Bioautomation, 2005, 3, 19 - 28 ISSN 1312 – 451X

Genetic Algorithms for a Parameter Estimation of a Fermentation Process Model: A Comparison

Olympia Roeva

Centre of Biomedical Engineering "Prof. Ivan Daskalov" - Bulgarian Academy of Sciences
105 Acad. G. Bonchev Str., Sofia 1113, Bulgaria
E-mail: olympia@clbme.bas.bg

Received: May 12, 2005 **Accepted: November 30, 2005**

Published: December 16, 2005

Abstract: *In this paper the problem of a parameter estimation using genetic algorithms is examined. A case study considering the estimation of 6 parameters of a nonlinear dynamic model of E. coli fermentation is presented as a test problem. The parameter estimation problem is stated as a nonlinear programming problem subject to nonlinear differential-algebraic constraints. This problem is known to be frequently ill-conditioned and multimodal. Thus, traditional (gradient-based) local optimization methods fail to arrive satisfied solutions. To overcome their limitations, the use of different genetic algorithms as stochastic global optimization methods is explored. These algorithms are proved to be very suitable for the optimization of highly non-linear problems with many variables. Genetic algorithms can guarantee global optimality and robustness. These facts make them advantageous in use for parameter identification of fermentation models. A comparison between simple, modified and multi-population genetic algorithms is presented. The best result is obtained using the modified genetic algorithm. The considered algorithms converged very closely to the cost value but the modified algorithm is in times faster than other two.*

Keywords: *Genetic algorithms, Parameter estimation, Fermentation processes*

Introduction

Mathematical optimization can be used as a computational engine to arrive at the best solution for a given problem in a systematic and efficient way. In the context of fermentation processes, coupling optimization with suitable simulation modules opens a whole new area of possibilities. Fermentation processes are complex, highly nonlinear, dynamic systems and their modeling and optimization is a complicated and rather time consuming task. The dynamic behavior of the considered process is described by known structure (a system of deterministic nonlinear differential equations) according to the mass balance. In order to optimize a real fermentation process, the model must be regarded as a step to reach more easily the final aim. The model must describe those aspects of the process that significantly affect the process performance. The important part of the model development is the choice of a certain optimization procedure for a parameter estimation, so with a given set of experimental data, to calibrate the model in order to reproduce the experimental results in the best possible way. This mathematical problem is a big challenge for traditional local optimization methods. As an alternative to surmount the parameter estimation difficulties, global optimization methods are used.

Global optimization methods can be roughly classified as deterministic and stochastic strategies [13]. Stochastic methods for global optimization ultimately rely on probabilistic approaches and can locate the vicinity of global solutions with good efficiency. Furthermore,

stochastic methods are usually quite simple to implement and use, and they do not require transformation of the original problem, which can be treated as a black box.

There are many different kinds of stochastic methods for global optimization, but the following groups must be highlighted: adaptive stochastic methods; clustering methods; evolutionary computation; simulated annealing and other meta-heuristics [13]. The most competitive stochastic optimization method, especially for large problems is the evolutionary computation, also known as biologically inspired methods, or population-based stochastic methods. This is a very popular class of methods based on the ideas of biological evolution, which is driven by the mechanisms of reproduction, mutation, and the principle of survival of the fittest. Similarly to biological evolution, evolutionary computing methods generate better and better solutions by iteratively creating new "generations" by means of those mechanisms in numerical form. Evolutionary computation methods are usually classified into three groups: genetic algorithms, evolutionary programming and evolution strategies.

In this study a set of selected genetic algorithms, namely simple genetic algorithm, modified genetic algorithm and multi-population genetic algorithm, is considered [8, 9, 12, 16, 20]. The selection has been made based on their results for a set of estimation parameters problems of fermentation processes [15, 17, 18, 19]. Three different genetic algorithms are examined in solving the associated parameter estimation problem of *E. coli* fermentation process. The main objective is to find the more efficient and reliable algorithm for the considered class of problems.

Statement of the parameter estimation problem

The parameter estimation problem of fermentation processes models is stated as minimization of a cost function. The cost function measures the goodness of the model fit with respect to a given experimental data set, subject to the dynamics of the system (acting as a set of differential equality constraints) plus possibly other algebraic constraints. Generally, the formulation is:

Find p to minimize

$$J = \int_0^{t_f} \left(y_{exp}(t) - y_{mod}(p,t) \right)^T W(t) \left(y_{exp}(t) - y_{mod}(p,t) \right) dt \rightarrow min \quad (1)$$

subject to

$$f\left(\frac{dX}{dt}, x, y, p, v, t \right) = 0 \quad (2)$$

$$x(t_0) = x_0 \quad (3)$$

$$h(x, y, p, v) = 0 \quad (4)$$

$$g(x, y, p, v) \leq 0 \quad (5)$$

$$p^L \leq p \leq p^U \quad (6)$$

where J is the cost function to be minimized, p is the vector of decision variables of the optimization problem (the set of parameters to be estimated), y_{exp} is the experimental measure of a subset of the output state variables, $y_{mod}(p, t)$ is the model prediction for those outputs, $W(t)$ is a weighting (or scaling) matrix, x is the differential state variables, v is a vector of other (usually time-invariant) parameters that are not estimated, f is the set of differential and algebraic equality constraints described the system dynamics (i.e. the nonlinear process model), and h and g are the possible equality and inequality path and point constraints that

express additional requirements for the system performance. Finally, p is subject to upper and lower bounds acting as inequality constraints.

The general formulation above is that of a nonlinear programming problem with differential-algebraic constraints. Due to the nonlinear and constrained nature of the system dynamics, parameter estimation problems very often are multimodal (nonconvex). Therefore, if these problems are solved via standard local methods, such as the standard Levenberg-Marquardt method or Nielder-Mead Simplex search method, it is very likely that the solution found will be of local nature.

Genetic Algorithms

Today the most common direct methods used for global optimization are evolutionary algorithms such as genetic algorithms (GA). Genetic algorithms are directed random search techniques, based on the mechanics of natural selection and natural genetics, which can find the global optimal solution in complex multidimensional search spaces. Recently, GA have been used extensively in solving many optimization-searching problems including mathematical function optimization, very large scale integration chip layout, molecular docking, parameter fitting, scheduling, manufacturing, clustering, machine learning, etc. [2, 3, 4, 6, 7, 10, 14, 15, 17, 18]. Compared with conventional optimization methods, GA simultaneously evaluates many points in the parameter space. It is more likely to converge towards the global solution. A genetic algorithm does not assume that the space is differentiable or continuous and can also iterate many times on each data received. A GA requires only information concerning the quality of the solution produced by each parameter set (objective function value information). This characteristic differs from optimization methods that require derivative information or, worse yet, complete knowledge of the problem structure and parameters. Since GA do not demand such problem-specific information, they are more flexible than most search methods. Also GA do not require linearity in the parameters which is needed in iterative searching optimization techniques. Genetic algorithms can solve hard problems, are noise tolerant, easy to interface to existing simulation models, and easy to hybridize. Therefore, theses properties make genetic algorithms suitable and more workable in use for a parameter estimation of fermentation models.

A brief survey (presentation) of the examined here simple, modified and multi-population genetic algorithms, is presented below.

Simple Genetic Algorithm

Simple genetic algorithms (SGA) are guided largely by the mechanisms of three operators: reproduction, crossover and mutation [8, 9, 12]. To derive a solution to a problem, the SGA initializes a single population of n randomly encoded chromosomes (individuals). The objective functions (cost values) of generated population are then evaluated. In the next step individuals represented by their associated costs are ranked and the corresponding individual fitness is received. According to their fitness the best chromosomes from a population are selected (better fitness, bigger chance to be selected). Thus solutions from one population are taken and used to form a new population. This is motivated by a hope, that the new population will be better than the old one. A certain function performs the selection concordant with the generation gap. Selected individuals are then recombined. To form a new offspring (children) the parents have to be cross overed with a crossover probability. If no crossover is performed, the offspring is the exact copy of parents. A mutation is then applied

with determinate probability. Crossover and mutation operators are realized to yield improved offspring for successive generations. For the new individuals the objective function and fitness function values are calculated. The new offspring is inserted in the population. The new generated population is used for a further run of the algorithm. Natural evolution of the population continues until a predetermined number of generations is reached.

The *Matlab* code for the generational loop of SGA is listed in Fig. 1.

```
% Initialise population
Chrom = crtbp(NIND, NVAR*PRECI);
% Counter
gen = 0;
% Evaluate initial population
ObjV = objfun1(bs2rv(Chrom, FieldD));
% Generational loop
while gen < MAXGEN,
        % Assign fitness values to entire population
        FitnV = ranking(ObjV);
        % Select individuals for breeding
        SelCh = select('sus', Chrom, FitnV, GGAP);
        % Recombine individuals (crossover)
        SelCh = recombin('xovsp', SelCh, 0.7);
        % Apply mutation
        SelCh = mut(SelCh);
        % Evaluate offspring, call objective function
        ObjVSel = objfun1(bs2rv(SelCh, FieldD));
        % Reinsert offspring into population
        [Chrom ObjV] = reins(Chrom, SelCh ,1 ,1 ,ObjV , ObjVSel);
        % Increment counter
        gen = gen+1;
end
```

Fig. 1 A simple genetic algorithm

Modified Genetic Algorithm

As it was mentioned, the reproduction in SGA is considered for determining which chromosomes will be chosen as a basis of the next generation. Generating population from only two parents may cause loss of the best chromosome from the last population. Reached good solution may be destroyed by either the crossover or the mutation, or both operations. Thereby, the best solution in SGA popped up from the new population may be inferior to the old generations. The aim of the proposed modified genetic algorithm (MGA) [16, 20] is to prevent this demerit. MGA possesses a structure similar to SGA. However, MGA distinguishes itself from SGA in that the reproduction is processed after both the crossover and mutation have been performed. Thus the deterioration problem never happens since the best solution from the current generation will be superior to or at least the same with the past.

In the beginning the modified genetic algorithm creates an initial population of n chromosomes. In the next step the algorithm evaluates the objective values (cost values) of the individuals in the current population. After that individuals are reproduced. During the reproduction, recombination (or crossover) first occurs. Genes from parents combine to form a whole new chromosome. The newly created offspring then mutates. Then MGA ranked

individuals represented by their associated costs, to be “minimized”, and returns the corresponding individual fitnesses. Next the most fitted individuals from offspring are selected. Then the objective values of the individuals in the offspring are evaluated and re-insertion of offspring in population replacing parents is done. MGA is terminated when some criteria are satisfied, e.g. a certain number of generations, a mean deviation in the population, or when a particular point in the search space is encountered.

The *Matlab* code for the generational loop of MGA is listed in Fig. 2.

```
% Generational loop
while gen < MAXGEN,
        % Recombine individuals (crossover)
        NewCh = recombin('xovsp', Chrom, 0.7);
        % Apply mutation
        NewCh = mut(NewCh);
        % Assign fitness values to entire population
        FitnV = ranking(NewCh);
        % Select individuals for breeding
        SelCh = select('sus', NewCh, FitnV, GGAP);
        % Evaluate offspring, call objective function
        ObjVSel = objfun1(bs2rv(SelCh, FieldD));
        % Reinsert offspring into population
        [Chrom ObjV] = reins(Chrom, SelCh ,1 ,1 ,ObjV , ObjVSel);
        % Increment counter
        gen = gen+1;
end
```

Fig. 2 Generational loop of a modified genetic algorithm

Multi-population Genetic Algorithm

To receive better results a single population genetic algorithm can be improved by introducing many populations, called subpopulations [4, 5, 8]. These algorithms are known as multi-population genetic algorithm (MpGA).These subpopulations evolve independently from each other for a certain number of generations (isolation time), like the single population genetic algorithm. After the isolation time a number of individuals is distributed between the subpopulations (migration). The migration rate, the selection method of the individuals for migration and the scheme of migration determines how much genetic diversity can occur in the subpopulations and the exchange of information between subpopulations. The selection of the individuals for migration can be uniform at random (pick individuals for migration in a random manner) and fitness-based (select the best individuals for migration). There are many possibilities for the structure of the migration of individuals between subpopulations. The most general migration strategy is that of unrestricted migration (complete net topology). Here, individuals may migrate from any subpopulation to another. For each subpopulation, a pool of potential immigrants is constructed from the other subpopulations. The individual migrants are then uniformly at random determined from this pool.

The multi-population genetic algorithm models the evolution of a species in a way more similar to nature than the single population genetic algorithm. The MATLAB code for the generational loop of MpGA is listed in Fig. 3.

```
% Generational loop
while gen < MAXGEN,
        % Fitness assignment to whole population
        FitnV = ranking(ObjV, 2, SUBPOP);
        % Select individuals from population
        SelCh = select(SEL_F, Chrom, FitnV, GGAP, SUBPOP);
        % Recombine selected individuals
        SelCh=recombin(XOV_F, SelCh, XOVR, SUBPOP);
        % Mutate offspring
        SelCh = mutate(MUT_F, SelCh, FieldD, [MUTR], SUBPOP);
        % Calculate objective function for offsprings
        ObjVOff = feval(OBJ_F, SelCh);
        % Insert best offspring replacing worst parents
        [Chrom, ObjV] = reins(Chrom, SelCh, SUBPOP, [1 INSR], ObjV, ObjVOff);
        % Increment counter
        gen=gen+1;
        % Migrate individuals between subpopulations
        if (rem(gen, MIGGEN) == 0)
                [Chrom, ObjV] = migrate(Chrom, SUBPOP, [MIGR, 1, 1], ObjV);
        end
end
```

Fig. 3 Generational loop of a multi-population genetic algorithm

Parameter estimation of *Escherichia coli* Fed-batch fermentation model

The mathematical formulation of the nonlinear dynamic model of *E. coli* fermentation is commonly described according to the mass balance as follows:

$$\frac{dX}{dt} = \mu_{max} \frac{S}{k_S + S} X - \frac{F}{V} X \tag{7}$$

$$\frac{dS}{dt} = -\frac{1}{Y_{SX}} \mu_{max} \frac{S}{k_S + S} X + \frac{F}{V}(S_{in} - S) \tag{8}$$

$$\frac{dA}{dt} = \frac{1}{Y_{AX}} \mu_{max} \frac{S}{k_S + S} X - \frac{F}{V} A \tag{9}$$

$$\frac{dO_2}{dt} = -\frac{1}{Y_{O_2X}} \mu_{max} \frac{S}{k_S + S} X + k_L a(O_2^* - O_2) - \frac{F}{V} O_2 \tag{10}$$

$$\frac{dV}{dt} = F \quad , \tag{11}$$

where: X is the concentration of biomass, [g/l]; S – concentration of substrate (glucose), [g/l]; A – concentration of acetate, [g/l]; O_2 – concentration of dissolved oxygen, [%]; O^*_2 – saturation concentration of dissolved oxygen, [%]; F – feeding rate, [l/h]; V – bioreactor volume, [l]; S_{in} – initial concentration of substrate in the feeding solution, [g/l]; μ_{max} – maximum values of the specific growth rates, [h^{-1}]; k_S – saturation constant, [g/l]; Y_{SX}, Y_{AX}, Y_{O_2X} – yield coefficients, [gg^{-1}]; $k_L a$ – volumetric oxygen transfer coefficient, [h^{-1}].

The optimization problem consists of the estimation of all 6 kinetic and yield parameters of the nonlinear dynamic model, formed by 5 ordinary differential equations (7)–(11), described the alteration of the process variables in time.

The experiment is carried out in the *Institute of Technical Chemistry, University of Hannover*. Experimental data set of *E. coli MC4110* fermentation is used [1]. On the basis on feeding rate data and off-line measurements of biomass, substrate (glucose), acetate and dissolved oxygen, a parameter estimation of the considered mathematical model is carried out.

The application of the simple genetic algorithm, as well as modified and multi-population genetic algorithms for considered problem is fulfilled in *Matlab* 5.3 environment. *Genetic Algorithm Toolbox* [4, 5, 11] is used with some changes and improvements of the algorithms.

The global optimization problem is stated as the minimization of a weighted distance measure J between experimental and model predicted values of the state variables, represented by the vector y:

$$J = \sum_{i=1}^{n} \sum_{j=1}^{m} w_{ij} \left\{ \left[y_{exp}(i) - y_{mod}(i) \right]_j \right\}^2 \rightarrow min \qquad (12)$$

where n is the number of measurements for each variable, m is the number of variables, y_{exp} represents the known experimental data, and y_{mod} is the vector of states that corresponds to the predicted theoretical evolution using the model with a given set of 6 model parameters. Furthermore, w_{ij} corresponds to the different weights taken to normalize the contributions of each term:

$$w_{ij} = \left\{ \frac{1}{max\left[y_{exp}(i) \right]_j} \right\}^2 \qquad (13)$$

Some adjustments of the genetic algorithm parameters, according to the regarded problem, are done to improve the algorithms. Inappropriate choice of operators and parameters in the evolutionary process makes the GA susceptible to premature convergence. Primary choice of the genetic operators and parameters depends on the problem, as well as on the chosen encoding. There is no common theory about tuning the genetic algorithm parameters.

Together with the conventional GA parameters, such as generation gap (GGAP), crossover (XOVR) and mutation (MUTR) rate, precision of binary representation (PRECI), number of individuals (NIND) and generations (MAXGEN), set of other parameters associated with MpGA are defined. Here, insertion rate (INSR) specifies how many of the individuals produced at each generation are reinserted into the subpopulation. SUBPOP specifies the number of subpopulations. Each subpopulation contains a certain number of individuals. Migration rate (MIGR) defines the number of exchanged individuals.

Fitness-based reinsertion with insertion rate equal to 0.9 is used. This means that for each subpopulation the least-fit 10% of the offspring are not reinserted. Individuals in MpGA migrate between populations at some interval. Here, at every MIGGEN = 20 generations, migration takes place between subpopulations. The most fit 20% (MIGR = 0.2) of each subpopulation (SUBPOP = 5) is selected for a migration. Nearest neighbour subpopulations then exchange these number of individuals amongst their subpopulations, uniformly reinserting the immigrant individuals. The values of genetic algorithm parameters are

summarized in Table 1. The chosen types of encoding, selection (SEL_F), crossover (XOV_F), mutation (MUT_F) and fitness (FIT_F) functions are listed in Table 2.

Table 1. Genetic parameters

Parameter	Value
GGAP	0.97
XOVR	0.70
MUTR	0.05
PRECI	20
NIND	100
MAXGEN	100
MIGR	0.20
INSR	0.20
SUBPOP	5
MIGGEN	20

Table 2. Genetic operators

Operator	Type
encoding	binary
reinsertion	fitness-based
selection	roulette wheel selection
crossover	crossover double point
mutation	bit inversion
fitness	linear ranking

The proposed genetic algorithm operators and parameters are accepted on the basis of lot of experiments to improve the optimization capability and the decision speed.

All the computations are performed using a PC/Pentium IV (3.00 GHz) platform running Windows XP. The obtained results are presented in Table 3. For all considered genetic algorithms the cost values are approximately equal, as well as the model parameter values. In view of this fact main difference between regarded genetic algorithms is the total computation time. The best value of cost value (J = 7.305) is obtained using MpGA algorithm after a total computation time of 943.172 s. Almost equally good result (J = 7.327) is reached using the MGA algorithm in a less than two minute (79.843 s). From a practical point of view this is better result. As it can be seen in Table 3, MGA is the fastest algorithm.

Table 3. Results from the model parameter estimation

Parameter	SGA	MGA	MpGA
J	7.369	7.327	7.305
CPU time (s)	282.063	79.843	943.172
μ_{max}, [h^{-1}]	0.477	0.479	0.473
k_S, [g/l]	0.015	0.018	0.013
Y_{SX}, [gg^{-1}]	0.505	0.499	0.504
Y_{AX}, [gg^{-1}]	76.747	76.419	71.728
Y_{O_2X}, [gg^{-1}]	49.053	44.616	47.384
k_La, [h^{-1}]	169.267	178.345	158.252

Fig. 4 presents simulation results for the considered best decision vector, found with MGA. As can be seen, the correlation between the experimental and predicted model concentrations of the process variables is very good.

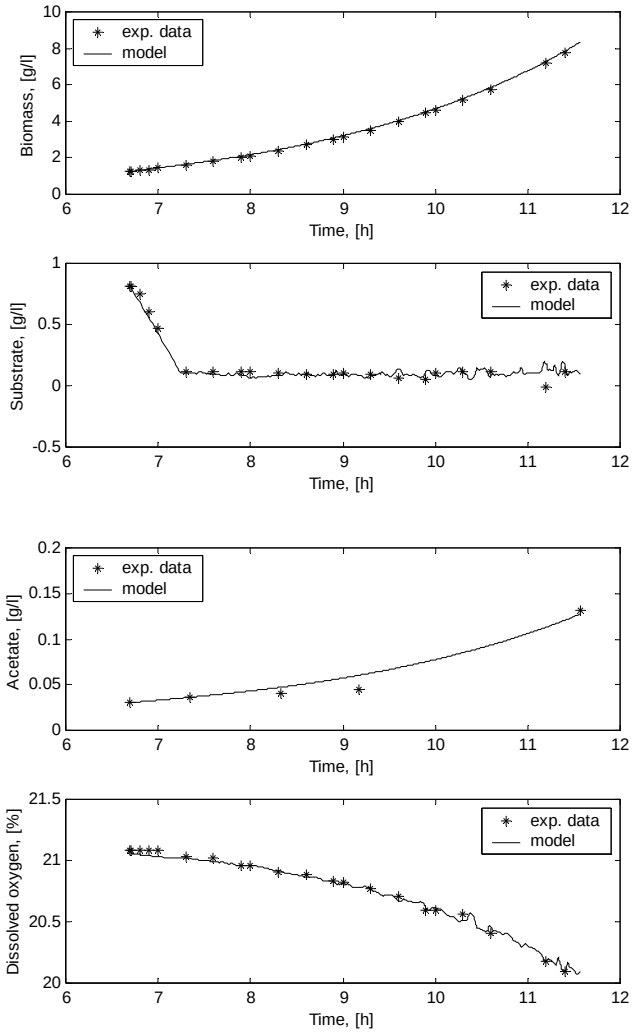

Fig. 4 Experimental and model data for biomass, glucose, acetate and dissolved oxygen

Conclusion

In this work the parameter estimation problem of *E. coli* fermentation process using genetic algorithms is performed. A comparison of three genetic algorithms, namely simple, modified and multi-population genetic algorithms is examined. The algorithm functions and parameter adjustments that significantly improve the optimization capability and the decision speed are proposed. The results show that the genetic algorithms can solve the regarded problem very well. As a result three very similar solutions of the model parameter values are obtained. Deviations of the cost values are inessential – around 0.4%. The juxtaposition of the overall computation times shows that MGA reached the solution in times faster than the other genetic algorithms, i. e. MGA solved the problem 11.8 times faster than MpGA and 3.5 times than SGA. So, although the minimum value of the cost function is achieved using the MpGA (J = 7.305), as the most appropriate algorithm for parameter estimation of fermentation processes the MGA (J = 7.327) is recommended.

Acknowledgements

This work is partially supported from National Science Fund Project № MI – 1505/2005

References

1. Arndt M., B. Hitzmann (2001). Feed forward/feedback control of glucose concentration during cultivation of *Escherichia coli*, Proceedings of the 8th IFAC Int. Conf. on Comp. Appl. in Biotechn., Quebec City, Canada, 425-429.
2. Carrillo-Ureta G. E., P. D. Roberts, V. M. Becerra (2001). Genetic Algorithms for Optimal Control of Beer Fermentation, Proceedings of the 2001 IEEE International Symposium on Intelligent Control, Mexico City, Mexico, 391-396.
3. Charbonneau P. (2002). An Introduction to Genetic Algorithms for Numerical Optimization, NCAR Technical Note, Boulder, Colorado.
4. Chipperfield A. J., P. J. Fleming (1995) The Matlab Genetic Algorithm Toolbox.
5. Chipperfield A. J., P. J. Fleming, H. Pohlheim, C. M. Fonseca (1994). Genetic Algorithm Toolbox for Use with MATLAB. User's Guide. Version 1.2. Dept. of Automatic Control and System Engineering, University of Sheffield, U. K.
6. Cordon O., F. Herrera (2001). Hybridizing Genetic Algorithms with Sharing Scheme and Evolution Strategies for Designing Approximate Fuzzy Rule-based Systems, Fuzzy Sets and Systems, 118, 235-255.
7. Fleming P. J., R. C. Purshouse (2001). Genetic Algorithm in Control Systems Engineering, Department of Automatic Control and Systems Engineering, University of Sheffield, Research Report No. 789.
8. Goldberg D. (1989). Genetic algorithms in search, optimization and machine learning, Addison-Weslcy Publishing Company, Massachusetts.
9. Holland J. (1975). Adaptation in natural and artificial systems, MIT Press.
10. Lu J., S.-C. Fang (2001). Solving Nonlinear Optimization problems with fuzzy relation Equation Constraints, Fuzzy Sets and Systems, 119, 1-20.
11. MatWorks Inc. (1999). Genetic Algorithms Toolbox, User's Guide.
12. Michalewicz Z. (1994). Genetic Algorithms + Data Structures = Evolution Programs, Second, Exended Edition, Springer-Verlag, Berlin, Heidelberg.
13. Moles C., G., P. Mendes, J. R. Banda (2003). Parameter Estimation in Biochemical Pathways: A Comparison of Global Optimization Methods, Genome Research, 13, 2467-2474.
14. Na J.-G., Y. K. Chang, B. H. Chung, H. C. Lim (2002). Adaptive Optimization of Fed-batch Culture of Yeast by Using Genetic Algorithms, Bioprocess and Biosystems Engineering, 24, 299-308.
15. Roeva O. (2003). Application of Genetic Algorithms in Fermentation Process Identification, Journal of the Bulgarian Academy of Sciences, CXVI, 3, 39-43.
16. Roeva O., (2005). A Modified Genetic Algorithm for Parameter Identification of Fermentation Processes, Biotechnology and Biotechnological Equipment, (in press).
17. Roeva O., St. Tzonkov (2003). Parameter Identification of Fermentation Processes Using Multi-population Genetic Algorithms, Technical Ideas, XL, 3-4, 18-26.
18. Roeva O., T. Pencheva, B. Hitzmann, St. Tzonkov (2004). A Genetic Algorithms Based Approach for Identification of *Escherichia Coli* Fed-batch Fermentation, Bioautomation, 1, 30-41.
19. Roeva O., T. Pencheva, Y. Georgieva, B. Hitzmann, S. Tzonkov (2004). Implementation of Functional State Approach for Modelling of *Escherichia coli* Fed-batch Cultivation, Biotechnology and Biotechnological Equipment, 18 (3), 207-214.
20. Roeva O. (2005). A Modification of Simple Genetic Algorithm, International Symposium "Bioprocess Systems'2005-BioPS'05", Sofia, Bulgaria, October 25-26, I.1-I.14.