Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

Dynamic Optimisation of Chemical Engineering Processes Using the Bees Algorithm

D.T. Pham*, Q.T. Pham, A. Ghanbarzadeh*, ***, M. Castellani***

** Manufacturing Engineering Centre, Cardiff University, CF24 3AA, U.K. (Tel: +44-(0)2920874429; e-mail: phamdt, ghanbarzadeha1, castellanim, @cardiff.ac.uk).*
*** University of New South Wales, Sydney 2052, Australia (e-mail: tuan.pham@unsw.edu.au)*
**** Mechanical Engineering Department, Engineering Faculty, Shahid Chamran University, Ahvaz, Iran*

Abstract: An improved version of the Bees Algorithm is proposed for solving dynamic optimisation problems. This new formulation of the Bees Algorithm includes new search operators, and a new selection procedure that enhances the survival probability of newly formed individuals. The proposed algorithm is tested on six benchmark dynamic optimisation problems. The benchmark problems include minimisation and maximisation tasks of different levels of complexity. For all the problems, the Bees Algorithm finds very satisfactory optima. The very small standard deviation of the results proves the high reliability of the proposed technique. Experimental tests show that the Bees Algorithm outperforms the state-of-the-art Ant Colony Optimisation algorithm. The Bees Algorithm improves also the best results published in the literature for the six benchmark problems.

1. INTRODUCTION

Many real-world engineering problems imply the dynamic optimisation of a system of differential equations. For an example, in chemical and biological engineering applications the speed of reactions is usually determined by differential equations involving various control variables (e.g., pressure, temperature, catalyst concentration, degree of steering, etc.). The engineer manipulates the control variables over time in order to optimise given quality parameters such as product quality and yield, minimise the production costs, etc.

The solution of dynamic control problems is usually complex. Several state and control variables are needed to describe the system response, and the input-output relationship is often highly nonlinear and ill-behaved. Implicit discontinuities and constraints on both state and control variables are also frequent. The desired control policy is the one that maximises a user-defined performance index. The performance index is expressed through an objective function (fitness function).

Different algorithms were proposed in the literature for the solution of dynamic optimisation problems. Classical methods are based on hill climbing of the fitness landscape (Binder et al., 2000). Unfortunately, gradient-based dynamic optimisation procedures are prone to sub-optimal convergence to local peaks of performance (Angira and Santosh, 2007). For this reason, several studies considered the application of population-based global search algorithms. Pham (1998; 2007), Angira and Santosh (2007), and Roubos et al. (1999) used different kinds of evolutionary algorithms (Fogel, 2000) to solve various dynamic control problems. Ant colony optimisation (Dorigo and Stützle, 2004) was applied by Rajesh et al. (2001) to a number of chemical engineering benchmark problems.

This paper presents the experimental results achieved by applying an improved version of the Bees Algorithm to a range of benchmark dynamic control problems. The Bees Algorithm was recently developed by Pham (2005; 2006a; 2006b). The optimisation results obtained by the Bees Algorithm are compared to the known best results in the literature, and to the results obtained using the popular ant colony optimisation algorithm. Section 2 states the problem domain. Section 3 describes the proposed algorithm. Section 4 presents the results of the experimental comparison. Section 5 concludes the paper.

2. PROBLEM DOMAIN

Dynamic control problems in engineering are defined by the following equations and constraints:

$$\underset{u(t)}{\text{Maximize}} f(u, y) \tag{1}$$

subject to

$$\frac{dy}{dt} = F(t, y(t), u(t)), \quad t \in [0, t_f] \tag{2}$$

$$y(0) = y_0 \tag{3}$$

$$u^{min} \le u(t) \le u^{max} \tag{4}$$

where t is the independent variable (usually the time), y the state variable vector (size m), $\mathbf{u}$ the control variable vector (size n), and t_f the final time. The equations are almost always integrated numerically.

The aim of the optimisation problem is to manipulate $\mathbf{u}(t)$ in order to maximise the objective function f. The interval $[0, t_f]$ is partitioned into a finite number of time steps (20–40 in this work), and each control variable u_i is specified at each of

these steps. Within each time interval, the control variables are assumed to ramp between the end points. Each variable u_i is represented by a vector of p elements, each vector defining p-1 time intervals. The n u_i vectors are juxtaposed to form an overall solution vector of length $n \times p$, which represents a solution to the dynamic optimisation problem.

3. THE BEES ALGORITHM

The Bees Algorithm is a stochastic global optimisation method that models the foraging behaviour of honey bees. The search strategy of the Bees Algorithm combines global random exploration with local neighbourhood sampling. Explorative search and exploitative search are clearly differentiated, and they are independently varied through a set of parameters. This clear decoupling between exploration and exploitation facilitates the tuning of the algorithm.

The Bees Algorithm is best suited for objective functions that are multimodal, include flat regions and points of discontinuity. Being a stochastic global optimisation procedure, it is also robust to noisy fitness evaluations. The weakness is that for optimisation of smooth unimodal functions it is much slower than deterministic methods.

This section presents the original formulation of the Bees Algorithm and the improved version which is used for this study.

3.1 Bees foraging behaviour in nature

During the harvesting season, a colony of bees keeps a percentage of its population as scouts (Von Frisch, 1976) and uses them to explore the field surrounding the hive for promising flower patches. The foraging process begins with the scout bees being sent to the field where they move randomly from one patch to another.

When they return to the hive, those scout bees that found a patch of a sufficient quality (measured as the level of some constituents, such as sugar content) deposit their nectar and go to the "dance floor" to perform a dance known as the "waggle dance" (Seeley, 1996). This dance communicates to other bees three pieces of information regarding a flower patch: the direction in which it will be found, its distance from the hive, and its quality rating (or fitness) (Von Frisch, 1976; Camazine et al., 2003). This information helps the bees watching the dance to find the flower patches without using guides or maps. After the waggle dance, the dancer (i.e. the scout bee) goes back to the flower patch with follower bees recruited from the hive. The number of follower bees will depend on the overall quality of the patch. Flower patches with large amounts of nectar or pollen that can be collected with less effort are regarded as more promising and attract more bees (Seeley, 1996; Bonabeau, 1999). In this way, the colony can gather food quickly and efficiently.

3.2 The Bees Algorithm

This section reviews the main steps of the Bees Algorithm. For more details, the reader is referred to Pham (2005; 2006a; 2006b). Figure 1 shows the pseudo code for the Bees Algorithm. The algorithm requires a number of parameters to be set, namely: number of scout bees (n), number of sites selected for local search (out of n visited sites) (m), number of top-rated (elite) sites among m selected sites (e), number of bees recruited for the best e sites (nep), number of bees recruited for the other (m-e) selected sites (nsp), the initial size of each patch (ngh) (a patch is a region in the search space that includes the visited site and its neighbourhood), and the stopping criterion. The algorithm starts with n scout bees randomly distributed in the search space. The fitness of the sites (i.e. the performance of the candidate solutions) visited by the scout bees is evaluated in step 2.

In step 4, the m sites with the highest fitnesses are designated as "selected sites" and chosen for neighbourhood search. In steps 5 and 6, the algorithm searches around the selected sites, assigning more bees to search in the vicinity of the e sites of highest fitness. Search in the neighbourhood of the best e sites – those which represent the most promising solutions - is made more detailed. As already mentioned, this is done by recruiting more bees for the best e sites than for the other selected sites. Together with scouting, this differential recruitment is a key operation of the Bees Algorithm.

1. Initialise population with random solutions.
2. Evaluate fitness of the population.
3. While (stopping criterion not met)
   //Forming new population.
4. Select sites for neighbourhood search.
5. Recruit bees for selected sites (more bees for best *e* sites) and evaluate fitnesses.
6. Select the fittest bee from each patch.
7. Assign remaining bees to search randomly and evaluate their fitnesses.
8. End While

Figure 1. Pseudo code of the basic Bees Algorithm

In step 6, for each patch only the bee of highest fitness value is selected to form the next bee population. In nature, there is no such restriction. This restriction is introduced here to reduce the number of points to be explored. In step 7, the remaining bees in the population are placed randomly around the search space to scout for new potential solutions.

At the end of each iteration, the colony has two parts to its new population: representatives from the selected patches, and scout bees assigned to conduct random searches. These steps are repeated until a stopping criterion is met.

The algorithm described above is designed for finding the optimum solution in a given search space. If a problem requires the discovery of the largest possible number of

solutions that satisfy a given quality criterion, a filtering method can be adopted to keep all non-identical satisfactory individuals generated during the evaluation process.

3.3 The Improved Bees Algorithm

The Bees Algorithm represents candidate solutions as vectors of real numbers, each vector encoding a continuous variable. To explore the solution space, the Bees Algorithm uses a random search operator which is applied within a neighbourhood window. The modified Bees algorithm complements random search with the following operators: mutation, creep, crossover, interpolate and extrapolate.

The mutation operator assigns random values, uniformly distributed within the variables' range, to randomly chosen elements of the variables vector. The creep operator applies small Gaussian changes (standard deviation equal to 0.001 of the variable range) to all elements of the solution vector. The crossover operator combines the first e elements of a parent vector with the remaining elements of a second parent vector. Interpolation is the element-by-element addition of 0.7 times the fitter parent and 0.3 times the less fit parent, giving a solution which is biased to the first. Extrapolation is the element-by-element addition of 1.3 times the fitter parent and −0.3 times the less fit parent. Once a solution is selected for the application of one of the two-parent modification operators, the second parent is chosen randomly from the rest of the population.

The probabilities of the reproduction operators are based on previous results (Pham, 2007).

From one generation to the next, the best of the population is retained (truncation selection). Each surviving solution (bee) is given an opportunity to improve by evolving a few times using the above operators. At each step, if a change results in an improvement, the new solution replaces its parent, else the original solution is retained. The number of search steps, m, is a decreasing function of the fitness rank of the solution:

$$m = \text{int}\left[(n_0 - 1)\left(\frac{m - Rank}{m - 1} \right)^f \right] + 1 \qquad (5)$$

where *Rank* is 1 for the fittest member. n_0 is the number of search steps applied to the fittest member of a generation. The least fit selected member always receives only one search step. The exponent f is an indication of the selective pressure: for $f = 1$ the number of search steps varies linearly with rank, while for $f > 1$ the number of search steps decreases more quickly with rank.

As in the original version of Bees Algorithm, the least fit members are deleted and new random bees are created to replace them. However, forcing new solutions to compete immediately with previous solutions, which have evolved for a long time, is counterproductive, just as it would be in a society where newborn were forced to compete with adults without going through a period of nurturing. With multimodal objective functions in particular, a new solution may land at the foot a high peak yet still be worse than old solutions that have stalled at a lesser peak. Therefore, this paper introduces a new category of "young members", those that have evolved for a certain number M of steps or less. Young members only compete among themselves, until they reach "adulthood" after evolving more than M steps.

In summary, the algorithm strongly favours the best bees, but also continuously create young bees and nurture them, so that the pool of "best bees" may be continuously rejuvenated if necessary.

The pseudocode of the modified Bees Algorithm is given in Figure 2.

```
Initiate N random bees
Iterate
    Sort all the bees in the present population
    Select the e fittest bees into the new population
    Sort the young bees (age ≤ M) in the present
        population
    Select the g fittest young bees into the new
        population
    Create N−e−g random new bees and put them in
        the new population
    Let the e fittest bees in the new population
        evolve m times, where m is given by eq.(1)
        (*)
    Let the other bees evolve once each (*)
Until a stopping criterion is met

(*) In the evolution steps, an offspring replaces its
parent only if it is better than the parent.
```

Figure 2. Pseudo code of the modified Bees Algorithm

The pseudocode implies that a very good young bee may be selected twice, once as one of the fittest bees, and another time as one of the fittest young bees. This feature not only makes the algorithm simpler but also encourages exceptional newcomers. The possibility of a solution to be selected twice is however small, except at the early stages of the algorithm.

4. EXPERIMENTAL TESTS

The improved version of the Bees Algorithm is tested on six benchmark problems. The level of difficulty of the problems is varying to show the ability of the Bees Algorithm to solve different problems. The first two problems are minimisation problems. In this case, the objective of the optimisation process is to maximise the inverse of the fitness function ($-f$). The six benchmark functions are listed in Table 1. and fully described by Rajesh et al. (2001).

The learning results of the Bees Algorithm are compared to the results achieved by Ant Colony Optimisation. The Ant Colony Optimisation algorithm is considered a good term for comparison because it is a popular and well-established state-

Table 1. Test objective functions

Problem	References	Reported Optimum	Ant Colony Optimisation	Bees Algorithm	Standard Deviation	Description
1	Rajesh et al. (2001) (Problem 1), Luus (1991)	0.76159	0.76238	0.761646	0.000017	Unconstrained math system
2	Rajesh et al. (2001) (Problem 2), Luus (1990)	0.12011	0.12904	0.119690	0.000066	Non-linear unconstrained mathematical system
3	Rajesh et al. (2001) (Problem 3), Dadebo and McAuley (1995)	0.57353	0.57284	0.573471	0.000024	Tubular reactor parallel reaction
4	Rajesh et al. (2001) (Problem 4), Dadebo and McAuley (1995), Ray (1981)	0.610775	0.61045	0.610775	0.000007	Batch reactor consecutive reactions, A→B→C
5	Rajesh et al. (2001) (Problem 5), Dadebo and McAuley (1995)	0.476946	0.47615	0.477444	0.000015	Plug flow reactor catalyst blend, A↔B→C
6	Rajesh et al. (2001) (Problem 6), Tieu et al. (1995)	0.480047	0.4793	0.480078	0.000013	Consecutive reactions, A→B→C

of-the-art population-based search algorithm

4.1 Experimental Settings

For each of the six dynamic control benchmark problems, the Bees Algorithm is run 25 times with different random initialisations. The learning results are estimated as the average values of the 25 independent learning trials. The duration of the Bees Algorithm is set to 263 learning cycles. This figure corresponds to 5000 function (i.e. candidate solutions) evaluations.

The optimisation of the Bees Algorithm is carried out according to experimental trial and error. Once the learning parameters are optimised, they are fixed and kept unchanged for all the optimisation problems. Table 2. gives the final setting of the learning parameters.

The optimisation results achieved by the Bees Algorithm are compared to the results obtained by Rajesh et al. (2001) using

Table 2. The Bees Algorithm parameters

Improved Bees Algorithm	
Population Size	6
Best fitness fraction	0.8
Best new fraction	0.1
Maturity age	10
Max. no. of exploitation steps n0	10
Distribution exponent f	4
Number of evaluation	5000

Ant Colony Optimisation. For each benchmark problem, the results published by Rajesh et al. (2001) are the average value of 25 independent runs. For each problem, the function optimum found by the Bees Algorithm is also compared to the best-so-far optimum reported in the literature (Rajesh et al., 2001; Pham, 2007).

4.2 Experimental Results

For each benchmark problem, Figures 3 to 8 show the evolution of the function optimisation process over the duration of a sample run. The value of the objective function (fitness function) is plotted versus the number of function evaluations (search space samples). In the the first two optimisation problems, the evolution of the fitness function is represented by a descending curve (minimisation problem). In the remaining four problems, the curve of the fitness function is ascending (maximisation problem).

In the plot referring to problem 1, after about 400 samples of the optimisation landscape the Bees Algorithm localises the minimum of the objective function. In problems 2 to 5, the Bees Algorithm needs about 1000 evaluations to find the optimum of the objective function. In all the cases, the fitness evolution curve is characterised by a steep initial improvement (descent in the case of problem 1-2, ascent in the case of problems 3-6), followed by a slower fine tuning of the solution. This behaviour is typical of population-based search algorithms. In general, the optimisation results support the claim that the Bees Algorithm converges reasonably fast to the optimum point.

Table 1. compares the results obtained by the Bees Algorithm to the results obtained using Ant Colony Optimisation, and the best optimisation results reported in the literature. Table 1. shows that the Bees Algorithm outperforms Ant Colony Optimisation in all the benchmark problems. Save for

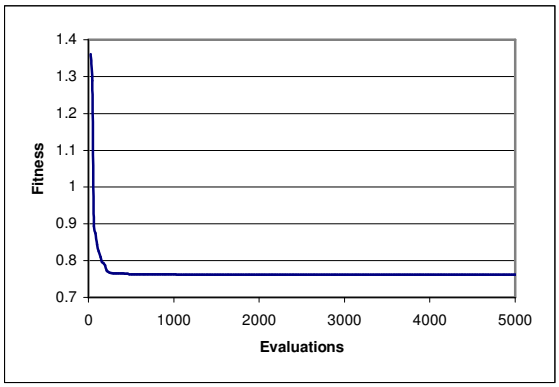

Figure 3. Evolution of fitness with number sampling for Problem 1

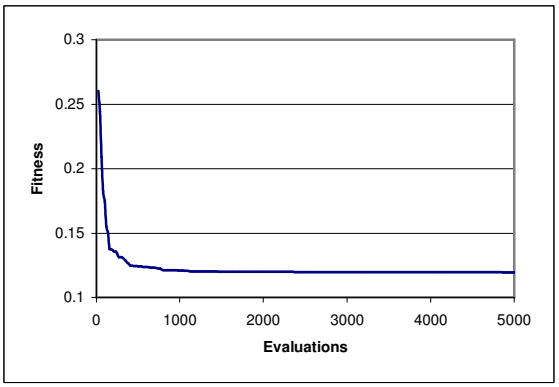

Figure 4. Evolution of fitness with number sampling for Problem 2

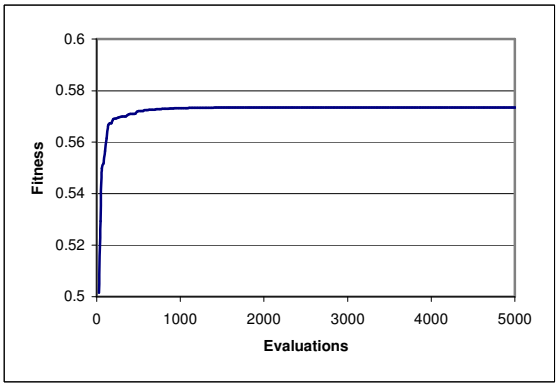

Figure 5. Evolution of fitness with number sampling for Problem 3

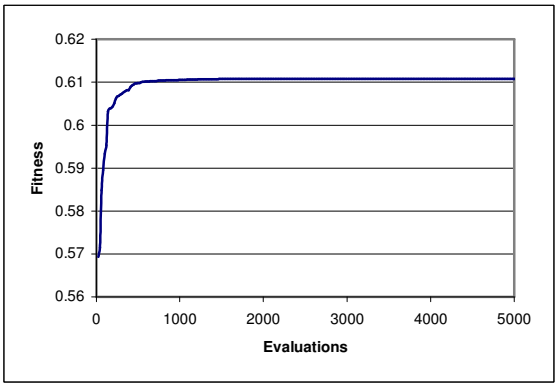

Figure 6. Evolution of fitness with number sampling for Problem 4

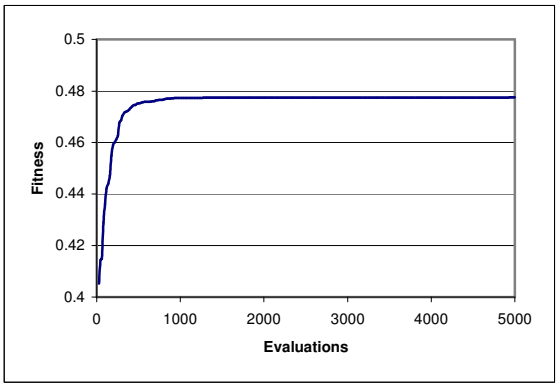

Figure 7. Evolution of fitness with number sampling for Problem 5

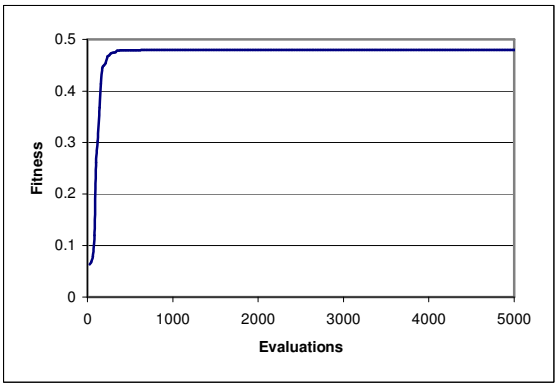

Figure 8. Evolution of fitness with number sampling for Problem 6

problem 1, the Bees Algorithm finds also better solutions than the best known solutions reported in the literature. In the case of problem 1, the Bees Algorithm approximates very closely the known analytical solution.

For each problem Table 1. reports also the standard deviation of the optimisation results over the 25 independent runs. The very small variation of the results proves the robustness of the proposed algorithm.

5. CONCLUSIONS

This paper presented the application results of a new version of the Bees Algorithm to six benchmark dynamic optimisation problems. The six problems include minimisation and maximisation tasks of different levels of complexity. For all the problems, the Bees Algorithm finds very satisfactory optima. The very low standard deviation of the optimisation results over 25 independent optimisation trials proves the robustness of the proposed algorithm. In the five cases where the optimum is not known, the Bees Algorithm finds better a solution than the best known results reported in the literature. In the remaining case, the solution found by the Bees Algorithm is very close to the known analytical optimum.

The results obtained by the Bees Algorithm are compared to the results obtained on the same benchmark problems by the Ant Colony Optimisation algorithm. In all the six cases, the Bees Algorithm outperformed Ant Colony Optimisation.

The Bees Algorithm solves the dynamic optimisation tasks without any domain information, apart from the formulation of the objective function. In this respect, the Bees Algorithm shares the advantages of global search algorithms such as Evolutionary Algorithms and Ant Colony Optimisation.

ACKNOWLEDGEMENTS

The authors are members of the EC FP6 Innovative Production Machines and Systems (I*PROMS) Network of Excellence.

REFERENCES

Angira, R. and Santosh, A. (2007). Optimization of dynamic systems: A trigonometric differential evolution approach. *Computers and Chemical Engineering* 31, 1055–1063

Binder, T., Cruse, A., Villar, C.A.C., and Marquardt, W. (2000). Dynamic optimizationusing a wavelet based adaptive control vector parameterization strategy. *Computers and Chemical Engineering*, 24, 1201–1207.

Bojkov, B. and Luus, R. (1992). Use of random admissible values for control initerative dynamic programming. IEC Research, 31, 1308–1314.

Bonabeau, E., Dorigo, M. and Theraulaz G. (1999). *Swarm Intelligence: from Natural to Artificial Systems*. New York: Oxford University Press.

Camazine, S., Deneubourg, J.L. Franks, N.R. Sneyd, J. Theraulaz, G. and Bonabeau, E. (2003). *Self-Organization in Biological Systems*. Princeton: Princeton University Press.

Dadebo, S.A. and McAuley, K.B. (1995). Dynamic optimization of constrained chemical engineering problems using dynamic programming. *Computers and Chemical Engineering*, 19, 513–525.

Fogel, D.B. (2000). *Evolutionary Computation: Toward a New Philosophy of Machine Intelligenc*e. 2nd edition, IEEE Press, New York.

Lapidus, L. and Luus, R. (1967). *Optimal Control of Engineering Processes*. Blaisdell Pub., Waltham, MA, pp.155-229.

Luus, R. (1990). Optimal Control by Dynamic Programming Using Systematic Reduction in Grid Size. *Int. J. Control*, 51, 995–1013.

Luus, R. (1991). Application of iterative dynamic programming to state constrained optimal control problems. *Hung. J. Ind. Chem.*, 19, 245–254.

Pham, D.T., Ghanbarzadeh, A. Koc, E., Otri, S., Rahim S. and Zaidi, M. (2005). *Technical Note: Bees Algorithm.* Manufacturing Engineering Centre, Cardiff University.

Pham, D.T., Ghanbarzadeh, A. Koc, E. and Otri, S. (2006a). Application of the Bees Algorithm to the Training of Radial Basis Function Networks for Control Chart Pattern Recognition. in 5th CIRP *International Seminar on Intelligent Computation in Manufacturing Engineering* (CIRP ICME '06). 711-716. Ischia, Italy.

Pham, D.T., A. Ghanbarzadeh, E. Koc, S. Otri, S. Rahim and M. Zaidi (2006b). The Bees Algorithm, A Novel Tool for Complex Optimisation Problems. in *2nd Int Virtual Conf on Intelligent Production Machines and Systems* (IPROMS 2006) 454-459.

Pham, Q.T. (1998). Dynamic optimization of chemical engineering processes by an evolutionary method. *Computers and Chemical Engineering*, 22, 1089–1097.

Pham Q.T. (2007). Using statistical analysis to tune an evolutionary algorithm for dynamic optimization with progressive step reduction. Computers and Chemical Engineering 31, 1475–1483.

Rajesh, J., Gupta, K., Kusumakar, H.S., Jayaraman, V.K. and Kulkarni, B.D. (2001). Dynamic optimization of chemical processes using ant colony framework. *Computers and Chemical Engineering*, 25, 583–595.

Ray, W.H. (1981). *Advanced process control*. New York: McGraw-Hill.

Roubos, J.A., van Straten, G. and van Boxtel, A.J.B. (1999). An evolutionary strategy for fed-batch bioreactor optimization: Concepts and performance. *Journal of Biotechnology*, 67, 173–187.

Seeley, T.D. (1996). *The Wisdom of the Hive: The Social Physiology of Honey Bee Colonies.* Cambridge, Massachusetts: Harvard University Press.

Tieu, D., Cluett, W.R. and Penlidis, A. (1995). A comparison of collocation methods for dynamic optimization problems. *Comput. Chem. Engng,* 19, 375–381.

Von Frisch, K. (1976). *Bees: Their Vision, Chemical Senses and Language*. Revised Edition ed, Ithaca, N.Y.: Cornell University Press.