, a food grade protein, chymosin (C), and a detergent enzyme, cryophilic protease (D). As illustrate in Figure 3 the flowsheet of the multiproduct batch plant considered in this study. All the proteins are produced as cells grow in the fermenter.

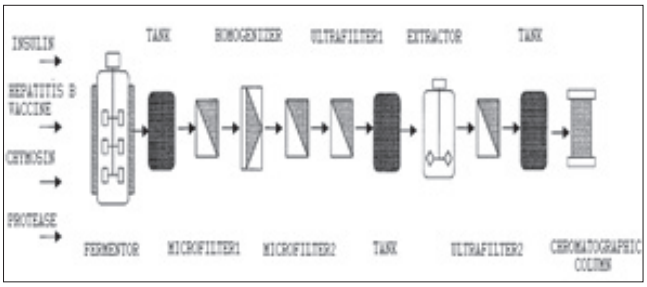

Figure 2: Multiproduct batch plant for protein production.

Vaccines and protease are considered to be intracellular: the first microfilter 1 is used to concentrate the cell suspension, which is then sent to the homogenizer for microfilter 2 is used to remove the cell debris from the solution proteins.

The ultrafiltration 1 step is designed to concentrate the solution in order to minimize the extractor volume. In the liquid–liquid extractor, salt concentration (NaCl) is used solution in order to minimize the extractor volume. In the liquid–liquid extractor, salt concentration (NaCl) is used to first drive the product to a poly-ethylene-glycol (PEG) phase and again into an aqueous saline solution in the back extraction. Ultrafiltration 2 is used again to concentrate the solution. The last stage is finally chromatography, during which selective binding is used to better separate the product of interest from the other proteins. Insulin and chymosin are extracellular products. Proteins are separated from the cells in the first microfilter 1, where cells and some of the supernatant liquid stay behind. To reduce the amount of valuable products lost in the retentate, extra water is added to the cell suspension. The homogenizer and microfilter 2 for cell debris removal are not used when the product is extracellular. Nevertheless, the ultrafilter 1 is necessary to concentrate the dilute solution prior to extraction. The final step of extraction, ultrafiltration 2 and chromatography are common to both the extracellular and intracellular products.

On the other hand, the Figure 1 shows the allocation of intermediate storage tanks. Three tanks have been selected: the first after the fermenter, the second after the first ultrafilter, and the third after the second ultrafilter.

Results and discussion

The typical results obtained by light gradient boosted trees regressor (GBMs) were run 1000 epochs times starting from random initial population guarantees the stochastic nature of the algorithms with demand modeled by gaussian probability distribution, minimizing the cost plant. The results are developed as shown in the following Table 1: Plant Cost, Hi and CPU time. Neverthless, the structure of equipment was illustrated in Table 2.

Min (Cost plant)	1659000 [$]
%Std	0.5%
Hi	6000(h)
CPU time	<5(s)*

**CPU time was calculated to this method on Microsoft Windows 10 Pro Intel(R)D CPU 2.80 Ghz, 2.99 GB of RAM.*

Table 1: Results obtained by gradient boosting algorithms.

Stage	1	2	3	4	5	6	7	8
V_i	22	7	2	2	9	1	1	1
R_i	[-]	15	1	8	100	[-]	16	[-]
V_i	27	[-]	[-]	[-]	2	[-]	1	[-]
m_i	1	1	1	1	1	1	1	1
n_i	1	1	1	1	1	1	1	1

Table 2: Equipment structure according to Table 1

The total production time computed by gradient boosting algorithms is 6000h to fulfill the eventual increase of future demand caused by market fluctuations. The table showed also a very small standard deviation. In addition, gradient boosting algorithms results in a faster convergence (less than one second).

On the other hand, the gradient boosting algorithms allow the reduction of the idle time to the stage. Table 3 shows the idle times obtained by boosting gradient algorithms.

	Unit							
Product	1	2	3	4	5	6	7	8
Insulin	0	0	[-]	[-]	0	0.01	0	0
Vaccine	0	1.93	0.04	0	2.91	0	0.17	0
Clavenosin	0	0.01	[-]	[-]	0	0	0.31	0.17
Protease	0	2.09	0	0	3.07	0	0.5	0

Table 3: Idle Times in Plant with Parallel Units and Intermediate Storage Tanks by gradient boosting algorithms.

From these results, we can see that the results obtained by gradient boosting algorithms are power.

However, since the case study has been taken from Montagna et al (2000) [10], they solved the problem using rigorous mathematical programming (MINLP) which is solved to global optimality (minimize the capital cost $829,500) with implementation of the outer approximation/equality relaxation/ augmented penalty method. However in previous work (Montagna et al 2000) [10], they didn't mentioned anything about CPU time, also in their model, they didn't take into account operation costs. Nonetheless, their model needed a long computational time and require severe initial values to the optimization variables. Montagna et al. (2000) [10], also showed in their paper that the behavior of the demand was completely deterministic. However, this assumption does not seem to be always a reliable representation of the reality, since in practice the demand of pharmaceutical products resulting from the batch industry is usually changeable.

Gradient boosting algorithms performed effectively and gave a solution within 0.5% of the global optimal 1659000 [$], gradient boosting algorithms provided also interesting solutions, in terms of quality as well as of computational time.

Furthermore, gradient boosting algorithms results in a faster convergence. However, gradient boosting algorithms is designed to deal with problems of a more complicated as our problem, DMBP, successfully and the computing time(<5s) is more less than MINLP.

These results are important, because they demonstrate the effectiveness of gradient boosting algorithms in solving the complicated design problem of DMBP, which is due to gradient boosting algorithms searching from population (not a single point), and its parallel computing nature and can be applied to deal with uncertain demand.

Now, some observation about some important aspects in our implication of gradient boosting algorithms and some problems in practice: The most important of all is the method of coding, because the codification is a very important issue when the gradient boosting algorithms is designed to deal with the combinatorial problem, as well as also the characteristics and inner structure of the DMBP.

According to the inner structure of the design problem of multiproduct batch that gives us some clues for designing the above mixed continuous discrete coding method. As it is evident to the results of application, this coding method is well fitted to the proposed problem.

Our experience makes it clear that the parameter's of gradient boosting machine can solve the premature problem effectively and conveniently.

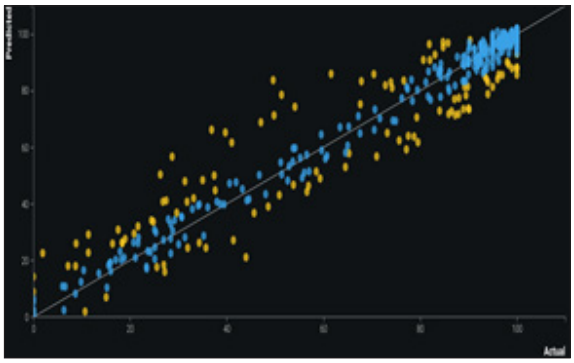

Figure 3: Prediction distribution of crossvalidation about compressive strength of concrete structure.

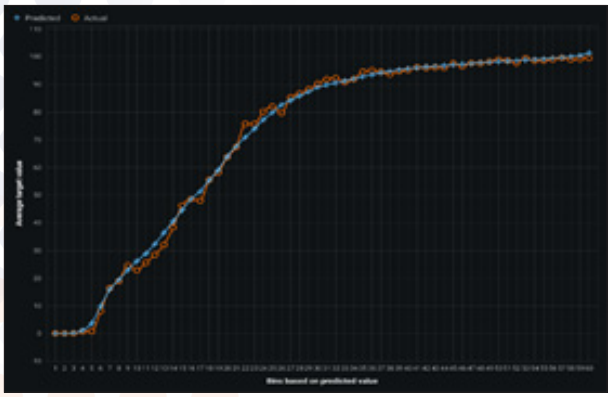

Figure 4: Lift crossvalidation about compressive strength of concrete structure.

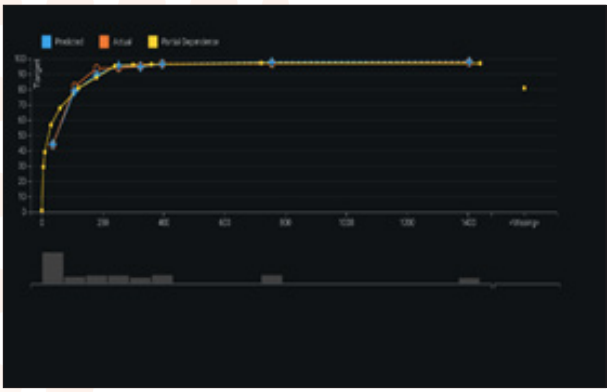

Figure 5: Prediction and simulation of compressive strength of concrete structure.

Conclusions

In this paper, we describe our solution to modelling and prediction of compressive strength of hydraulic concrete structure in multiproduct batch plant design for protein production using extreme gradient boosting regressor with grid search support. We use the extreme gradient boosting regressor with grid search support with early stopping. We also take advantage of feature engineering based on pharmaceutical process to extract more information. Experimental results on the contest data demonstrate the accuracy and effectiveness of the technique proposed by this paper. One of the challenge of multiproduct batch plant for pharmaceutical industry is the large volume of the data. It is also interesting to explore other function classes that are more significantly into pharmaceutical technology. Finally, this framework provides an interesting decision/making approach to improve design multiproduct batch plants under conflicting goals.

Appendix A. Data Set

The experimental data of DMBP based on published data (Datar and Rosen, 1990 ; Petrides et al. 1996 ; Andrews et al. 1999, Asenjo and Patrick, 1990) [23-26]. The plant is divided into sub-processes, consists of six batch stages [B(1-6)] to manufacture in four products A,B,C,D.

The Table shows the values for processing times $\tau_{i,j}(h)$, size factor for the units, cost data, and the production requirement for each product quantifying the uncertainty on the demand. Here, we assume that the demand of products A, B, C and D are uncertain following normal probability distribution function. The data set are summarized in the following Table A1 and Table A2.

	Demand of the product i(kg)		Processing time $\tau_{i,j}(h)$						Size factors(l/kg)					
			B1	B2	B3	B4	B5	B6	B1	B2	B3	B4	B5	B6
A	1500 ± 75		1.15	3.98	9.86	5.28	1.2	3.57	8.28	6.92	9.7	2.95	6.57	10.6
B	1000 ± 50		5.95	7.52	7.01	7	1.08	5.78	5.58	8.03	8.09	3.27	6.17	6.57
C	3000 ± 150		3.96	5.07	6.01	5.13	0.66	4.37	2.34	9.19	10.3	5.7	5.98	3.14
D	6000 ± 300		2.75	4.05	8.02	6.05	1.05	3.54	2.30	5.15	8.05	3.5	5.75	5.45
%			0.4	0.29	0.33	0.3	0.2	0.35						

	Unit price for product i ($/kg)		Coefficients $c_{i,j}$					
	C_P	C_Q	B1	B2	B3	B4	B5	B6
A	0.70	0.08	0.2	0.36	0.24	0.4	0.5	0.4
B	0.74	0.1	0.15	0.5	0.35	0.7	0.42	0.38
C	0.80	0.07	0.34	0.64	0.5	0.85	0.3	0.22
D	0.75	0.05	0.17	0.45	0.25	0.67	0.45	0.25

Operating cost						
	B1	B2	B3	B4	B5	B6
C_E	20	30	15	35	37	18

fermentor = $63400V^{0.6}$
micro-and ultrafilters = $5750V^{0.6}$
homogenizer = $12100cap^{0.75}$
extractor = $23100V^{0.65}$
chromatography=360000$V^{0.995}$
(Volume V in liter)

Horizontal time H
H=6000h
Lower bound =250 l
Upper bound = 10000 l

Table A1: Data used in the problem of batch plant design.

Unit	Size	Cost
Fermenter	V_j (m^3)	$63400.V^{0.6}$
Micro and ultrafilter	$V_{retentate}$(m^3) $V_{permeate}$(m^3) V_{filter}(m^3)	$5750.V^{0.6}$ $5750.V^{0.6}$ $2900.V^{0.6}$
Homogenizer	$V_{holding}$(m^3) Cap(m^3/h)	$5750.V^{0.6}$ $12100.cap^{0.75}$
Extractor	V_{extr}(m^3) $V_{holding}$(m^3)	$23100.V^{0.65}$ $5750.V^{0.6}$
Chromatography column	V_{clucen}(m^3)	$360000.V^{0.995}$
Storage vessel	V_{sto}(m^3)	$5750.V^{0.6}$

Table A2: Cost coefficient.

References

1. Reklaitis GV (1992). Overview of Scheduling and Planning of Batch Process Operations. *NATO Advanced Study Institute-Batch Process Systems Engineering*.
2. Crougham M, Caldwell V, Randlev B, Billeci K, Nieder M (1997). Prediction of culture performance through cell cycle analysis: Potential tool in operations scheduling. *Proceedings of the biochemical engineering*.
3. Montagna JM, Vecchietti AR, Iribarren OA, Pinto JM, Asenjo JA (2000). Optimal design of protein production plants with time and size factor process models. *Biotechnology Programming, 16*(2), 228-237.
4. Robinson JD, Loonkar YR (1972). Minimizing Capital Investment for Multiproduct Batch Plants. *Process Technology International, 17*(11), 861-63.
5. Grossmann IE, Sargent RW (1979). Optimal design of multipurpose chemical plants. I*ndustrial engineering and chemistry process design, 18*(2), 343-48.
6. Knopf FC, Okos MR, Reklaitis GV (1982). Optimum design of batch/semicontinuous processes. *Industrial Engineering and Chemical Process Design Development, 21*(1), 79-86.
7. Yeh NC, Reklaitis GV(1987). Synthesis and sizing of batch/semicontinuous processes: Single product plants. *Computers and Chemical Engineering, 11*(6), 639-654.
8. Voudouris VT, Grossmann IE (1992). Mixed integer linear programming reformulations for batch process design with discrete Equipment sizes. *Industrial Engineering and Chemistry Research, 31*(5), 1315-25.
9. Salomone HE, Iribarren, OA (1992). Posynomial modeling of batch plants: A procedure to include process decision variables. *Computers and chemical engineering, 16*(3), 173-84.
10. Salomone HE, Montagna JM, Iribarren OA (1994). Dynamic simulations in the design of batch processes. *Computers and Chemical Engineering, 18*(3), 191-204.
11. Henning N, Charles A, William P, Robert H (1988). Financial markets and the economy. *New Jersey Press Inc*.
12. Hasebe S (1979). Optimal scheduling and minimum storage tank capacities in a process system with parallel batch units. *Computer and chemical engineering, 3*(1-4), 185-95.
13. Floudas A (2005). Global optimization in the 21st century: Advances and challenges. *Computers and chemical engineering, 29*(6), 1185-1202.
14. Ponsich A, Azzaro-Pantel C, Domenech S, Pibouleau L (2007). Mixed-integer nonlinear programming optimization strategies for batch plant design problems. *Industrial and Engineering Chemistry Research, 46*(3), 854–863.
15. Aguilar-Lasserre AA, Azzaro-Pantel C, Pibouleau L, Domenech S (2007). Enhanced genetic algorithm-based fuzzy multiobjective strategy to multiproduct batch plant design. *Proceedings of the international fuzzy systems association world congress*.
16. Bautista MA (2007). Modelo y software para la interpretación de cantidades difusas en un problema de diseño de procesos. *MBA Thesis, Instituto Técnologico de Orizaba.*
17. Cao DM, Yuan XG (2002). Optimal design of batch plants with uncertain demands considering switch over of operating modes of parallel units. *Industrial engineering and chemistry research, 41*(18), 4616–25.
18. Salvendy G (1982). Industrial engineering handbook. *Wiley & Sons Press Inc.*
19. Breiman Leo. Arcing the edge. Technical Report 486, Statistics Department, *University of California at Berkeley,* 1997
20. Friedman Jerome H (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, 1189-1232
21. Freund Yoav, and Robert E Schapire (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences 55*(1), 119-139.
22. Karimi M (1989). Design of multiproduct batch processes with finite intermediate storage. *Computers and Chemical Engineering, 13*(1-2), 127-39.
23. Datar R, Rosen CG (1990). Downstream process economics in separation processes in biotechnology. *New York Press Inc.*
24. Petrides D, Sapidou E, Calandranis J (1995). Computer-aided process analysis and economic evaluation for biosynthetic human insulin production - A case study. *Biotechnology Bioengineering, 48*(5), 529-541.
25. Andrews BA, Salamanca M, Barria C, Achurra P, Thaysen M, Mancilla M, Asenjo J A (1999). Purification characterization and process considerations of cryophilic proteases of marine origin. *Presented at the Biochemical Engineering XI Conference (United Engineering Foundation)*.
26. Asenjo JA, Patrick I (1990). Large scale protein purification in protein purification applications: A Practical Approach. *Oxford Press Inc*.