New Experimental Design Method for Highly Nonlinear and Dimensional Processes

Guiying Zhang
Dept. of Food Science and Technology, University of California, Davis, CA 95616

Matthew M. Olsen
Dept. of Chemical Engineering and Materials Science, University of California, Davis, CA 95616

David E. Block
Dept. of Viticulture and Enology and Dept. of Chemical Engineering and Materials Science, University of California, Davis, CA 95616

DOI 10.1002/aic.11226
Published online June 25, 2007 in Wiley InterScience (www.interscience.wiley.com).

A novel nonlinear experimental design scheme to identify the optimum of a multidimensional system was computationally evaluated. The method, a hybrid neural network-genetic algorithm coupled with a fuzzy clustering technique, was developed to accumulate former experimental knowledge and suggest optimal operating conditions for future experimentation. Because of the uncertainty associated with complex systems, several response surfaces with various degrees of complexity and dimensionality were used as case studies for evaluating the new methodology to gain insight into the challenges that may be presented by arbitrary experimental optimization problems in practice. The long-term objective of this work is to develop an experimental design scheme that will discover the best process operating conditions for any difficult process optimization problem in as few experiments as possible. The simulation results demonstrate that the algorithm performance is not significantly affected by parameters such as the number of experiments in a batch of experimentation and convergence criteria for the genetic algorithm as well as for various noise levels below 15%. However, higher noise levels, complexity, and dimensionality decrease its performance. Even at higher levels of noise, complexity, and dimensionality, though, the new approach finds significantly better solutions than those found with traditional statistical experimental design methods, with similar or only slightly more experiments. © 2007 American Institute of Chemical Engineers *AIChE J,* 53: 2013–2025, 2007
Keywords: bioengineering, optimization, nonlinear experimental design, hybrid neural network-genetic algorithm

Introduction

As manufacturing processes and objectives become more complex, experimental process optimization is becoming more difficult in all branches of engineering. In an industrial setting, a desirable design of experiments (DOE) technique for process optimization is one that is capable of systematically analyzing the data generated from experiments on processes that include complex interactions among many factors, so that the maximum amount of unbiased information can be extracted from fewest experiments at the various design stages. Eventually process and production goals

Correspondence concerning this article should be addressed to D. E. Block at deblock@ucdavis.edu.

can then be attained from the optimal operating conditions found.

Optimization is a methodology that makes a process (or system) as fully functional or effective as possible. Specifically, optimization refers to finding a minimum, maximum, or target value and involves three important components: objective functions f, variables x, and constraints c. Mathematically, the goal of optimization is to minimize an objective function subjected to constraints on its variables[1]: minimize $f(x)$ under $c(x) \geq 0$ [alternatively, we could also maximize $-f(x)$]. The process to identify the relationships between input space and output space for a given problem is known as modeling. Once a model is constructed, an optimization procedure can then be used to seek the solution of this model. For small-scale problems, classical optimization techniques (e.g., exhaustive search) usually suffice, however, in global optimization or approximation, one of the main obstacles to efficient solving of various practical problems is the multidimensionality of real complex models.[2] The volume of the multivariate search region grows geometrically with dimensionality, leading to the phenomenon described as the "Curse of Dimensionality."[3]

One common type of industrial process, fermentation, is a good example of this kind of complex system. More than 10 process variables are expected, which include batch media ingredients (source, ingredient, and concentration), induction strategies (concentration and timing of inducer addition), nutritional feed (concentrations and rates), and environmental conditions, such as temperature, pH, agitation rate, and dissolved oxygen. Therefore, it is desirable to use an efficient experimental approach to design optimal processes for these kinds of complex systems. However, considering the multidimensionality of fermentation processes, the likely secondary (and higher degree) interactions between variables, and inherent nonlinearity, previous approaches to experimental optimization are likely inefficient at finding truly optimal process conditions.

Most existing process optimization methods fall into two categories, explicit mathematical modeling and statistically designed experiments. Explicit mathematical modeling techniques for search and optimization have a long history of success in many situations, specifically those with low dimensionality and linear or simple nonlinear relationships.[4–6] Most require precise mathematical formulation of problems and assume that comprehensive information about the process is available. This model can then be used to determine the search direction in a deterministic manner at every step of the algorithm.[3] In an arbitrary nonlinear system (e.g., industrial fermentation), however, such fixed form models and detailed information are generally unavailable, indicating that this function-based optimization technique is inappropriate in many real cases.[1]

Traditional statistical experimental design methods, which include factorial or fractional factorial design and efficient response surface methods (RSMs),[7,8] can be applied to optimize many nonlinear experimental systems. Classical RSMs are sequential design strategies for mapping the input–output space and determining variables x that optimize the process in a complex system. While statistical experimental design methods have proven to be useful in many cases, they use only local information from the preceding experimental set (but not from information accumulated at all the previous iterations) and are fundamentally a gradient-based approach, increasing the possibility of being trapped in a local optimum. RSM, generally the most effective of these methods for optimization, is generally of less utility with high dimensional problems where many local minima may exist and the combinatorial nature of this approach is likely to make it impractical in many cases with greater than five process inputs due to significantly higher experimental requirements and corresponding potential limitations in the availability of pilot or process scale equipment and other resources.

All the aforementioned reasons thereby lead to a necessity to develop an effective experimental design method, which can take advantage of new computational methodologies to cope with the complex characteristics of highly nonlinear and dimensional systems, giving good process design directions. Our objective in this work is to develop a general experimental design method that is more efficient and effective than current statistical and model-based approaches for complex processes. Few previous papers have explored the use of a developing database to meet this objective. Chen et al.[9] used standard feed-forward neural networks (NN), nongradient-based search, fuzzy classification, and information theory to address the optimization of low-dimensional complex systems. Recently, Coleman and Block[10] have reported the use of Bayesian regularized NNs with a relative information gain metric to choose optimal experiments for process improvement. Our new approach combines NNs and stochastic optimization to suggest optimal operating conditions for future experimentation based on a developing knowledge base of experimental data. Here we discuss the development of the algorithm, application to various two-dimensional (dim) simulated complex optimization surfaces in the presence of simulated noise, and application to surfaces of up to the 10 dimensions expected of complex fermentation systems.

Methods

Response surfaces used

To make this new approach as capable as possible of addressing arbitrary complex systems that might be encountered in practice, three difficult 2-dim analytical functions that are easy to visualize were used as test surfaces (Figure 1).

Function 1: Modified Himmelblau function (MHF)[9]

$$f(X) = \left(x_1^2 + x_2 - 11\right)^2 + \left(x_1 + x_2^2 - 7\right)^2 + x_1 + 3x_2 + 57 \qquad (1)$$

in which, the input space (X) of this function is uniform in $[-5, 5]$, the average output over the constrained input space is 198.7, the global optimum (opt_g) is 43.3 at the coordinates $(-3.80, -3.32)$, and the three local optima (minima in this case) are 63.5, 54.9, and 65.9, respectively.

Function 2: Maechler Additive function (MAF)[11]

$$f(X) = 1.3356\Big\{1.5(1 - x_1) + \exp(2x_1 - 1)\sin\left[3\pi(x_1 - 0.6)^2\right] + \exp[3(x_2 - 0.5)]\sin\left[4\pi(x_2 - 0.9)^2\right]\Big\} \qquad (2)$$

where X is uniform in [0, 1], with an average output of 2.2, an opt_g of 0.02 at the input coordinates of (0.66, 0.30), and five local optima (0.6, 0.7, 1.2, 1.7, 1.9) as well.

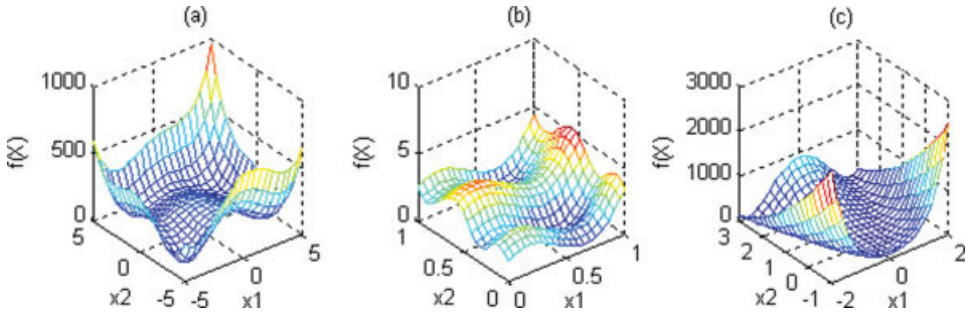

Figure 1. Surfaces of three 2-dim functions.

(a) MHF; (b) MAF; (c) 2-dim MRF. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

Function 3: 2-dim modified Rosenbrock function (MRF)[12]

$$f(X) = \sum_{i=1}^{n-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + (1 - x_i)^2 \right] \tag{3}$$

where $n = 2$, the bound of x_1 is $[-2, 2]$, the bound of x_2 is $[-1, 3]$, the average output is 294.1, and the opt_g is 0 at the point (1, 1).

For optimization problems representative of the higher dimensionality of fermentation processes, in which we are interested, function 3 can also be expanded to higher dimensions[10]: the 5-dim MRF with $n = 5$ and the 10-dim MRF with $n = 10$. In both functions, though, their input spaces are expanded to $[-2\ 2;\ -1\ 3;\ -2\ 2;\ -1\ 3;\ -2\ 2]$ and $[-2\ 2;\ -1\ 3;\ -2\ 2;\ -1\ 3;\ -2\ 2;\ -1\ 3;\ -2\ 2;\ -1\ 3;\ -2\ 2;\ -1\ 3]$, respectively. Their global optima remain the same as that of the 2-dim MRF at the input coordinates of (1,1,1,1,1) and (1,1,1,1,1,1,1,1,1,1), and their average outputs over the entire search space are 3283 and 6851, correspondingly.

Neural network training

In this work a new approach that combines NNs and stochastic optimization (i.e., genetic algorithm, GA) has been proposed to solve the optimization problem discussed earlier. We first describe NN training, followed by our GA and then the integrated optimization algorithm.

An appropriate model is the first step in the process of optimization. Mathematically, a model that is too simplistic cannot give useful insight into practical situations, while a model that is too complex might make the true optimal solutions too difficult to find. A NN-based process model can be developed solely from mapping input–output data.[13] NN techniques have been used in this work to model the evolving experimental database, mainly for their ability to cope with multidimensional systems, capture the nonlinear underlying phenomena contained in the training data set, and eliminate the need to develop a more basic model when prior knowledge of system behavior is not available.[14] For most real process systems, development of actual first principles or mechanistic models will not be possible or practical, and thus, the latter point is critical. Specifically, we have used radial basis function neural networks (RBFNNs) because this type of network is more effective than other types of networks for building a model from a small database that is typical in this work (especially in the initial steps of the algorithm).[13–16] Since defining a "small" database is highly dependent on the dimensionality, complexity, and noise level (NL) of the database, using RBFNN is preferable for cases such as this one, for which all of these factors may not be known a priori.

Structurally, a RBFNN has a similar topology to that of a feed-forward NN,[15] and the training error between the desired output (t_i) of the training set and its actual output (net_i) can also be calculated by the following criterion function[17]

$$J(\underline{w}) = \frac{1}{2} \sum_{i=1}^{n} \|t_i - \text{net}_i\|^2 \tag{4}$$

where, $i = 1, 2, \ldots, n$ in a training set with n data points. However, minimizing the criterion function for training error alone cannot guarantee that a trained network possesses satisfactory generalization performance to meet the end use of modeling. To ensure that a network can be properly used for the purposes of interpolation and extrapolation, early stopping and data splitting have been applied to prevent the network training from poor learning (i.e., underfitting or overfitting). The choice of splitting method is critical, especially when small databases are used for training. To overcome the possible loss of vital information from the experimental (training) data, a statistical technique called k-fold cross-validation is implemented. This method splits the whole data set (with m data points) into k roughly equal subsets using K-means sampling (a clustering technique that groups a dataset by minimizing the within-cluster Euclidean distance).[18] Each NN is trained k times at different numbers of hidden nodes with the $m\left(1 - \frac{1}{k}\right)$ data points rounded towards minus infinity (floor)

$$n = \text{floor}\left[m\left(1 - \frac{1}{k}\right) \right], \quad \text{where}, k = 1, 2, \ldots, m. \tag{5}$$

in the training set, as well as evaluating the constructed network model after training by the left out m/k data in the test set. When $k = m$, this splitting method is also called leave-one-out (LOO) cross-validation,[19] which has been applied in the first batch of experiments in the algorithm developed, as the database at this stage is at its smallest.

Generally, the optimal network structure is then chosen by minimizing the mean-squared error (MSE) between target

output t_s and actual NN output net_s from all the k subsets. The MSE is defined as follows

$$\text{MSE} = \frac{1}{p}\sum_{s=1}^{p}(\text{net}_s - t_s)^2 \tag{6}$$

where $s = 1, 2, \ldots, p$ and $p = m/k$. In the early stage of network training, the MSE of each subset gets smaller with the decrease of $J(\underline{w})$ in the corresponding training set. It then starts to increase as ovefitting of the training set commences.

To maximize neuron output at any given moment, we originally used the MSE as a criterion to select a reasonable spread constant (SC, the distance of an input vector away from a neuron's weight vector to generate an output of 0.5)[20] for small databases. In our experience, a SC that is too large can lead to underfitting (i.e., each neuron is effectively responding in the same large area of the input space), whereas one that is too small may cause overfitting (i.e., failing to generate a response that is strong enough to overlap the active input region).[20]

However, we found that a SC chosen by minimizing the MSE of a test set cannot guarantee identification of an optimal NN model from a very small database. For instance, a correlation coefficient R between the desired functional output t_s and the corresponding actual NN output net_s could be far away from 1 (or even below 0); similarly, an intercept B could be far away from 0 as well, even if the corresponding R is close to one (if systematic bias is present). We then developed a novel training criterion, MBR, to replace the commonly used MSE in the context of k-fold cross-validation

$$\text{MBR} = |1 - R| + \frac{\left|\tan^{-1} M - 45\right|}{45} + |B| \tag{7}$$

where M, B, and R are the slope, the intercept, and the correlation coefficient of the regression line ($t_s = M \times \text{net}_s + B$) from the test set, respectively. For a perfect fit, the MBR will be 0. As model fit diminishes, however, the MBR will increase.

Genetic algorithm

With a model at hand, new optimal operating conditions for a particular optimization criterion can then be found by optimization techniques. Since a NN is a large-scale nonlinear model without explicitly-defined functions, use of numerical optimization techniques is warranted. Unlike standard gradient optimization methods, a GA is an efficient stochastic searching technique that requires no strong assumptions about the form of objective functions.[21] It borrows a vocabulary used in natural genetics, and emulates the evolutionary process by implementing genetic inheritance and a "survival of the fittest" strategy.[21–25] A GA performs a multidirectional search by maintaining a population of potential solutions and encouraging the formation and exchange of information between these directions.[21] It thereby is less likely to get stuck in local minima than gradient approaches. Given these advantages, a pure GA capable of treating both continuous and discrete variables was developed for use in this work, with stopping criteria of maximum generation reached or lack of further fitness improvement.

To satisfy the requirements in different experimental stages (that will be described in the stepwise procedure of the algorithm later), two GA variants were developed in this work. One variant of GA (GA-*v*1) stops searching only when two similar optima are located in two consecutive iterations of the pure GA. Typically, we used a criterion of an output change less than 1–5% of the output range for stopping. This variant is used for the earlier experimental stages when only a poor NN model can be constructed from minimal experimental points available at this stage. The other (GA-*v*2) runs the pure GA four times to get four solutions. It then selects the one with best fitness, which is regarded the optimum; this GA variant is applied to the stages of new nonlinear experimental design (*n*-DOE) where more information has been accumulated from previous experiments. Though a GA is very effective in seeking optimal or near-optimal solutions, there is no guarantee that it can always locate the opt_g of any given problem. To improve the values of the opt_g, a hybrid GA (HGA) was then developed. This HGA combines one of the GA variants with a gradient direct search method.[26] The HGA uses the current optimum (found by a GA variant) as an initial estimate (or starting point) in a gradient-based algorithm to find a more accurate solution (i.e., opt_g) for the given problem; therefore, GA-*v*1 and GA-*v*2 are converted into HGA-*v*1 and HGA-*v*2, respectively.

Truncated genetic algorithm

When a GA converges on an optimum, all the best solutions accumulated are fairly similar. They represent a relatively small region of the search space, compared with the overall input space. To take advantage of the information from the GA while avoiding locating all new experiments around an optimum that may not yet be the true optimum in the developing database and model, we used a truncated genetic algorithm (TGA) to select suboptimal experimental points.

While searching for the optimum of a NN model, a TGA is used to suggest a master list (ML) with d individual solutions, which are selected from different best solutions accumulated by ranking all the solutions of a GA from the population of each generation. Destined to be the next experimental set, this ML then continues to be updated with the increase of generations. Since the average Euclidean distance ($\overline{\text{ED}}$, which is in the range of 0–1, due to the normalized data fed into a NN) between the input space of the best solution so far and those of ($d - 1$) suboptimal solutions (based on the NN model) declines as the GA progresses towards its final solution, a critical distant (CD, where, $0 < \text{CD} \leq \overline{\text{ED}} < 1$) is used as a criterion to stop this GA from further convergence, so that the ML can be diverse enough to provide useful information for constructing the next NN model. Once the $\overline{\text{ED}}$ is smaller than CD, the search of the GA stops, and the ML becomes the next-to-last one.

After selecting a ML from the TGA, we found that some solutions in this ML might still be close to each other in terms of the input space, even though their outputs were actually different from a numerical point of view. To exclude the homogeneous solutions in the ML, a clustering technique was applied to select the most representative individuals. A fuzzy c-means (FCM) clustering is an unsupervised fuzzy

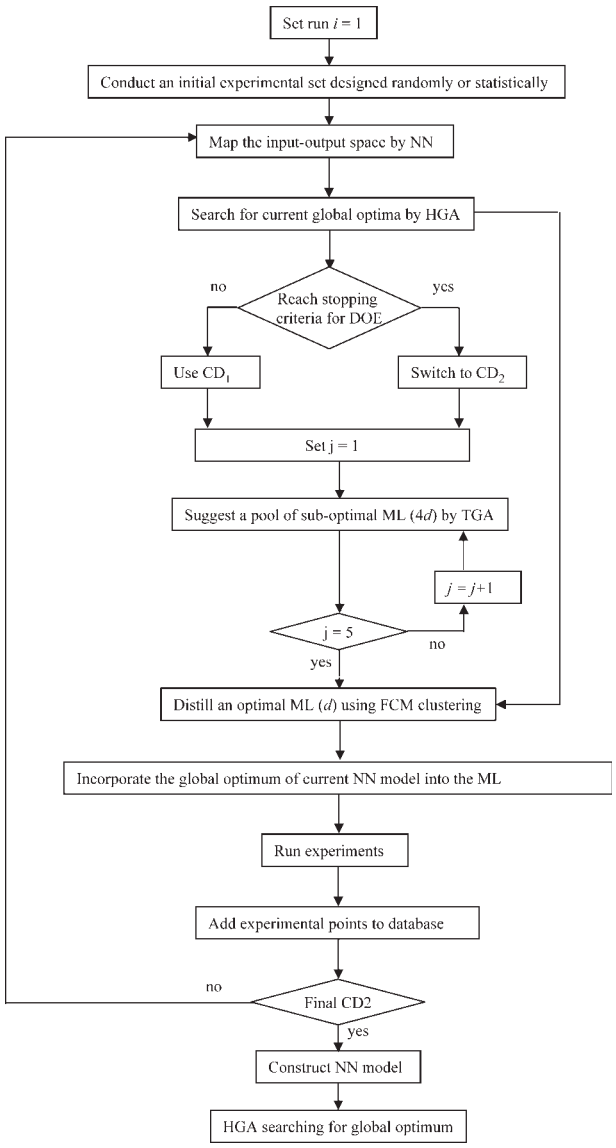

Figure 2. Flowchart of new *n*-DOE.

classification technique. It derives a preset number of clusters (d) from a large data set, by means of an optimal set of fuzzy IF-THEN rules.[9,27] In practice, we run the TGA four times to get four MLs, whose candidates (totally $4d$ solutions) are then used by the FCM clustering to distill d solutions to compose a new ML.

Theoretically, the fitness of all the solutions in the ML up to this point is lower than that of the opt_g. To ensure that all of the useful information is incorporated into the next experiment, the ML is then finalized by replacing the solution having the lowest fitness with the global optimal solution of the current NN model, which has been found by the HGA.

Hybrid neural network—genetic algorithm

A flowchart that describes the hybrid neural network-genetic algorithm (NN-GA) optimization algorithm utilizing the subroutines described earlier can be seen in Figure 2. A stepwise description of the algorithm can also be stated as follows.

- *Step A*. Statistically or randomly design the first experimental set.
- *Step B*. Map the input–output relationship of the database using the NN algorithm.
- *Step C*. Use the HGA-$v1$ in the first two batches (or the HGA-$v2$ beginning with the third batch) to search for the opt_g of current NN model.
- *Step D*. Use a critical distance-1 (CD_1), running the TGA algorithm four times to suggest four suboptimal MLs.
- *Step E*. Distill an optimal ML from these $4d$ experimental candidates by FCM clustering.
- *Step F*. Finalize the ML by incorporating the current model opt_g.
- *Step G*. Run new experiments.
- *Step H*. Iterate between *Step B–Step G* until the stopping criterion for the n-DOE is reached (i.e., two similar opt_g are located from the corresponding NNs of two consecutive batches).
- *Step I*. Use a critical distance-2 (CD_2, a CD which is less than CD_1 and designed for final convergence of the GA) to generate a final set of experimental points that are very close to the current model opt_g. Go to *Step E–Step G* for a final time.
- *Step J*. Train final NN process model.
- *Step K*. Use the HGA-$v2$ to search for the predicted opt_g.

Response surface method

Though RSM is commonly employed in DOE, in practice variations in this methodology of various efficiencies exist.[28,29] A true, efficient RSM[30,31] is one that starts with an initial experimental set (e.g., a central composite design) to construct a best-fit model that is of first degree, second degree, or higher degree. A new set of experiments is then conducted using the center of the previous experimental set as the starting point and then experimentally following along the direction of the steepest descent obtained from the first degree model, or using the minimum calculated from the second or higher degree model. Further batches of experimentation (e.g., a central composite design) are then centered on the new local minimum found. To compare the efficiency and efficacy of our new optimization algorithm with more traditional methods, one of these efficient RSMs (i.e., central composite design integrated with a gradient descent to find the center of the next set of experiments) was followed by simulation, using three functions with different dimensionality. The 2-dim MHF was used as the test case of two variables with full factorials (plus 2 repetitions at each corner and 3 center points) as the initial experimental set. For the 5-dim MRF, ½ fractional factorial initial experimental sets were used. It would be uncommon in practice to use RSM with greater than 5 inputs as it would likely prove to be overly expensive from an experimental point of view as noted earlier. While simulation of a 10-dim RSM was attempted, it proved too computationally intensive for the 10-dim MRF to yield meaningful results; hence, this was not pursued.

Indicators for successful search algorithms

In assessing the performance of the n-DOE algorithm with various parameters, it is important to have well-defined

measures for success. Four different indicators of success were developed for this purpose.

The first indicator is based on the input spaces of an opt_g found by an algorithm variant. The MHF in Figure 1a can be taken as an example. If both x_1 and x_2 are located between -3.0 and -4.1, the corresponding optimum is then considered to be a true opt_g. Though this indicator works well for 2-dim problems, it is more difficult to apply to the case studies with higher dimensionality.

The second indicator is the mean process output of optimal conditions identified by the algorithm. This output can be compared with the target output of a real process (in practice) or the true output of a mathematical model (in this simulation work). However, this indicator is only useful for the comparison among the case studies with the same function, but not those functions with different formulae or different dimensionality.

The third indicator is the mean number of experiments ($\overline{N}_{exp}$) needed to locate an optimum by a certain algorithm variant. It reveals the efficiency of any experimental design method, although the efficacy of an optimization algorithm is not assured by focusing on this indicator.

The last, but perhaps the most important indicator for this work, is the successful decrease percentage (SDP). This indicator is derived from the mean output of all optima located using a particular algorithm ($mean_{opt}$), the mean output ($\bar{t}$) of a mathematical model over the entire search space, and the true opt_g of this mathematical model.

$$SDP_{mean} = \frac{(\bar{t} - opt_g) - (mean_{opt} - opt_g)}{(\bar{t} - opt_g)} \times 100\% \quad (8)$$

This equation represents the degree to which the algorithm has lowered the output from the mean over the entire surface to the true minimum. For this equation, 100% would correspond to finding the exact minimum and 0% would correspond to finding conditions that give the mean output that would be found over the entire input space. For minimization, this assumes that $\bar{t}$ is greater than $mean_{opt}$, which will always be the case for a reasonable model. If we use the median of all optima located using a particular algorithm (med_{opt}) instead of the $mean_{opt}$ to measure the center of a database, however, we get

$$SDP_{med} = \frac{(\bar{t} - opt_g) - (med_{opt} - opt_g)}{(\bar{t} - opt_g)} \times 100\% \quad (9)$$

This SDP indicator is very useful for indicating the efficacy of an experimental design method, especially for facilitating the development of the algorithm. In a real application, however, the opt_g will not be known for an arbitrary experimental system, so this measure would be modified to calculate the degree of targeted product improvement based on the optimum of previous batches and the optimum of the current batch with a tolerance for minimum improvement defined prior to the experimentation based on the economics of the process being optimized.

It should be noted here that the first indicator is not practical in terms of higher dimensionality; therefore, all studies in this work will rely on the last three indicators.

Computing environment

MATLAB® Version 7.0.1 (The Math Works, Natick, MA), as well as two MATLAB® toolboxes (the Neural Network Toolbox and the Statistical Toolbox), were used to compose the algorithms. Eight computers with different capacities (two with Intel (R) Pentium (R) 4 CPU 3 GHz and 2 GB of RAM, two with Intel (R) Pentium (D) 4 CPU 3 GHz and 504 MB of RAM, and four with Intel (R) Pentium (R) 4 CPU 3.4 GHz and 504 GB of RAM) were used for the computational work here.

Results

Implementation of algorithm developed

MHF is a 2-dim surface that can be seen in Figure 1a generated using function 1. As a highly nonlinear function with an opt_g of 43.3, this modified function also has three other local optima located within the bounds $[-5, 5]$ for both variables. Thus, the multioptimality of MHF makes it difficult for classical optimization techniques to locate its opt_g.

To illustrate more graphically how the new optimization algorithm works, we have applied it to the MHF surface in Figure 3, which depicts the progression of the algorithm towards finding the opt_g of MHF with a 10% simulated NL. In this figure, each sub-figure (a) shows the distribution of all experimental points that will be used to construct a NN model against the contour of MHF. Each sub-figure (b) records the information found by searching for the global optimum using a HGA against the contour of the current NN model constructed from those experimental points shown in the corresponding sub-figure (a). Each sub-figure (c) illustrates the ML (i.e., the next set of experiments to be performed) distilled by the FCM clustering technique from the data set suggested by the TGA.

It can be seen that after the first batch of experiments in Figure 3, the initial NN model, as seen in the associated contour plot, is relatively poor, and the locations of corresponding optima for subsequent models move through the regions of local optima. However, from Batch #2 the contour plots of NN models become increasingly better, due to the improvement of generalization abilities resulting from accumulating information. At the same time, the hybrid NN-GA progressively steers its search to the precise location of opt_g.

Investigation of factors critical to the optimization algorithm

Three studies were performed to evaluate factors that may be critical to the performance of the new *n*-DOE proposed. These factors include the number of experiments in the ML (i.e., the number of experiments in the next set of experimentation, N_{ML}), selection of the critical distance that controls TGA convergence, and the effect of noise on algorithm performance. Because of the stochastic nature of the algorithm, multiple runs (100 runs in this work) of each algorithm variant were carried out to assess the variability in the algorithm results. The same simulated optimization surface, MHF, was used for this study. The results are presented in Figure 4.

Originally, we hypothesized that if more experiments in a single batch (i.e., a ML) were performed, better results would be obtained in a shorter period of time. To assess this, algorithm variants with different numbers of experi-

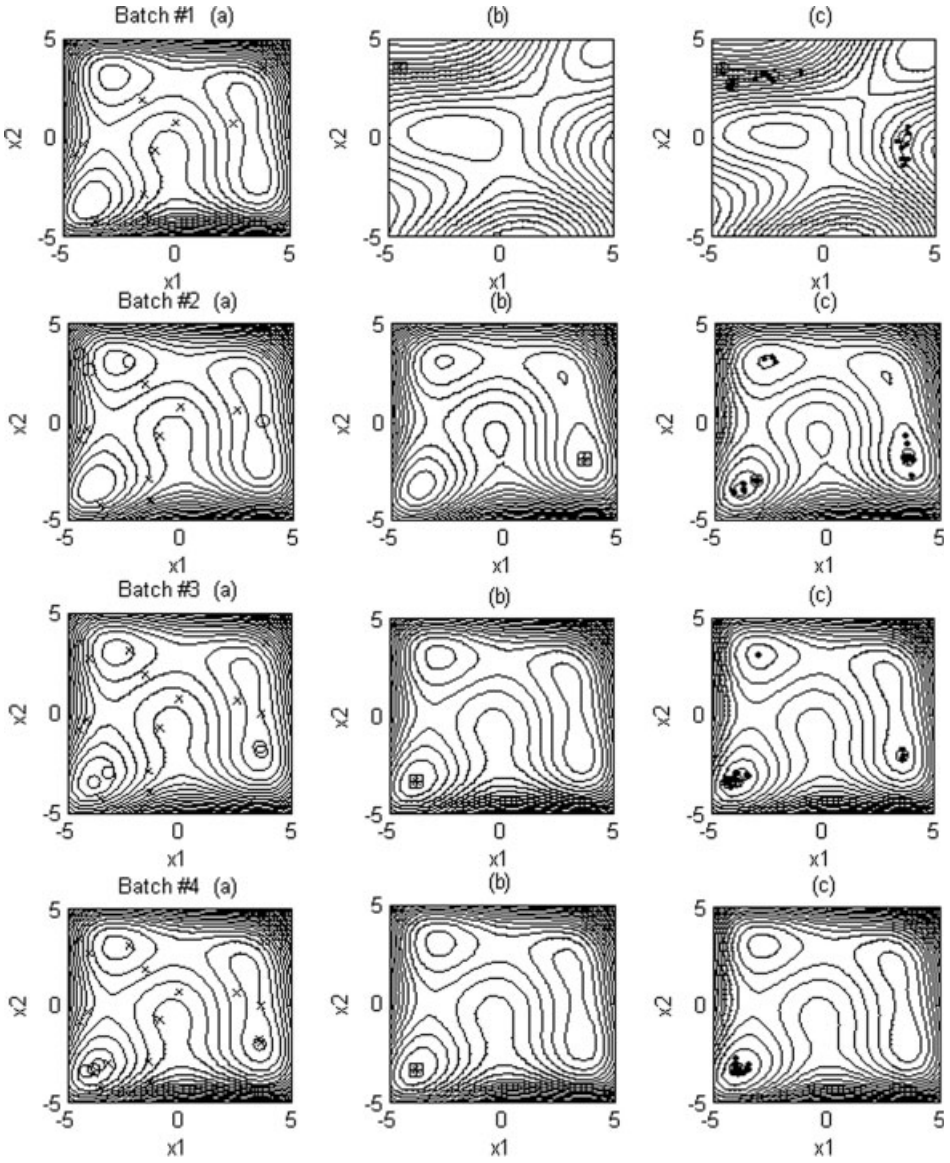

Figure 3. An evolution process of searching the global optimum of MHF by *n*-DOE.

(a) Previous and current experimental points against the contour of MHF: x, previous experimental points; o, current experimental points (from the previous ML). (b) Information about searching optimum by HGA against the contour of current NN models: □, optimum searched by GA; *, final optimum searched by HGA. (c) ML distilled by FCM clustering against current contour of NN model: ●, candidate experimental points; o, experimental points in a ML for next experimental set [the same as those in (a)]. The initial experimental set in Batch #1 (a) is chosen randomly over the search space. The first NN model in Batch #1 (b) was then constructed using the data from this first set of experiments, LOO cross-validation, and criterion of MSE. A TGA with a reasonable CD_1 is applied to suggest a pool of new experiments (total 16 candidates here), followed by the FCM clustering to distill 4 simulated new experimental points for a ML. Again, these 4 experimental points, plus those from the previous batch, are fed to the NN training algorithm to get a new NN model in Batch #2 (b) (the *k*-fold cross-validation and the criterion of MBR are used in this step), which starts a new batch of optimization using HGA, TGA, and FCM clustering. The same procedures are then repeated as shown in Figure 2 until no further improvement is made on the suggested opt_g.

ments in a ML were studied, with the remaining parameters kept constant (statistical design method for the first batch of experimentation, $CD_1 = 0.3$, $CD_2 = 0.05$, and NL = 10%). As shown in Figure 4a, with more experimental points included in a ML, the $mean_{opt}$ decreases as expected. The coefficient of variance ($CV = SD/mean_{opt}$, where, SD is the standard deviation) indicates that inherent variability in the algorithm decreases as the number of experiments in each batch increases. Because of the stochastic initialization of the NN training, a small number of cases result in poor network learning, which generally increases algorithm variability, especially for low numbers of experiments.

Interestingly, for the MHF, the corresponding successful decrease percentage (SDP_{mean}) is not a strong function of the number of experiments in the ML. Essentially, the true optimum was found in most cases. This trend is similar to the one for $mean_{opt}$, but is less pronounced because of the larger range of the SDP_{mean} (SDP_{mean} ranges from 0 to 100%, but the values for the optimizations shown in Figure 4

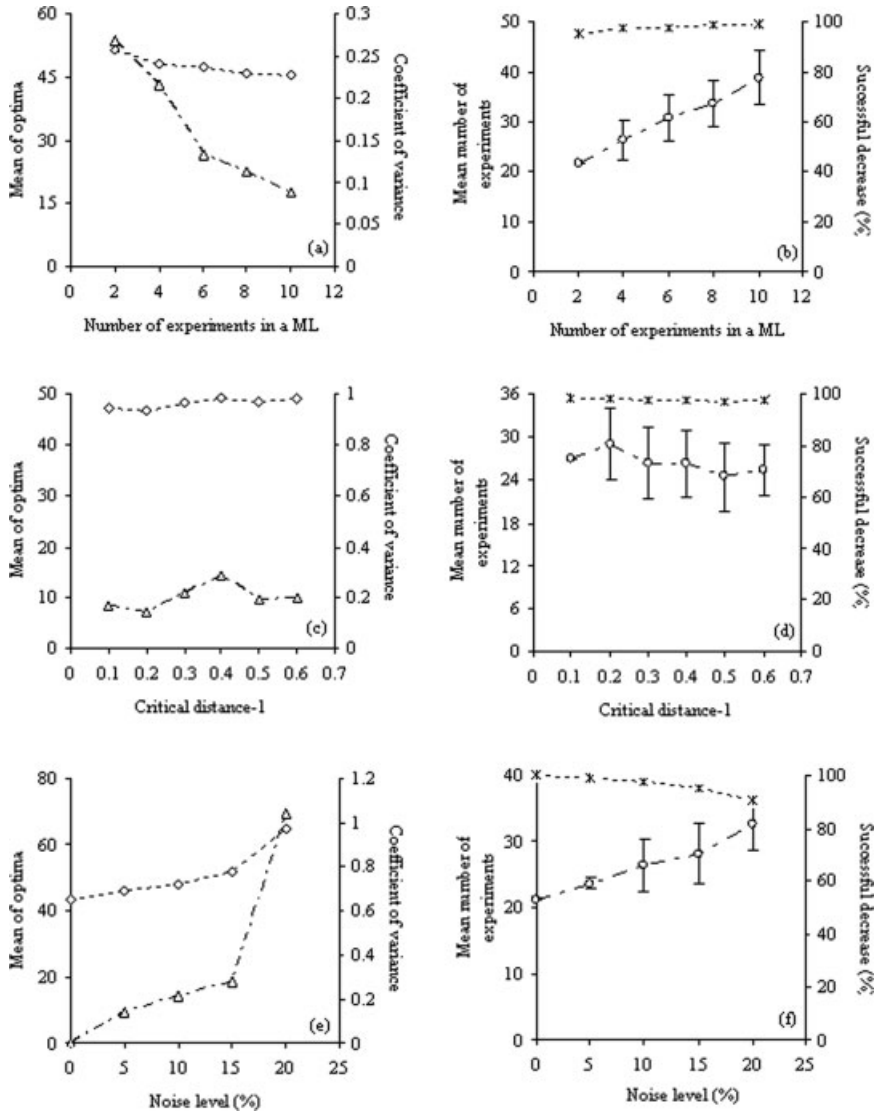

Figure 4. Algorithm performance of 2-dim MHF under different initial conditions.

Performance of *n*-DOE as a function of (a, b) number of experiments in the ML (N_{ML}), (c, d) critical distance-1 (CD_1), and (e, f) noise level (NL). ◇, mean of all optima located from NN model ($mean_{opt}$); Δ, coefficient of variation of the mean of all optima located from NN model ($mean_{opt}$); *o*, mean number of experiments ($\overline{N}_{exp}$); *, successful decrease% (SDP_{mean}).

are all above 95%). It could also be an indication that only a few of the experiments in each batch are contributing significant new information for optimization, though this could change with the increased complexity or dimensionality of problem or equation. On the other hand, the number of experiments needed to reach the optimum increases significantly with the number of experiments in the ML. As a compromise mainly between the number of experiments and algorithm variability, we then chose 4 experiments in a single batch (which has $mean_{opt}$ of 47.2 ± 9.4, $\overline{N}_{exp}$ of 26.4 ± 4.7, and SDP_{mean} of 97.5%), to reach a balance between the resulting productivity and the experimental cost to achieve these results in a real experimental world.

During the course of simulation, we also hypothesized that the CD chosen for the TGA might affect algorithm performance. We then conducted a series of computational experiments that use different CD_1 spanning between 0.01 and 0.9 under the same initial conditions (statistical design method for the first batch of experimentation, $N_{ML} = 4$, $CD_2 = 0.05$, and NL = 10%). We found that algorithm variants with a CD_1 that are too small (e.g., $CD_1 < 0.1$) or too large (e.g., $CD_1 > 0.7$) cannot simultaneously satisfy all the requirements for diversity, convergence, and computing time in a TGA (data not shown). From Figures 4c,d, it can be seen that the performance of the algorithm is not a strong function of CD_1 in the range of $0.1 \leq CD_1 \leq 0.6$ for the MHF surface.

Experimental noise is an important factor in any experimental optimization process. In an industrial fermentation setting, for example, up to 15–20% noise might be encountered, which will obviously complicate the optimization process when potential improvements are of this magnitude or even smaller. To test the capacity of the proposed algo-

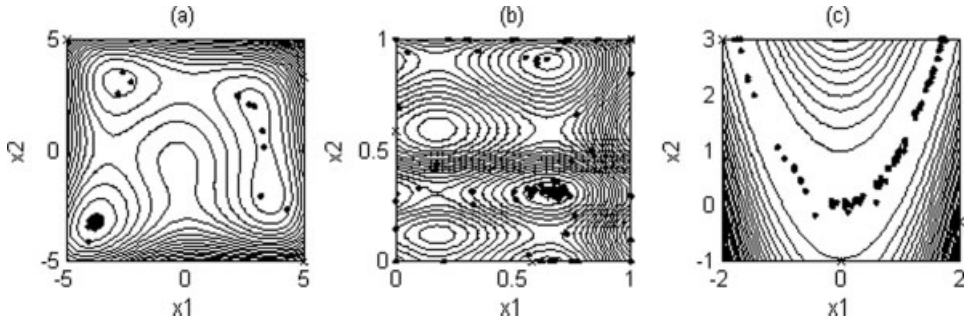

Figure 5. Contour plots of three 2-dim functions with their optima found under the following initial conditions: initial experimental set designed statistically, $CD_1 = 0.3$, $N_{ML} = 4$, NL = 10%, runs = 100.

Results are shown for the (a) MHF, (b) MAF, and (c) 2-dim MRF surfaces. x, initial experimental points; ●, optima located from NN model; *, global optimum.

rithm to operate in the presence of noise, a random error was added to the outputs of this simulation at various levels up to 20%. In practice, if the actual value of a function is t_s, then the functional output after processed by noise (t_n) is

$$t_n = t_s(1 + r) \tag{10}$$

where, r is the random number generated from a normal distribution with population mean of 0 and population standard deviation equal to the NLs 0, 0.05, 0.10, 0.15, and 0.20 (corresponding to NL percents of 0, 5, 10, 15, and 20%). The results are illustrated in Figures 4e,f.

As shown in Figures 4e,f, with the increase in NL, the SDP_{mean} generally drops, while the corresponding $mean_{opt}$ and $\overline{N}_{exp}$ rise. However, the results are not significantly different up to 15% NL. This is a characteristic of the algorithm that will be beneficial for real systems containing noise. A significant degradation of algorithm performance and an increase in the number of required experiments can be observed if NL is more than 15%. However, modifying the stopping criteria (i.e., maximum generation or fitness improvement) of the GA variants at higher NL might improve the SDP_{mean}, although at the expense of increasing $\overline{N}_{exp}$ as well.

Effects of surface complexity on algorithm performance using three 2-dim functions

To test the robustness of this new approach, three 2-dim functions were used to study the effects of surface complexity on algorithm performance. Figure 1 show the surfaces of these functions. Typical examples of the algorithm performance for these three surfaces are illustrated in Figure 5, which give us a basic idea of the distribution of optima located from corresponding NN models in 100 runs of an algorithm variant. These distributions of optima are encouraging because, from a practical point of view, each of the optima found results in a better process in relatively few experiments.

We then investigated if algorithm parameters affected performance in similar ways for all surfaces, as this would be an indicator of the robustness of the algorithm. The main differences observed are with the choice of initial experimental design method used to generate the first data set. We studied two kinds of initialization methods: a random generation method and a statistical design method (Figures 6a,b). For two of the surfaces, MHF and MRF, the initial choice of experiments does not significantly affect the algorithm outcome. However, for the MAF, performance is improved, at least in SDP_{mean}, for a random initial experiment. This is likely a function of the relative position of the five MAF local minima and the statistically-designed experiments. We also examined the effect of CD_1 and initial number of experiments on algorithm performance for all three surfaces, with all other initial algorithm conditions remaining the same. In agreement with the results described earlier for the MHF surface, no significant effects were found for the success criteria studied. Therefore, an acceptable CD_1 (e.g., 0.3) and a reasonable number of initial experiments (e.g., 4) are likely to result in a robust algorithm for all 2-dim surfaces with different complexity.

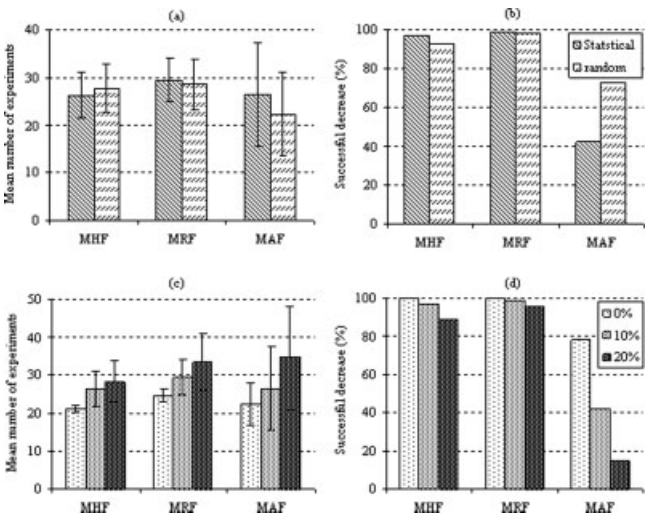

Figure 6. Algorithm performance of the surfaces of three 2-dim functions under the same initial conditions.

Successful decrease% (SDP_{mean}) is shown for (a) two different design methods of the first set of experimentation, where, NL = 10%, $N_{ML} = 4$, $CD_1 = 0.3$; and (b) different noise levels, where, $N_{ML} = 4$, $CD_1 = 0.1$, initial experimental set designed statistically.

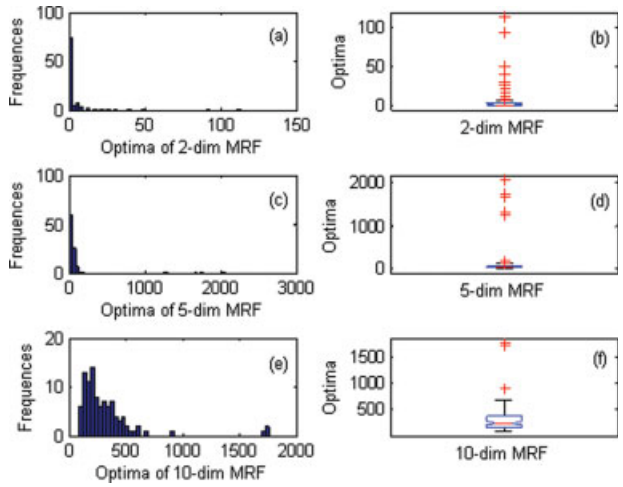

Figure 7. The typical distribution of optima from three functions of different dimensions.

All the initial conditions are as follows: initial experimental set designed statistically, $CD_l = 0.5$, $N_{ML} = 8$, NL = 10%, runs = 100. (a), (c), and (e) display the distribution of optima in bar graph format, while each boxplot shown in (b), (d), or (d) identifies the median of the database, to demonstrate that most of the optima are located within the interquartile (the range between the first quartile and third quartile of all optima), and to locate outliers (+) excluded outside of the lower or upper limits of the box. Histograms are shown for (a) 2-dim MRF, (c) 5-dim MRF, and (e) 10-dim MRF. Boxplots are given for (b) 2-dim MRF, (d) 5-dim MRF, and (f) 10-dim MRF. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

In addition, Figures 6c,d examine the influence of increasing NL (0, 10, 20%) on the algorithm performance with the three different surfaces. As seen in Figure 6, both criteria of success drop to some extent as the NL increases. The $\overline{N}_{exp}$ increases with noise for each surface, but not significantly except for 20% NL with MHF as described earlier. The corresponding SDP_{mean} decrease to different extents for all surfaces, but is more significant for the MAF surface. This is likely to be related to the surface's large number of fairly shallow local minima, since two of the local minima are within 20% of the average output for the entire constrained surface.

Effects of function dimensionality on algorithm performance

To evaluate algorithm performance on surfaces with different dimensions but similar complexity, we applied this new approach to three MRFs with different numbers of variables: 2-, 5-, and 10-dim MRFs. To show the typical outcomes, three histograms of optima found are plotted for different dimensional functions in Figures 7a,c,e. From the frequency distributions in the histograms, we can see that most optima located for the 2-, 5-, and 10-dim MRFs by the *n*-DOE algorithm are within the ranges of 0–0.5, 0–50, and 40–400, respectively. However, it also can be noticed that there exist some significantly larger optima in each function test case, most likely due to occasional poor learning of the NN models. To facilitate comparison, generally we can remove those data whose optima are greater than three standard deviations from the $mean_{opt}$.[32] Compared with the original databases, smaller $mean_{opt}$ result from this treatment. However, we can also turn to a more robust criterion, median, for this analysis. A boxplot method[32] was applied to each database to locate the median of the optima found for each function, as well as the relevant outliers (Figures 7b,d,f). From the information available in these sub-figures, it is obvious that, the median of optima located from NN models increases with increasing dimensionality, even though the true opt_g for each function is 0.

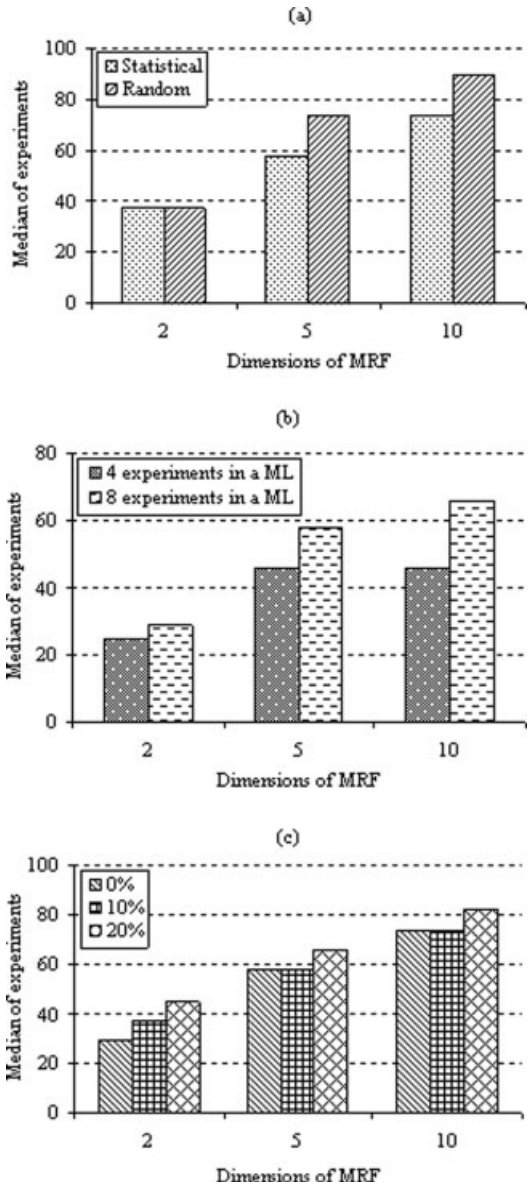

Figure 8. Algorithm performance for functions of different dimensions.

(a) Effects of two different design methods of the first set of experimentation on median number of experiments (med_{exp}), where, NL = 10%, $N_{ML} = 8$, $CD_l = 0.5$; (b) Effects of the number of experiments in a ML (N_{ML}) on med_{exp}, where the initial experimental set is designed statistically, NL = 0%, $CD_l = 0.3$; (c) Effects of noise levels on med_{exp}, where the initial experimental set is designed statistically, $N_{ML} = 8$, $CD_l = 0.5$.

Table 1. Comparison of the *n*-DOE with Tradition Experimental Design (an Efficient RSM) at 10% Noise Level Using Three Functions with Different Dimensionality

		Mean Experiments		Mean Optima		Successful Decrease Percentage	
Functions	dims	RSM	*n*-DOE	RSM	*n*-DOE	RSM	*n*-DOE
MHF	2	27 ± 12	26.2 ± 4.7	131 ± 40	48.1 ± 10.3	17.0	97.5
MRF	5	42 ± 15	56.4 ± 11.3	175 ± 212	105.7 ± 437.9	0	96.8
MRF	10	N/A	74.7 ± 18.3	N/A	314.0 ± 254.3	N/A	95.4

To further study and assess the impact of dimensionality on algorithm performance, we examined effects of algorithm parameters at different dimensions with a special focus on mitigating the increased demand for experiments. Figure 8a compares the med_{exp} with different initial experimental design methods at different dimensions. Generally speaking, the results from the random generation method are not significantly different from those using the statistical design method in the case of the low dimensionality (e.g., 2-dim MRF). However, for the 5-dim and 10-dim cases, greater than 20% more experiments are required when the first set of experiments is chosen randomly. Figure 8b was directed at selecting the optimal number of experiments in a ML with all other initial conditions remaining the same. As the number of experiments in a ML increases from 4 to 8, the corresponding med_{exp} of each surface increases, as well, to achieve a similar SDP_{med} (data not shown). Figure 8c illustrates the effects of NL on the med_{exp} at different dimensions. At higher dimensions, the algorithm performance decreases (i.e., med_{exp} increases) only at NLs on the order of 20%.

Comparison of this new approach with RSM

To compare the new optimization algorithm with traditional methods, the performance of an efficient RSM was selected to compare with the performance of the new optimization algorithm, using three functions with 10% NL as test cases (Table 1). The first case study is the 2-dim MHF. Using the efficient RSM, the $mean_{opt}$ is 137 ± 36 with a $\overline{N}_{exp}$ of 24 ± 9. These results are less efficient than those from our new approach, which have a $mean_{opt}$ of 48.1 ± 10.4 with only a slightly higher $\overline{N}_{exp}$ of 26.2 ± 4.7. The second test case is the 5-dim MRF. The results ($mean_{opt}$ and $\overline{N}_{exp}$) of RSM in this test case are also not as good as those from the new approach. This is even clearer when we examine the med_{exp} for the new algorithm, which is 54 as seen in Figure 8c. Moreover, the $mean_{opt}$ and the $\overline{N}_{exp}$ (or med_{exp} = 74 shown in Figure 8c) for 10 dimensions would be considered extremely reasonable for a new process development program in industry.

Discussion

The experimental design scheme developed here is a new approach that incorporates two main numerical techniques, artificial NNs and GAs, to discover optimal operating conditions for nonlinear systems with arbitrary dimensionality in fewer experiments. Using the MHF surface, we demonstrated the effects of algorithm parameters on its performance. We also evaluated the feasibility of this new approach by comparing three difficult 2-dim surfaces with different complexity, as well as three complex surfaces with increasing dimensionality. These optimization problems were chosen because they are likely to be of similar or greater difficulty than most process optimization cases in the real world. Compared with classical experimental design methods, this approach has the advantages of handling highly nonlinear systems, of decomposing multidimensionality, of locating global optima, and of accumulating previous information to determine the numbers and locations of future experimental sets.

During the course of developing the optimization algorithm, RBFNNs were utilized to create better NN models (a critical step in this approach) for the relatively small databases encountered in this work. However, we found that the MSE criterion was not always satisfactory for selecting a RBNN spread constant (which is one of the main parameters in constructing a RBFNN model) for very small databases. We therefore developed the novel training criterion, MBR, for NN modeling to select the best SC in any situation, except when using LOO cross-validation (for this case, only one data point exists in the test set). As expected, in most cases the SC found by minimizing this new criterion MBR is similar to that found by minimizing the criterion MSE, although this new criterion can successfully avoid some NN structures that result in a poor fit stemming from underfitting or overfitting. The improvement in choosing the best SC leads to better NN generalization of a database with limited data. Besides the RBFNN, another approach that might lead to consistently generalized NN models would be to utilize Bayesian regularized NNs or ensemble averaging of NN.[10]

Based on the results of our testing, the algorithm parameters chosen using the 2-dim MHF were fairly robust for other 2-dim surfaces, which are likely to be more complex than typical real-world processes. As shown here, a critical distance, CD_1, in the range of 0.1–0.6 gives acceptable convergence on the optimum in a reasonable number of experiments. For most of the algorithms in this work, a CD_1 of 0.3 was used successfully to find the optima. Similarly, the optimal number of experiments to complete in each batch of experimentation (i.e., the ML) was found to be four for 2-dim and 5-dim. It may, however, be necessary to increase this number to eight as the dimension of the optimization problems gets as high as 10-dim or higher. In general, we also found that a statistically-designed first set of experiments gives better algorithm performance, especially at higher dimensions. This is likely because this type of experiment acquires information over a larger portion of the search space then random experiments.

Experimental noise is an important factor in the efficiency and efficacy of experimental optimization methods. It would not be unusual, for instance, to have 10, 15, or even 20% NL in a fermentation process. Therefore, it is critical that new optimization algorithms perform robustly in the presence of these levels of noise. In evaluating our algorithm with various 2-dim surfaces, as well as higher dimensional surfaces (up to 10-dim), noise did not seem to significantly affect algorithm performance until 20% NL was applied. At this point, the number of experiments needed to find the optimum increased significantly and the optimum found was not as close to the true optimum. However, even in these high noise situations, the algorithm still significantly improved upon the initial starting point for the process. Therefore, both for the algorithm parameters and experimental noise, we have shown that the algorithm developed will be robust.

Many statistical design methods reported in the literature and currently used by industry practitioners are in fact far less efficient than the efficient RSM simulated in this work, which closely approximates a computational gradient optimization algorithm. All the research results presented showed that the RSM is generally of less utility to deal with high-dimensional problems, in which multiple minima are located, while our algorithm gives better optimization with the same or slightly more experiments. Since the RSM used here is significantly more efficient than those typically practiced in industry, we therefore can say that our new *n*-DOE approach is likely to be a good alternative to classical experimental optimization methods.

From an application point of view, the optimization algorithm developed here is very promising for the efficient development of new robust processes in industries where large numbers of factors contribute to process performance and product quality (e.g., biotechnology, pharmaceuticals, food, petroleum refining), by accumulating information from previous experimentation and suggesting an optimal operating region. This methodology would also be applicable to the further optimization of existing processes where historical databases (with discrete and continuous process variables) are present and can be used to build the initial model as described here. Then, new experiments could be suggested from this model to improve on the existing process in an iterative fashion. Depending on the nature of the process variables that are found to be most important, it may also be possible to use the models developed as part of a process control strategy.[33] One of the key issues that remains in applying this work to arbitrary experimental systems with large numbers of experimental variables is the robustness of the NN model performance in the face of potentially unimportant variables.[34] We are currently exploring the integration of data mining techniques (e.g., decision tree analysis) into our algorithm to address this potential issue.

In summary, we have developed and tested, through simulation, a new nonlinear DOE method for experimental optimization of complex systems. This method is robust to various degrees of complexity and dimensionality and generally finds significantly better optima than traditional statistically-designed experiments and RSM with the same or only slightly higher, number of experiments. It is likely that by improving on the NN modeling step, especially in early stages with limited data, or by introducing a means to dynamically reduce the dimensionality of the problem using data mining approaches over the course of successive batches of experimentation, this method will continue to find better optima, but with even fewer total experiments.

Acknowledgments

The authors would like to acknowledge the California Dairy Research Foundation and the UC Discovery Grant program for funding this research, and thank Dr. Matthew C. Coleman for his valuable suggestions on MATLAB® programming for this research.

Notation

B = The intercept of regression line
CD = Critical distance
dim = Dimensional
D = The number of individuals in a master list
DOE = Design of experiments
$\overline{\text{ED}}$ = Average Euclidean distance
FCM = Fuzzy c-means
GA = Genetic algorithm
GA-$v1$ = GA Variant 1
GA-$v2$ = GA Variant 2
HGA = Hybrid GA
LOO = Leave-one-out
M = The slope of regression line
MAF = Maechler additive function
mean_{opt} = The mean of all optima located from NN models
med_{exp} = The median number of experiments
med_{opt} = The median of all optima located from NN models
n-DOE = Nonlinear experimental design
net_s = Actual NN output in the test set
MHF = Modified Himmelblau function
ML = Master list
MRF = Modified Rosenbrock function
$\overline{\text{MSE}}$ = Mean-squared error
$\overline{N}_{\text{exp}}$ = The mean number of experiments
N_{ML} = The number of experiments in a ML
NL = Noise level
NN = Neural network
NN-GA = Neural network–genetic algorithm
opt_g = Global optimum
R = The correlation coefficient of regression line
RBFNN = Radial basis function neural network
RSM = Response surface method
SC = Spread constant
SDP = Successful decrease percentage
SDP_{mean} = The SDP in terms of mean_{opt}
SDP_{med} = The SDP in terms of med_{opt}
$\bar{t}$ = The mean of a mathematical model
t_n = The functional output after processed by noise
t_s = Target output in the test set
TGA = Truncated genetic algorithm

Literature Cited

1. Nocedal WJ, Stephen J. *Numerical Optimization*. New York: Springer, 1999.
2. Dzemyda G, Saltenis V, Zilinskas A. *Stochastic and Global Optimization*. Netherlands: Kluwer, 2002.
3. Spall JC. *Introduction to Stochastic Search and Optimization*. Hoboken, New Jersey: Wiley, 2003.
4. Schepers AW, Thibault J, Lacroix C. *Lactobacillus helveticus* growth and lactic acid production during pH-controlled batch cultures in whey permeate/yeast extract medium. I. Multiple factor kinetic analysis. *Enzyme Microbial Technol.* 2002;30:176–186.
5. Schepers AW, Thibault J, Lacroix C. *Lactobacillus helveticus* growth and lactic acid production during pH-controlled batch cultures in whey permeate/yeast extract medium. II. Kinetic modeling and model validation. *Enzyme Microbial Technol.* 2002;30:187–194.

6. Schepers AW, Thibault J, Lacroix C. Comparison of simple neural networks and nonlinear regression models for descriptive modeling of *Lactobacillus helveticus* growth in pH-controlled batch, cultures. *Enzyme Microbial Technol*. 2000;26:431–445.
7. Li C, Bai JH, Cai ZL, Fan OY. Optimization of a cultural medium for bacteriocin production by Lactococcus lactis using response surface methodology. *J Biotechnol*. 2002;93:27–34.
8. Tellez-Luis SJ, Moldes AB, Alonso JL, Vazquez M. Optimization of lactic acid production by Lactobacillus delbrueckii through response surface methodology. *J Food Sci*. 2003;68:1454–1458.
9. Chen JH, Wong DSH, Jang SS, Yang SL. Product and process development using artificial neural-network model and information analysis. *AIChE J*. 1998;44:876–887.
10. Coleman MC, Block DE. Nonlinear expeimental design using bayesian regularized neural networks. *AIChE J*. 2007;53:1496–1509.
11. Hwang JN, Li H, Maechler M, Martin D, Schimert J. Projection pursuit learning networks for regression. *Eng Appl Artif Intell*. 1992; 5:193–204.
12. Rosenbrock HH. Some general implicit processes for the numerical solution of differential equations. *Computer J*. 1963;5:329–330.
13. Pollard JF, Broussard MR, Garrison DB, San KY. Process identification using neural networks. *Comput Chem Eng*. 1992;16:253–270.
14. Lanouette R, Thibault J, Valade JL. Process modeling with neural networks using small experimental datasets. *Comput Chem Eng*. 1999;23:1167–1176.
15. Ranaweera DK, Hubele NF, Papalexopoulos AD. Application of radial basis function neural-network model for short-term load forecasting. *IEE Proc-Generation Transm Distrib*. 1995;142:45–50.
16. Nandi S, Ghosh S, Tambe SS, Kulkarni BD. Artificial neural-network-assisted stochastic process optimization strategies. *AIChE J*. 2001;47:126–141.
17. Duda RO, Hart PE, Stork DG. *Pattern Classification*, 2nd ed. New York: Wiley, 2001.
18. Jing LP, Ng MK, Xu J, Huang JZ. Subspace clustering of text documents with feature weighting K-means algorithm. *Adv Knowledge Discov Data Mining Proc*. 2005;3518:802–812.
19. Reich Y, Barai SV. Evaluating machine learning models for engineering problems. *Artif Intell Eng*. 1999;13:257–272.
20. Demuth H, Beale M. *Neural Network Toolbox—For Use with MATLAB*, 4th ed. MA: Mathworks, 2003.
21. Michalewicz Z. *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd ed. New York: Springer-Verlag, 1996.
22. Forrest S. Genetic algorithms—principles of natural-selection applied to computation. *Science*. *13* 1993;261:872–878.
23. Joines JA, Culbreth CT, King RE. Manufacturing cell design: an integer programming model employing genetic algorithms. *IIE Trans*. 1996;28:69–85.
24. Reeves C, Rowe J. *Genetic Algorithms-Pinciples and Perspectives—A Guide to GA Theory*. Boston: Kluwer, 2003.
25. Falkenauer E. *Genetic Algorithms and Grouping Problems*. New York: Wiley, 1998.
26. Le Mauff F, Duc G. Designing a low order robust controller for an active suspension system thank LMI, genetic algorithm and gradient search. *Eur J Control*. 2003;9:29–38.
27. Yager R, Filev D. Generation of fuzzy rules by mountain clustering. *J Intell Fuzzy Syst*. 1994;2:209–219.
28. Roberto IC, Sato S, Demancilha IM, Taqueda MES. Influence of media composition on xylitol fermentation by *Candida guiliiermondii* using response-surface methodology. *Biotechnol Lett*. 1995;17:1223–1228.
29. Saval S, Pablos L, Sanchez S. Optimization of a culture-medium for streptomycin production using response-surface methodology. *Bioresour Technol*. 1993;43:19–25.
30. Myers RH, Montgomery DC. *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*, 2nd ed. New York: Wiley, 2002.
31. Cockshott AR, Sullivan GR. Improving the fermentation medium for Echinocandin B production. I. Sequential statistical experimental design. *Process Biochem*. 2001;36:647–660.
32. Samuels ML, Witmer JA. *Statistics for the Life Sciences*, 3rd ed. New Jersey: Pearson Education, Inc., 2003.
33. Coleman MC, Block DE. Retrospective optimization of time-dependent fermentation control strategies using time-independent historical data. *Biotechnol Bioeng* 2006;95:412–423.
34. Subramanian V, Buck KKS, Block DE. Use of decision tree analysis for determination of critical enological and viticultural processing parameters in historical databases. *Am J Enol Viticulture*. 2001; 52:175–184.

Manuscript received Feb. 14, 2007, and revision received May 8, 2007.