Hybrid Differential Evolution for Dynamic Optimization of A Fedbatch Bioreactor Process

Feng-Sheng Wang
Department of Chemical Engineering
National Chung Cheng University
Chia-Yi 621-02, Taiwan
chmfsw@ccunix.ccu.edu.tw

Abstract

A hybrid method of evolutionary algorithms is introduced in this study. The hybrid method includes two additional operations, acceleration and migrating operations. These two operations are used for the improvement of the convergence speed without decreasing diversity among the individuals. The hybrid method incorporated with multiplier updating is introduced to solve a dynamic optimization problem with constraints. This method can use the population size of five to obtain a more satisfactory solution, as compared to the original algorithm of differential evolution and BCPOL solver in IMSL library.

1. Introduction

Evolutionary algorithms (EAs) are a class of stochastic search and optimization methods that include genetic algorithms, evolutionary programming, evolution strategies, genetic programming, and among their variants (Back, et al., 1997). Recently, some engineering optimization problems have been solved by EAs, because EAs are robust and suitable for finding optimum effectively with a smaller probability of falling in local optima than other algorithms. Differential evolution (DE) developed by Stron and Price (1996,1997) is one of the best EAs. This method has been proved a promising candidate to solve real valued optimization problems. In DE, the fitness of an offspring is one-to-one competed to that of the corresponding parent. The one-to-one competition, which is different from the other EAs, has a faster convergent speed. However, this faster convergence results in a higher probability toward finding a local optimum or getting premature convergence. This fact results from diversity of population descending faster during solving progress. This drawback could be overcome by using a larger population. By doing so, a lot of computation time is expended to evaluate the fitness function. This fact is particularly obvious by using DE to solve optimal control problem (Wang and Chiou, 1997). In order to avoid the use of larger population, a hybrid method of DE is introduced in this paper.

Most engineering optimization problems include equality or/and inequality constraints. During the last few years EAs incorporated with several methods for handling such constrained optimization problems. Michalewicz and Schoenauer (1996) have surveyed and compared several constraint-handling techniques. The penalty methods transform the constrained problem into an unconstrained problem by penalizing those solutions, which are infeasible. However, the penalty function methods are easy to program. The main limitation of the penalty functions is the degree to which each constraint is penalized. Powell (1978) has noted that the classical optimization methods including penalty functions have certain weaknesses when penalty parameters are large. Such penalty functions tend to ill behaved near the boundary of the feasible domain where the optimum points usually lie. In this paper, the multiplier updating method is introduced to handle the constrained optimization problems to overcome such drawbacks.

2. Hybrid Differential Evolution

Hybrid differential evolution (HDE) is a simple population based stochastic function method and has extended from the original algorithm of DE (Storn and Price, 1996; 1997). The original algorithm of DE is used to solve the unconstrained nonlinear programming problems. The HDE with penalty function method has applied to solve fuzzy decision making problems (Wang,

et al., 1998). The basic operations of DE are similar to the conventional EAs as listed in Table 1.

By using EAs to optimize a function, the user generally has to decide a good trade-off between convergence and diversity. The convergence means fast descending even to a local minimum. On the other hand, the higher diversity guarantees higher probability of finding a global minimum. This trade-off procedure in DE can be achieved by slightly adjusting a mutation factor ρ_m and the population size N_p. By increasing N_p and simultaneously reducing ρ_m slightly, DE gets more robust. By doing so, a lot of computational time should be expended to evaluate the fitness function. This fact is particularly obvious by using DE to solve optimal control problems due to expending extensive CPU time to solve differential equations. Two additional operations as expressed in Table 1 are embedded in HDE to overcome such drawbacks. The basic operations of HDE are illustrated as follows:

1. Representation and Initialization

The HDE structure is a parallel direct search algorithm, which uses N_p vectors of the decision parameters $\mathbf{u} = (u_j)$, i.e., $\mathbf{U}_i^G \ i = 1,2,...,N_p$ as a population for each generation *G*. The decision vector (referred to chromosome) is represented as $(u_1,...,u_{N_u})$. Here, the decision parameters (genes), u_j, are directly coded by real values within its corresponding bounds. The initial population is randomly selected and should try to cover the entire search space uniformly in the form:

$$\mathbf{U}_i^0 = \mathbf{u}_{\min} + \rho_i(\mathbf{u}_{\max} - \mathbf{u}_{\min}), \quad i = 1,...,N_p \tag{1}$$

where ρ_i is the uniformly distributed random number. Here, $\mathbf{u}_{\min}$ and $\mathbf{u}_{\max}$ are expressed as the lower bound and upper bound of the decision parameters.

2. Mutation

The mutation operation of DE and HDE is the essential ingredient, compared to the other EAs. The mutation operation at the (*G*-1)-th generation begins by randomly selecting two population individuals $\mathbf{U}_j^{G-1}$ and $\mathbf{U}_k^{G-1}$. These two individuals are then combined to form a difference vector $\mathbf{D}_{jk}$. A perturbed individual $\hat{\mathbf{U}}_i^{G-1}$ is therefore generated based on the parent individual $\mathbf{U}_p^{G-1}$ by:

$$\hat{\mathbf{U}}_i^G = \mathbf{U}_p^{G-1} + \rho_m \mathbf{D}_{jk} = \mathbf{U}_p^{G-1} + \rho_m(\mathbf{U}_j^{G-1} - \mathbf{U}_k^{G-1}), \quad i = 1,\cdots,N_p \tag{2}$$

In DE, the mutation factor $\rho_m \in (0,1.0]$ is fixed and set by the user to ensure the fastest possible convergence. However, the mutation factor in HDE is randomly selected at every generation to obtain a more perturbed individual. Figure 1 shows a two-dimensional example that illustrates the mutually different individuals that play a part in the generation of the mutant individual. The difference of two random individuals acts as a search direction in the solving space. The mutation factor selected between zero and one is used to yield a perturbation to ensure the fastest possible convergence. Consequently, the mutant individual in (2) is essentially a perturbed replica of the parent individual. If the population diversity is small, the candidate individuals will rapidly cluster together so that the individuals are unable to be further improved. This fact may result in a premature convergence.

3. Crossover Operation

In order to increase the local diversity at the next population, the crossover operation is performed to reproduce an offspring at the next generation. In this crossover operation, the perturbed individual $\hat{\mathbf{U}}_i^G$ in (2) and the current individual $\mathbf{U}_i^{G-1}$ are chosen by a binomial distribution to generate a new individual. The binomial crossover operation is therefore expressed as

$$u_{ji}^G = \begin{cases} u_{ji}^{G-1}, & \text{if a random number} > C_R \\ \hat{u}_{Ji}^G, & \text{otherwise;} \quad j = 1,...,N_u; i = 1,...,N_p \end{cases} \tag{3}$$

where the crossover factor $C_R \in [0,1]$ is fixed and set by the user. Herein, u_{ji}^G is the *j*-th gene of the *i*-th individual at the *G*-th generation.

4. Selection and Evaluation

In HDE, the evaluation function of an offspring is one-to-one competed to that of its parent. This competition means that the parent is replaced by its offspring if the fitness of the offspring is better than that of its parent. On the other hand, the parent is retained in the next generation if the fitness of the offspring is worse than that of its parent. Two selection steps are performed in this evaluation operation. The first selection step is a one-to-one competition, and the next step is to select the best individual in the population. These steps are expressed as

$$\mathbf{U}_i^G = \arg\min\left(J(\mathbf{U}_i^{G-1}), J(\mathbf{U}_i^G)\right), \quad i = 1,...,N_p \tag{4}$$

$$\hat{\mathbf{U}}_b^G = \arg\min\left\{J(\mathbf{U}_i^G), i = 1,...,N_p\right\} \tag{5}$$

where argmin means the argument of the minima and *J* is the fitness function. From equation (5), we keep the last improvement of individuals and the best individual at each generation.

5. *Acceleration*

The acceleration operation and migrating operation in HDE act as a trade-off operator. The accelerated operation is used to speed up convergence. However, faster convergence usually results in finding a local minimum. The migrating operation is used to escape this local point. A new population of candidate individuals is randomly migrated away from the best individual to the whole search domain. Accordingly, the diversity of the candidates can be retained by the migrating operation.

According to our experience by using DE to solve optimization problems, the best fitness does not descend continuously from generation to generation. It usually descends toward a better next one after several generations. In this situation, the acceleration operation can be used to overcome such a drawback. If the mutation and crossover operations do not improve the best individual at the present generation any longer, the steepest descent method is applied to push the best individual to obtain a next better point. The acceleration operation is, therefore, expressed as

$$\mathbf{U}_b^G = \begin{cases} \hat{\mathbf{U}}_b^G, & \text{if } J(\hat{\mathbf{U}}_b^G) < J(\hat{\mathbf{U}}_b^{G-1}) \\ \hat{\mathbf{U}}_b^G - \rho_a \nabla J, & \text{otherwise} \end{cases} \tag{6}$$

where $\hat{\mathbf{U}}_b^G$ is the best individual obtained from equation (5). The gradient of the objective function, ∇J, can be approximately calculated with finite difference. The step size $\rho_a \in (0,1]$ in (6) is determined with the descent property. At first, ρ_a is set to be one to obtain the new individual $\mathbf{U}_b^N$. The fitness function $J(\mathbf{U}_b^N($ is then compared with $J(\mathbf{U}_b^G($. If the descent property is obeyed, i.e.,

$$J(\mathbf{U}_b^N(< J(\mathbf{U}_b^G(\tag{7}$$

then $\mathbf{U}_b^N$ becomes a candidate at the next generation to replace the worst individual. On the other hand, if the descent property fails, the step size reduces by 0.5 or 0.8, and the steepest descent method is repeated to find $\mathbf{U}_b^N$ until $\rho_a \nabla J$ is small enough or the assigned iterations achieve. As a result, the best fitness should be at least equal to or smaller than $J(\mathbf{U}_b^{G-1})$ as observed from equation (7).

6. *Migration operation*

The rate of convergence can be improved by the acceleration operation. However, faster descending usually results in finding a local minimum or getting premature convergence. In addition, performing the acceleration operation frequently can result in that the candidate individuals gradually clustered around the best individual so that the diversity of the population decreases quickly. Furthermore, the clustered individuals cannot reproduce the next better individuals by the mutation and crossover operations. This fact results in a premature convergence due to the diminishing of the difference vector as observed from equation (2) and (3). As a result, the migrating operation should be performed to regenerate a new population. The new candidates are regenerated based on the best individual $\mathbf{U}_b^G$. The *j*th gene of the *i*th individual is therefore regenerated by

$$u_{ji} = \begin{cases} u_{jb} + \delta_{ji}(u_{j\min} - u_{jb}), & \text{if } \hat{\delta}_{ji} < \dfrac{u_{jb} - u_{j\min}}{u_{j\max} - u_{j\min}} \\ u_{jb} + \delta_{ji}(u_{j\max} - u_{jb}), & \text{otherwise;} \end{cases} \tag{8}$$

where δ_{ji} and $\hat{\delta}_{ji}$ are the uniformly distributed random numbers. Here, $u_{j\min}$ and $u_{j\max}$ are expressed as the lower and upper bounds of the *j*th gene of the decision parameters. This diversified population is then used as the initial decision parameters to escape the local point.

The migrating operation in HDE is performed only if the measure of population diversity fails to match the desired tolerance. The measure ρ_η in this study is defined as

$$\rho_\eta = \left\{ \sum_{\substack{i=1 \\ i \neq b}}^{N_p} \sum_{j=1}^{n} \eta_{ji} \right\} / n(N_p - 1) < \varepsilon_1 \tag{9}$$

where

$$\eta_{ji} = \begin{cases} 1, & \text{if } \left| \{u_{ji} - u_{jb}\} / u_{jb} \right| > \varepsilon_2 \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

where ε_1 and ε_2 are the desired tolerance for the population diversity and the gene diversity with respect to the best individual. Here, η_{ji} is defined as an index of gene diversity. Its value of zero means that the *j*th gene of the *i*th individual closely clusters to the *j*th gene of the best individual.

The migrating operation is performed only if the degree of population diversity is smaller than the desired tolerance ε_1. The degree of population diversity is between zero and one as observed from (9). Its value of zero implies that all genes cluster around the best individual. Conversely, the value of one indicates that the current candidate individuals are a complete diversified population. The desired tolerance for population diversity is accordingly assigned within this region. The tolerance of zero implies that the migrating operation in HDE is switched off, whereas, the tolerance of one implies that the migrating operation is performed at every generation.

Figure 2 shows a numerical example to describe the

concept of the degree of population diversity. Suppose we attempt to minimize a function with two variables, and the tolerance, ε_2, of gene diversity is set to 0.1. Assume that after some generations of HDE, we get the following population

$\mathbf{u}_1 = (1.0, 1.0)$, $\mathbf{u}_2 = (1.05, 3.0)$, $\mathbf{u}_3 = (1.3\ 2.0)$, $\mathbf{u}_4 = (1.5, 1.06)$

Let the first individual $\mathbf{u}_1$ is the best one. The degree of population diversity in this example is explained that two third of genes in this population is dissimilar to the best individual. As presented, fewer dissimilar individuals is harder to explore better offspring by the mutation and crossover operations. This fact results from a premature convergence. Therefore, if the degree of population diversity is smaller than the assigned tolerance, the migrating operation should be performed to regenerate a next diversified population.

3. Dynamic Optimization Problem

A dynamic optimization problem for a fedbatch bioreactor system is considered in this example for illustrating and evaluating the proposed method. The dynamic equations of mass balance for the fedbatch process are expressed as follows:

$$\frac{dx}{dt} = \mu x - \frac{F}{V}x \tag{11}$$

$$\frac{ds}{dt} = -\frac{q_{p_1}}{Y_{p_1/s}}x - \frac{q_{p_2}}{Y_{p_2/s}}x + \frac{F}{V}(s_F - s) \tag{12}$$

$$\frac{dp_1}{dt} = q_{p_1}x - \frac{F}{V}p_1 \tag{13}$$

$$\frac{dp_2}{dt} = q_{p_2}x - \frac{F}{V}p_2 \tag{14}$$

$$\frac{dV}{dt} = F \tag{15}$$

where x is the concentration of cell mass, s is the concentration of glucose, p_1 is the concentration of ethanol, p_2 is the concentration of glycerol, $Y_{p_1/s}$ is the ethanol yield factor, and $Y_{p_2/s}$ is the glycerol yield factor. In this study, the unstructured kinetic models for the specific cell growth and product formation are expressed as following:

$$\mu = \frac{\mu_m s}{K_s + s + s^2/K_{s_I}} \frac{K_{p_1}}{K_{p_1} + p_1 + p_1^2/K_{p_{1I}}} \frac{K_{p_2}}{K_{p_2} + p_2 + p_2^2/K_{p_{2I}}} \tag{16}$$

$$q_{p_1} = \frac{\nu_{p_1} s}{K'_s + s + s^2/K'_{s_I}} \frac{K'_{p_1}}{K'_{p_1} + p_1 + p_1^2/K'_{p_{1I}}} \tag{17}$$

$$q_{p_2} = \frac{\nu_{p_2} s}{K''_s + s + s^2/K''_{s_I}} \frac{K''_{p_2}}{K''_{p_2} + p_2 + p_2^2/K''_{p_{2I}}} \tag{18}$$

The kinetic parameters in Eq.(16)-(18) have been determined by Jang and Wang (1998) and shown in Table 2.

Some physical constraints should be considered in the optimization problem. The total volume of the bioreactor is bounded by

$$g_1 = V(t) - V_f \le 0 \tag{19}$$

In addition, the stoichiometry of the ethanol and glycerol formation from glucose must be obeyed in the fermentation process, so that two additional constraints for each specific yield factor are restricted by

$$g_2 = \frac{p_1(t)V(t) - p_1(0)V(0)}{(V(t) - V(0)|s_F + V(0)s(0) - V(t)s(t)} - \hat{Y}_{p_1/s} \le 0 \tag{20}$$

$$g_3 = \frac{p_2(t)V(t) - p_2(0)V(0)}{(V(t) - V(0)|s_F + V(0)s(0) - V(t)s(t)} - \hat{Y}_{p_2/s} \le 0 \tag{21}$$

From stoichiometric point of view, both theoretical yield factors, $\hat{Y}_{p_1/s}$ and $\hat{Y}_{p_2/s}$, for the ethanol formation from glucose are 0.51. If the constraints (20) and (21) are not included in the optimization problem, the unrealistic predicted values of $p_1\{t_f\}$ may be found. In order to reduce the separation cost, the residue sugar is restricted by

$$g_4 = s(t) - s_r \le 0 \tag{22}$$

where s_r is the desired residue sugar.

The goal of this optimization problem is to determine the optimal feed rate $F(t)$ and fermentation time t_f to maximize the ethanol production rate. The objective function is therefore expressed as

$$\max_{F(t), t_f} J = \{p(t_f)V(t_f) - p(0)V(0)\}/t_f \tag{23}$$

where the decision variables are bounded in the solution space as follows:

$$0 \le F(t) \le F_{\max} \tag{24}$$

$$t_{f\min} \le t_f \le t_{f\max} \tag{25}$$

The feed rate $F(t)$ in this dynamic optimization problem is in terms of time so that such a problem is an infinite dimensional problem. To solve this problem efficiently, the feed rate is first approximated by a finite set of control actions $F(j)$ in the time interval $t_{j-1} \le t < t_j$ by

$$F(t) = F(j), \quad j = 1, \ldots, N_u \tag{26}$$

where N_u is the number of time partitions. In order to avoid a wide variation of feed control action, each action is bounded by

$$g_{4+l} = \left|F(l+1) - F(l)\right| - 0.25\,F_{\max} \le 0, \qquad l = 1,\ldots,N_u - 1 \tag{27}$$

The infinite dimensional problem (11)-(25) is then approximated as a parameter selection problem. In order to solve the constrained optimization problem, the penalty function method transforms the constraints into the unconstrained problem as

$$J_a(\mathbf{u}) = J + \sum_{k=1}^{4} \alpha_k \left\{ \int_0^{t_f} \left\langle g_k(\mathbf{z}(t), \mathbf{u}) \right\rangle_+^2 dt + \sum_{k=1}^{N_u - 1} \alpha_{4+k} \left\langle g_{4+k} \right\rangle_+^2 \right. \tag{28}$$

where the decision vector $\mathbf{u}$ consists of the control actions $F(j)$ and the fermentation time t_f, and the state variables $\mathbf{z}(t)$ is expressed as $\mathbf{z} = [x, s, p_1, p_2, V]^T$. Here, the penalty parameters α_k are positive constant and the bracket operator in (28) is defined as $\langle g_k \rangle_+ = \max(g_k, 0)$. Penalty terms associated with the constraints are added to the objective function. In such a way, penalty terms reflect the violation of the constraints and assign the high cost of the penalty function to candidate points far from the feasible region. While we use HDE to solve the penalty problem, any candidate individuals that violate the constraints would inherit a worse fitness and be hard to survive.

The main limitation of the penalty functions is the degree to which each constraint is penalized. Powell (1978) has noted that the classical optimization methods including penalty functions have certain weaknesses that become most serious when penalty parameters are large. More seriously, large penalty parameters make the penalty function ill conditioned so that it is difficult to achieve a good solution. On the other hand, if the penalty parameters are too small, the constraint violation does not contribute a high cost to the penalty function. Thus, the optimal solution based on the penalty function may not be feasible. Therefore, how to choose appropriate penalty parameters is not trivial.

Lagrange methods are classical methods for solving real-valued constrained optimization problems (Luenberger, 1984). Such methods can significantly improve the drawbacks of penalty methods. Kim and Myung (1997) have developed two-phase EP with Lagrange method to solve real-valued constrained optimization problems. In this paper, we introduce an incorporated algorithm of HDE with multiplier updating to solve the dynamic optimization problem. The augmented Lagrange function for the dynamic optimization is defined as

$$J_a(\mathbf{u}) = J + \sum_{k=1}^{4} \alpha_k \left(\int_0^{t_f} \left\{ \left\langle g_k\{\mathbf{z}(t), \mathbf{u}\} + \sigma_k \right\rangle_+^2 - \sigma_k^2 \right\} dt + \sum_{k=1}^{N_u - 1} \alpha_{4+k} \left[\left\langle g_{4+k} + \sigma_k \right\rangle_+^2 - \sigma_k^2 \right] \right. \tag{29}$$

where the corresponding multipliers are updated at each generation as

$$\sigma_k^G)t) = \sigma_k^{G-1})t) + \max\left\{ g_k^{G-1})\mathbf{z}(t), \mathbf{u}), -\sigma_k^{G-1})t) \right\}, \qquad k = 1,\ldots,4$$

and

$$\sigma_{4+k}^G = \sigma_{4+k}^{G-1} + \max\left(g_{4+k}^{G-1}, -\sigma_{4+k}^{G-1} \right), \; k = 1,\ldots,N_u \tag{30}$$

where G is the generation index in the HDE algorithm.

4. Computational Results

All computations were performed on a Pentium II computer using Microsoft Windows NT. We use Compaq Visual Fortran to implement the HDE algorithm. The setting factors used for all runs are listed as follows. The mutation factor ρ_m is taken as a random number in [0,1]. The crossover factor ρ_c is set to 0.5. Two tolerances, ε_1 and ε_2, used in migration operation are set to 0.05. The population size of 5 is used in the computation to illustrate the performance of HDE. The maximum generations of 50000 are used for all runs.

The initial fed and residue concentrations of glucose are set to be $200g/l$ and $0.1g/l$ in the computation. The maximum feed rate and total working volume are considered as $1.0l/h$ and $5l$. In order to solve this problem, the feed rate $F(t)$ is approximated by 10 time partitions. At first, we use the penalty parameters of one for the HDE with multiplier updating method (referred to HDE-MUM) to solve this dynamic optimization problem. The maximum production rate of $23.164g/h$ is obtained as shown in Table 3. Such a computation uses the acceleration operation of 3322, the migrating operation of 171, and the total function call of 323623 which is equivalent to the CPU time of about 17 minutes on a Pentium II 400 computer. We perform the problem 3 runs with various penalty parameters. Table 3 shows the best solution for the three runs. The best solutions for the three runs are feasible and nearly identical although the penalty parameters used for each run are a wide difference. The HDE with the penalty function method (HDE-PFM) is also applied to solve the problem. The best solution for various penalty parameters is also shown in Table 3 for comparison.

The objective function values obtained by HDE-PFM are larger than those by HDE-MUM. However, the best solutions obtained by HDE-PFM with $\alpha_k = 1$ and 10 are infeasible. While the penalty parameter is 10^6, the penalty function method is able to obtain a feasible solution.

The original algorithm of DE with penalty function method (DE-PFM) was also applied to solve this dynamic optimization problem for comparison. In the original algorithm of DE, the mutation factor is fixed and provided by a user. We tried various pair of mutation and crossover factors for the original DE to solve the problem. However, we always obtained a premature solution using the fixed mutation factor. We then used the random mutation factor at every generation. The other setting factors are identical to those used in HDE except switching off the migrating and acceleration operations. The premature convergence is still found by using the population size of five. A satisfied solution was unable to be obtained by DE using such a small population size due to the vanishing of the mutation and crossover operations in (2) and (3). A reasonable solution can be obtained by using the population size of 20. Table 3 shows the best solution for the three runs and the used total function call. From this table, we observe that the DE-PFM with $\alpha_k = 1$ and 10 is unable to obtain a feasible solution.

We additional use a direct search method, a subroutine BCPOL in IMSL Math/Library (1991), to solve the dynamic optimization problem. Since the solution obtained by the BCPOL subroutine is strongly depended on the provided initial guess and stopping tolerance, we therefore run the problem ten times with randomly assigned initial guess and the stopping tolerance of 10^{-7}. The maximum ethanol production rate of $22.012 g/h$ is obtained by the BCPOL with penalty parameters of 10^6. The total function call of 4514 is used in this case. However, the best solution is smaller than that of HDE-MUM.

5. Conclusions

We have applied the HDE with multiplier updating method to solve the constrained dynamic optimization problem. The hybrid method includes an acceleration operation and migrating operation so that the HDE can use a population size of five to obtain a global solution. Several methods have also employed to solve the problem. From the comparison, the HDE-MUM can use smaller penalty parameters to obtain a satisfied solution.

Acknowlwdgment

Financial research support from the National Science Council of the R.O.C. under grant No. NSC89-2214-E194-002 is greatly appreciated.

References

Back, T., Fogel, D., and Michalewicz, Z., *Handbook of Evolutionary Computation.* Brostol: Institute of Physics Publishing Ltd and New York: Oxford Univ. Press, 1997.

IMSL Math/Library User's Manual (1991), IMSL, Inc., Houston.

Jang,H.J. and Wang, F.S. (1998), Modeling of simultaneous fermentation of ethanol and glycerol using *Saccharomyces diastaticus* LORRE-316, Proceedings of the Third Conf. on Biochemical Engineering, Taiwan, 195-199.

Kim, J.H. and Myung, H. (1997), Evolutionary *programming* techniques for constrained optimization problems, *IEEE Trans. Evolutionary Computation*, **1**, 129-140.

Luenberger, D.G., *Linear and Nonlinear Programming*. Addison-Wesley, 1984.

Michalewicz, Z. and Schoenauer, M. (1996), Evolutionary algorithms for constrained parameter optimization problems, Evolutionary Computation, **4**, 1-32.

Powell, M. J. D. (1978), Algorithms for nonlinear constraints that use Lagrangian functions, *Math. Programming*, **14**, 224-248.

Storn, R. and Price, K.V. (1996) Minimizing the real function of the ICEC'96 contest by differential evolution. *IEEE Conf. on Evolutionary Computation*, Nagoya, 842-844.

Storn, R. and K. Price, K. (1997), Differential evolution: a simple and efficient heuristic for global optimization over continuous spaces, *J. Global Optimization*, **11**, 341-369.

Wang, F.S. and Chiou, J.P. (1997) Optimal control and optimal time location problems of differential-algebraic systems by differential evoultion. *Industrial & Engineering Chemistry Research* **36,** 5348-5357.

Wang, F. S., Jing, C. H. and Tsao, G. T. (1998) Fuzzy decision making problems of fuel ethanol production using a genetically engineered yeast. *Industrial & Engineering Chemistry Research.* **37**, 3434-3443.

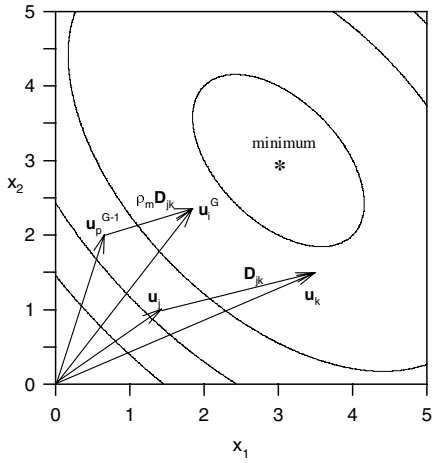

Figure 1. An example of a two-dimensional objective function showing its contour and the operation for generating mutant vectors. D_{jk} is the direction of differential variation $\mathbf{u}_k - \mathbf{u}_j$. $\rho_m D_{jk}$ is the real perturbation.

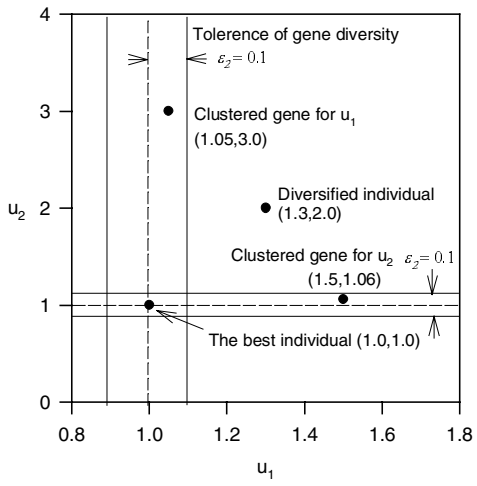

Figure 2. Illustration of the clustered gene and diversified individual with respect to the best individual.

Table 1. The basic operations of evolutionary algorithms and HDE.

Evolutionary Algorithm	**Hybrid Differential Evolution**
1. Representation and Initialization	1. Representation and Initialization
2. Mutation	2. Mutation
3. Crossover operation	3. Crossover operation
4. Selection and Evaluation	4. Selection and Evaluation
5. Repeat steps 2 to 4	5. Accelerated operation if necessary
	6. Migrating operation if necessary
	7. Repeat steps 2 to 6

Table 2. The kinetic parameters and initial conditions of the problem

$\mu_m = 0.827$, $K_s = 4.962$, $K_{s_I} = 357.14$, $\nu_{p_1} = 1.728$,

$K_s' = 10.262$, $K_{s_I}' = 1013.8$, $\nu_{p_2} = 0.214$, $K_s'' = 70.723$,

$K_{s_I}'' = 4154.3$, $Y_{p_1/s} = 0.469$, $Y_{p_2/s} = 0.272$, $K_{p_1} = 49.477$,

$K_{p_{1I}} = 8.04$, $K_{p_1}' = 398.757$, $K_{p_{1I}}' = 8.275$, $K_{p_2}'' = 10.79$,

$K_{p_{2I}}'' = 2.449$, $K_{p_2} = 413.987$, $K_{p_{2I}} = 82.108$, $x(0) = 1.6 g/l$,

$s(0) = 10.0 g/l$, $p_1(0) = 2.2 g/l$, $p_2(0) = 0.0 g/l$, $V(0) = 1.5 l$

Table 3. The maximum ethanol production rate for various methods.

Method \ Item		max J (g/h)	SIC	TFC
HDE-MUM (Np=5)	$\alpha_k = 1$	23.164	2.4905E-8	323623
	$\alpha_k = 10$	23.164	3.0441E-9	320836
	$\alpha_k = 10^6$	23.046	6.9404E-9	300046
HDE-PFM (Np=5)	$\alpha_k = 1$	25.925	3.0480	256375
	$\alpha_k = 10$	24.204	4.5355E-1	263148
	$\alpha_k = 10^6$	23.118	5.4561E-6	307043
DE-PFM (Np=20)	$\alpha_k = 1$	25.922	3.0676	1000020
	$\alpha_k = 10$	24.159	4.2298E-1	1000020
	$\alpha_k = 10^6$	23.039	7.0329E-6	1000020
BCPOL-PFM	$\alpha_k = 1$	25.925	3.0455	3797
	$\alpha_k = 10$	24.202	4.5302E-1	3721
	$\alpha_k = 10^6$	22.012	8.8971E-6	4514

SIC: Sum of the infeasible constraints defined as

$$SIC = \sum_{k=1}^{4} \alpha_k \left(\int_0^{t_f} \left\langle g_k(\mathbf{z}) t(, \mathbf{u} \right\rangle_+ dt + \sum_{k=1}^{N_u - 1} \alpha_{4+k} \left\langle g_{4+k} + \sigma_k \right\rangle_+ \right.$$

TFC: Total function call