Population Size Influence on the Genetic and Ant Algorithms Performance in Case of Cultivation Process Modeling

[1]Olympia Roeva, [2]Stefka Fidanova, [3]Marcin Paprzycki

[1]Institute of Biophysics and Biomedical Engineering, Bulgarian Academy of Science, Acad. G. Bonchev Str., bl. 105, 1113 Sofia, Bulgaria
olympia@biomed.bas.bg
[2]Institute of Information and Communication Technologies, Bulgarian Academy of Sciences, Acad. G. Bonchev str. bl25A, 1113 Sofia, Bulgaria
stefka@parallel.bas.bg
[3]Systems Research Institute, Polish Academy of Sciences, Warsaw and Management Academy, Warsaw, Poland
marcin.paprzycki@ibspan.waw.pl

Abstract. In this paper, an investigation of the influence of the population size on the Genetic Algorithm (GA) and Ant Colony Optimization (ACO) performance for a model parameter identification problem, is considered. The mathematical model of an *E. coli* fed-batch cultivation process is studied. The three model parameters – maximum specific growth rate (μ_{max}), saturation constant (k_S) and yield coefficient ($Y_{S/X}$) are estimated using different population sizes. Population sizes between 5 and 200 chromosomes and 5 and 100 ants in the population are tested with constant number of generations. In order to obtain meaningful information about the influence of the population size a considerable number of independent runs of the GA are performed. The observed results show that the optimal population size is 100 chromosomes for GA and 70 ants for ACO for 200 generations. In this case accurate model parameters values are obtained in reasonable computational time. Further increase of the population size, above 100 chromosomes for GA and 70 ants for ACO, does not improve the solution accuracy. Moreover, the computational time is increased significantly.

1 Introduction

Metaheuristics, such as genetic algorithms and ant colony optimization, are widely used to solve various optimization problems [8, 13]. They are highly relevant for industrial applications, because they are capable of handling problems with non-linear constraints, multiple objectives, and dynamic components – properties that frequently appear in the real-world problems [16]. Since their introduction and subsequent popularization [17], the GA and ACO have been frequently used as an alternative optimization tool to the conventional methods and have been successfully applied in a variety of areas, and still find increasing acceptance, for example:

– modelling and control of cultivation processes [7, 27, 28];

– model identification [1, 3, 11, 23], etc.

The metaheuristic algorithms require setting of the values of several algorithm components and parameters. These parameters values have great impact on performance and efficacy of the algorithm [14, 15, 22, 29]. Therefore, it is important to investigate the algorithm parameters influence on the performance of the developed metaheuristic algorithms. The aim is to find the optimal parameters values for the considered optimization problem. The optimal values for the parameters depend mainly on i) the problem; ii) the instance of the problem to deal with and iii) the computational time that will be spend in solving the problem. Usually in the algorithm parameters tuning a compromise between solution quality and search time should be done.

For the parameter setting of metaheuristics, several automated approaches exist. These methods use i) a single step of parameter tuning (prior to the practical use of the algorithm), or parameter control (self adaptation to the problem being optimized) [**?**]. Parameter control is well suited when one wants good average performances across diverse problems, but the needed computation overhead leads to less efficiency on specific problems, compared to parameter tuning [9]. Best known parameter tuning techniques are racing [**?**], sequential parameter optimization [5] and meta-parameter setting (sometimes referred as meta-algorithm [5]).

Population sizing has been one of the important topics to consider in evolutionary computation [2, 12, 30]. Various results about the appropriate population size can be found in the literature [25, 26]. Researchers usually argue that a "small" population size could guide the algorithm to poor solutions [18, 24, 30] and that a "large" population size could make the algorithm expend more computation time in finding a solution [18, 20, 21]. Due to significant influence of population size to the solution quality and search time [26] a more thorough research should be done for this GA parameter.

The main goal of this research is to carry out investigation of the influence of one of the key GA parameters – population size (number of chromosomes) – on the algorithm performance for identification of a cultivation process model. Parameter identification of non-linear cultivation process models is a hard combinatorial optimization problem for which exact algorithms or traditional numerical methods do not work efficiently. A non-linear mathematical model of fed-batch cultivation process of the most important host organism for recombinant protein production - bacteria *Escherichia coli* – is considered [26].

The paper is organized as follows. The problem formulation is given in Section 2. The GA and ACO algorithms are proposed in sections 3 and 4 respectively. The numerical results and a discussion are presented in Section 5. Conclusion remarks are done in Section 6.

2 Problem Formulation

2.1 *E. coli* Fed-batch Cultivation Model

Application of the general state space dynamical model [6] for the *E. coli* cultivation fed-batch process leads to the following nonlinear differential equation system [26]:

$$\frac{dX}{dt} = \mu_{max}\frac{S}{k_S + S}X - \frac{F_{in}}{V}X \tag{1}$$

$$\frac{dS}{dt} = -\frac{1}{Y_{S/X}}\mu_{max}\frac{S}{k_S + S}X + \frac{F_{in}}{V}(S_{in} - S) \tag{2}$$

$$\frac{dV}{dt} = F_{in} \tag{3}$$

where X is the biomass concentration, [g/l]; S is the substrate concentration, [g/l]; F_{in} is the feeding rate, [l/h]; V is the bioreactor volume, [l]; S_{in} is the substrate concentration in the feeding solution, [g/l]; μ_{max} is the maximum value of the specific growth rate, $[h^{-1}]$; k_S is the saturation constant, [g/l]; $Y_{S/X}$ is the yield coefficient, [-].

The initial process conditions are [4]:

- $t_0 = 6.68$ h,
- $X(t_0) = 1.25$ g/l and $S(t_0) = 0.8$ g/l,
- $S_{in} = 100$ g/l.

For the considered non-linear mathematical model of *E. coli* fed-batch cultivation process the parameters that should be identified are:

- maximum specific growth rate (μ_{max}),
- saturation constant (k_S),
- yield coefficient ($Y_{S/X}$).

2.2 Optimization Criterion

In practical view, modelling studies are performed to identify simple and easy-to-use models that are suitable to support the engineering tasks of process optimization and, especially of control. The most appropriate model must satisfy the following conditions:

(i) the model structure should be able to represent the measured data in a proper manner;
(ii) the model structure should be as simple as possible compatible with the first requirement.

The optimization criterion is a certain factor, whose value defines the quality of an estimated set of parameters. To evaluate the mishmash between experimental and model predicted data the Least Square Regression is used.

The objective consists of adjusting the parameters (μ_{max}, k_S and $Y_{S/X}$) of the non-linear mathematical model function (Eq. (1) - Eq. (3)) to best fit a data set. A simple data set consists of n points (data pairs) (x_i, y_i), $i = 1, 2, \ldots, n$, where x_i is an independent variable and y_i is a dependent variable whose value is found by observation. The model function has the form $f(x, \beta)$, where the m adjustable parameters are held in the vector β, $\beta = [\mu_{max}\ k_S\ Y_{S/X}]$. The goal is to find the parameter values for the model which "best" fits the data. The least squares method finds its optimum when the sum J of squared residuals:

$$J = \sum_{i=1}^{n} r_i^2$$

is a minimum. A residual is defined as the difference between the actual value of the dependent variable and the value predicted by the model. A data point may consist of more than one independent variable. For an example, when fitting a plane to a set of height measurements, the plane is a function of two independent variables, x and z, say. In the most general case there may be one or more independent variables and one or more dependent variables at each data point.

$$r_i = y_i - f(x_i, \beta).$$

3 Genetic Algorithm

GA was developed to model adaptation processes mainly operating on binary strings and using a recombination operator with mutation as a background operator. The GA maintains a population of chromosomes, $P(t) = x_1^t$, ..., x_n^t for generation t. Each chromosome represents a potential solution to the problem and is implemented as some data structure Ch. Each solution is evaluated to give some measure of its "fitness". Fitness of a chromosome is assigned proportionally to the value of the objective function of the chromosomes. Then, a new population (generation $t + 1$) is formed by selecting more fit chromosomes (selection step). Some members of the new population undergo transformations by means of "genetic" operators to form new solution. There are unary transformations m_i (mutation type), which create new chromosomes by a small change in a single chromosome ($m_i : Ch \to Ch$), and higher order transformations c_j (crossover type), which create new chromosomes by combining parts from several chromosomes ($c_j : Ch \times \ldots \times Ch \to Ch$). After some number of generations the algorithm converges – it is expected that the best chromosome represents a near-optimum (reasonable) solution. The combined effect of selection, crossover and mutation gives so-called reproductive scheme growth equation [16]:

$$\xi\left(Ch, t+1\right) \geq$$

$$\xi\left(Ch, t\right) \cdot eval\left(Ch, t\right) / \bar{F}\left(t\right) \left[1 - p_c \cdot \frac{\delta\left(Ch\right)}{m-1} - o\left(Ch\right) \cdot p_m\right]$$

```
begin
    i = 0
    Initial population P(0)
    Evaluate P(0)
    while (not done) do
    (test for termination criterion)
    begin
        i = i + 1
        Select P(i) from P(i − 1)
        Recombine P(i)
        Mutate P(i)
        Evaluate P(i)
    end
end
```

Fig. 1. Pseudocode for GA

The structure of the herewith used GA is shown by the pseudocode below (Figure 1).

Three model parameters are represented in the chromosome – μ_{max}, k_S and $Y_{S/X}$. The following upper and lower bounds of the model parameters are considered [28]:

$$0 < \mu_{max} < 0.7,$$

$$0 < k_S < 1,$$

$$0 < Y_{S/X} < 30.$$

Roulette wheel, developed by Holland [17] is the herewith used selection method. The probability, p_i, for each chromosome is defined by:

$$p[\text{Individual } i \text{ is chosen}] = \frac{F_i}{\sum_{j=1}^{PopSize} F_j}, \tag{4}$$

where F_i equals the fitness of chromosome i and $PopSize$ is the population size.

To reproduce the chromosomes simple crossover and binary mutation according to [28] are applied. In proposed genetic algorithm fitness-based reinsertion (selection of offspring) is used.

For the considered here model parameter identification, the type of the basic operators in GA are as follows [28]:

- encoding – binary,
- fitness function – linear ranking,
- selection function – roulette wheel selection,
- crossover function – simple crossover,
- mutation function – binary mutation,
- reinsertion – fitness-based.

The values of GA parameters are [28]:

- generation gap, ggap = 0.97,
- crossover probability, xovr = 0.75,
- mutation probability, mutr = 0.01,
- maximum number of generations, maxgen = 200.

4 Ant Colony Optimization (ACO)

The ACO is a stochastic optimization method that imitates the behavior of real ants colonies. They manage to establish the shortest rout to nutrishment sources and back. Real ants foraging for food lay down quantities of pheromone (chemical cues) marking the path that they follow. An isolated ant moves essentially at random but an ant encountering a previously laid pheromone will detect it and decide to follow it with high probability and thereby reinforce it with a further quantity of pheromone. Thus if more the ants follow a trail, the more attractive that trail becomes. The original idea comes from observing the exploitation of food resources among ants, in which ants' individually limited cognitive abilities have collectively been able to find the shortest path between a food source and the nest.

The ACO is usually implemented as a team of intelligent agents, which simulate the ants behavior, walking around the graph representing the problem to solve, using mechanisms of cooperation and adaptation. The requirements of the ACO algorithm are as follows [8, 13]:

- The problem needs to be represented appropriately, which would allow the ants to incrementally update the solutions through the use of a probabilistic transition rules, based on the amount of pheromone in the trail and other problem specific knowledge.
- A problem-dependent heuristic function, that measures the quality of components that can be added to the current partial solution.
- A rule set for pheromone updating, which specifies how to modify the pheromone value.
- A probabilistic transition rule based on the value of the heuristic function and the pheromone value, that is used to iteratively construct a solution.

The structure of the ACO algorithm is shown by the pseudocode below (Figure 2).

The transition probability $p_{i,j}$, to choose the node j when the current node is i, is based on the heuristic information $\eta_{i,j}$ and the pheromone trail level $\tau_{i,j}$ of the move, where $i, j = 1, \ldots, n$.

$$p_{i,j} = \frac{\tau_{i,j}^a \eta_{i,j}^b}{\sum\limits_{k \in Unused} \tau_{i,k}^a \eta_{i,k}^b}, \tag{5}$$

```
Ant Colony Optimization
Initialize number of ants;
Initialize the ACO parameters;
while not end-condition do
        for k = 0 to number of ants
            ant k choses start node;
            while solution is not constructed do
                    ant k selects higher probability node;
            end while
        end for
        Update-pheromone-trails;
end while
```

Fig. 2. Pseudocode for ACO

where $Unused$ is the set of unused nodes of the graph.

The higher the value of the pheromone and the heuristic information, the more profitable it is to select this move and resume the search. In the beginning, the initial pheromone level is set to a small positive constant value τ_0; later, the ants update this value after completing the construction stage. The ACO algorithms adopt different criteria to update the pheromone level.

The pheromone trail update rule is given by:

$$\tau_{i,j} \leftarrow \rho\tau_{i,j} + \Delta\tau_{i,j}, \tag{6}$$

where ρ models evaporation in the nature and $\Delta\tau_{i,j}$ is new added pheromone which is proportional to the quality of the solution. Thus better solutions will receive more pheromone than others and will be more desirable in a next iteration.

The values of ACO parameters in our application are :

- evaporation parameter $\rho = 0.5$,
- $a = b = 1$,
- number of generations = 200.

5 Numerical Results and Discussion

All computations are performed using a PC/Intel Core i5-2320 CPU @ 3.00GHz, 8 GB Memory (RAM), Windows 7 (64 bit) operating system and Matlab 7.5 environment.

A series of numerical experiments are performed to evaluate the influence of the population size in GAs and ACO on the accuracy of the obtained solution. Using mathematical model of the *E. coli* cultivation process (Eq. (1) - Eq. (3)) the model parameters – maximum specific growth rate (μ_{max}), saturation constant (k_S) and yield coefficient ($Y_{S/X}$) – are estimated. For the identification procedures consistently different population sizes (from 5 to 200 chromosomes in

the population for GA and from 5 to 100 ants for ACO) are used. The number of generations is fixed to 200. Because of the stochastic characteristics of the applied algorithms, series of 30 runs for each population size are performed.

In the Table 1 and 2, obtained average, best and worst objective function values for considered population sizes of GA and ACO respectively, are presented. The results observed for computational time are listed in Table 3 and 4 respectively.

Table 1. GA algorithm performance for various population sizes - objective function

Population size	**Objective function** J		
	Average	**Best**	**Worst**
5	6.1200	4.8325	9.2958
10	5.8000	4.8548	9.6175
20	4.7660	4.4753	5.3634
30	4.6519	4.4816	5.0094
40	4.6359	4.4437	4.9669
50	4.6070	4.4488	4.8636
60	4.5886	4.4625	4.8013
70	4.5648	4.4384	4.7357
80	4.5782	4.4474	4.7463
90	4.5711	4.4496	4.7211
100	4.5406	4.4252	4.7017
110	4.5455	4.4332	4.7319
150	4.5511	4.4575	4.6717
200	4.5453	4.4359	4.7206

The numerical experiments show that increasing the size of the population of 5 to 100 chromosomes significantly improves the resulting value of the objective function (average results) – from 6.1200 to 4.5406 (see Table 1). The further increase in the size of population (more than 100 chromosomes) does not lead to more accurate results. The subsequent increase in the population size leads only to an increase in computational time without improving the value of the objective function (average results) – from 26.8644 s (100 chromosomes) to 52.4782 s (200 chromosomes) vs. $J = 4.5406$ to $J = 4.5453$ (see Table 3).

Similar conclusions can be made for ACO algorithm performance regarding Tables 2 and 4. The numerical experiments show that increasing the population size from 5 to 70 ants improve the accuracy of the achieved average result – from $J = 6.2523$ to $J = 5.1350$. Further increase of the size of the population only increases the computational time without significant improvement of the results.

The best value of the objective function achieved by GA is similar to this achieved by ACO, but the average value achieved by GA is better. One generation performed by ACO is much slower than one generation performed by GA, thus the GA computational time is less than ACO computational time. We

Table 2. ACO algorithm performance for various population sizes - objective function

Population size	**Objective function J**		
	Average	**Best**	**Worst**
5	6.2523	5.0652	7.9011
10	6.0527	4.8083	8.0956
20	5.4330	4.9293	6.5924
30	5.2849	4.7408	6.2202
40	5.2853	4.8004	6.0784
50	5.2206	4.6598	4.1695
60	5.2184	4.8983	5.7759
70	5.1350	4.7739	5.6652
80	5.1324	4.8078	5.7891
90	5.1415	4.7856	5.6120
100	5.0885	4.8382	5.4866

Table 3. GA algorithm performance for various population sizes - computational time

Population size	**Computational time, s**		
	Average	**Best**	**Worst**
5	4.9457	4.5552	5.6004
10	6.0039	5.6316	6.3648
20	7.6482	7.3008	7.9561
30	11.1115	10.8265	11.5129
40	12.9824	12.4957	13.3537
50	14.9087	14.3989	15.5377
60	17.2766	16.6141	20.3113
70	19.7601	19.1725	20.0617
80	22.1880	21.7153	22.6669
90	24.3414	23.9150	24.8198
100	26.8644	26.4890	27.8306
110	29.7057	29.1878	30.2642
150	39.7273	39.1407	40.3887
200	52.4782	51.3087	55.8952

Table 4. ACO algorithm performance for various population sizes - comput. time

Population size	**Computational time, s**		
	Average	**Best**	**Worst**
5	16.8065	16.5673	17.0509
10	29.5950	29.3126	29.8274
20	55.0699	54.1323	56.7376
30	90.1941	88.9674	91.1202
40	111.2729	109.2163	116.0803
50	131.8193	131.0720	133.4745
60	151.7526	148.8406	159.7606
70	173.8225	172.5839	177.0143
80	197.8873	196.5457	199.5877
90	234.9069	232.1607	238.2759
100	260.4468	258.3689	268.0097

can conclude that GA performs better than ACO for this problem and the best population size for GA is 100 chromozoms.

For better interpretation the obtained numerical results are graphically visualized in the next figures. On Figure 3 the objective function values, obtained during the 30 GA runs for 5, 10, 20 and 30 chromosomes in the population, are shown. The graphical results show that the GA could not find accurate solution using small population size – 5 or 10 chromosomes. It needs at least 20 chromosomes in population for achieving a better solution. On Figure 4 the objective function values, obtained during the 30 algorithm runs for 100, 110, 150 and 200 chromosomes in the population, are shown. Here, it could be seen that using large population size (110, 150 or 200 chromosomes) did not result in an improvement of the objective function values.

The ANOVA test is applied and the values of the objective function for population size equal and more than 100 are statistically equal. Moreover, as can be seen from Figure 6 increasing the population size result in an acceleration of computational time. When the population size increases it leads to increase of the needed computational resources like time and memory which can be a problem for large-scale tests. Therefore we can conclude that populations with 100 individuals is optimal with respect to the value of the objective function and the needed computational resources.

All numerical experiments for the influence of the population size on the objective function value and on the computational time are summarized in Figure 5 and Figure 6. It can be concluded that for the considered here non-linear cultivation model parameter identification problem the optimal population size is 100 chromosomes in the population (for 200 generations).

In Table 5 the best parameter values (μ_{max}, k_S and $Y_{S/X}$), obtained using GA with 100 chromosomes in the population, are presented. According to [10, 19, 31] the values of the estimated model parameters are in admissible boundaries.

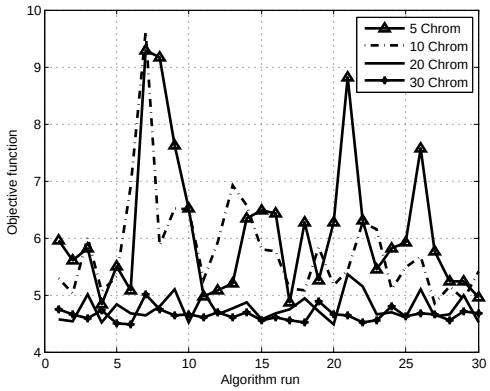

Fig. 3. Objective function values obtained during the 30 algorithm runs for 5, 10, 20 and 30 chromosomes in the population

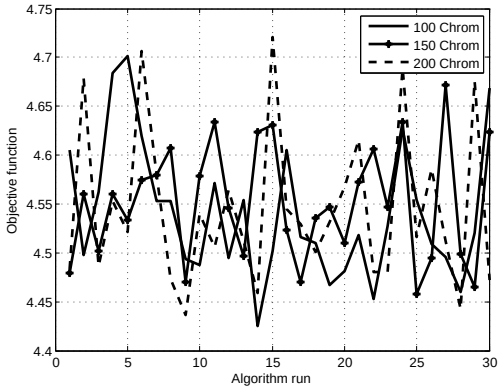

Fig. 4. Objective function values obtained during the 30 algorithm runs for 100, 150 and 200 chromosomes in the population

Table 5. Best parameter values of the model (100 chromosomes)

Parameter	**Value**
μ_{max}, [1/h]	0.4881
k_S, [g/l]	0.0120
$Y_{S/X}$, [-]	2.0193

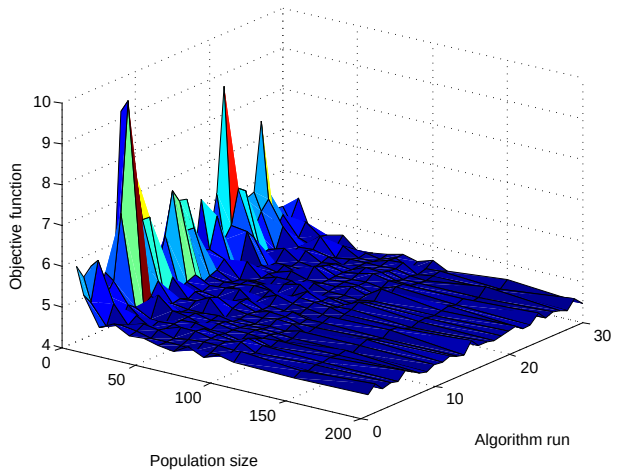

Fig. 5. Influence of the population size on the objective function value

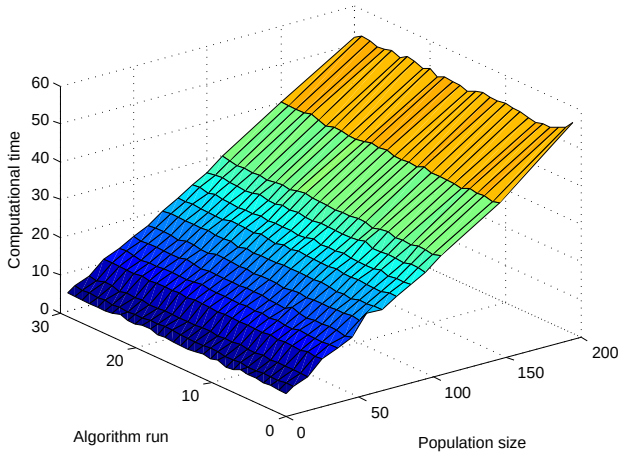

Fig. 6. Influence of the population size on the computational time

6 Conclusion

A good selection of the algorithm parameters improve both computation time and solution accuracy. Finding good parameter values is not a trivial task and requires human expertise as well as time. In this paper, the influence of one of the key GA and ACO parameters (population size) on the algorithm performance, is studied. As a test problem, the *E. coli* fed-batch cultivation model parameter identification, is considered. The three model parameters (maximum specific growth rate (μ_{max}), saturation constant (k_S) and yield coefficient ($Y_{S/X}$)) are identified. For a fixed number of the generations (200) different population sizes of the GA and ACO are explored. The numerical experiments are started with 5 chromosomes or ants in the population and consistently increased to 200 chromosomes for GA and 100 ants for ACO. The obtained results show that the optimal population size, for the GA considered here case study, is 100 chromosomes and 70 ants for ACO. Thus, accurate model parameters values are obtained with reasonable computational efforts. The use of smaller populations result in lower accuracy of the solution, obtained for a smaller computational time. The further increase of the population size increases the accuracy of solution. This effect is observed to a population size of 100 chromosomes for GA and 70 ants for ACO. The use of larger populations does not improve the solution accuracy and only increase the needed computational resources. The GA algorithm performs better than ACO for this application. It is faster and achieves better average value of objective function.

Acknowledgment

Work presented here is a part of the Poland-Bulgarian collaborative Grant "Parallel and distributed computing practices" and by European Commission project ACOMIN.

References

1. Akpinar S., G. M. Bayhan G. M.: A Hybrid Genetic Algorithm for Mixed Model Assembly Line Balancing Problem with Parallel Workstations and Zoning Constraints, Engineering Applications of Artificial Intelligence, Vol. 24, No 3, pp. 449–457, (2011).
2. Alander J. T.: On optimal population size of genetic algorithms, In Proceedings of the IEEE Computer Systems and Software Engineering, pp. 6569, (1992).
3. Al-Duwaish H. N.: A Genetic Approach to the Identification of Linear Dynamical Systems with Static Nonlinearities, International Journal of Systems Science, Vol. 31, No. 3, pp. 307-313, (2000).
4. Arndt M., Hitzmann B.: Feed Forward/feedback Control of Glucose Concentration during Cultivation of *Escherichia coli*, 8th IFAC Int. Conf. on Comp. Appl. in Biotechn, Canada, pp. 425-429, (2001).
5. Bartz-Beielstein T.: Experimental Research in Evolutionary Computation: The New Experimentalism, *Natural Computing Series*, Springer, (2006).

6. Bastin G., Dochain D.: *On-line Estimation and Adaptive Control of Bioreactors*, Els. Sc. Publ, (1991).
7. Benjamin K. K., Ammanuel A.N., David A., Benjamin Y.K.: Genetic Algorithm using for a Batch Fermentation Process Identification, J of Applied Sciences, Vol. 8, No. 12, pp. 2272-2278, (2008).
8. Bonabeau E., Dorigo M., Theraulaz G.: *Swarm Intelligence: From Natural to Artificial Systems*, New York,Oxford University Press, (1999).
9. Clune J., Goings S., Punch B., Goodman E.: Investigations in meta-gas: panaceas or pipe dreams, In GECCO 05 Proceedings, pp. 235241, (2005).
10. Contiero J., Beatty C., Kumari S., DeSanti C. L., Strohl W. L., WolfeA.: Effects of mutations in acetate metabolism on high-cell-density growth of *Escherichia coli*, Journal of Industrial Microbiology and Biotechnology, Vol. 24, pp. 421-430, (2000).
11. da Silva M. F. J., Perez J. M. S., Pulido J. A. G., RodriguezM. A. V.: AlineaGA - A Genetic Algorithm with Local Search Optimization for Multiple Sequence Alignment, Appl Intell, Vol. 32, pp. 164-172, (2010).
12. Diaz-Gomez P.A., Hougen D. F.: Initial Population for Genetic Algorithms: A Metric Approacs, Proceedings of the International Conference on Genetic and Evolutionary Methods, GEM 2007, Las Vegas, Nevada, USA, Hamid R. Arabnia and Jack Y. Yang and Mary Qu Yang (Eds), pp. 43-49,(2007).
13. Dorigo M., Stutzle T., *Ant Colony Optimization*, MIT Press,(2004).
14. Eiben Á. E., Hinterding R., Michalewicz Z.: Parameter Control in Evolutionary Algorithms, IEEE Transactions on Evolutionary Computation, Vol. 3, No. 2, (1999).
15. Fidanova S.: Simulated Annealing: A Monte Carlo Method for GPS Surveying, *Computational Science*, Lecture Notes in Computer Science No 3991, pp. 1009-1012, (2006).
16. Goldberg D. E.: *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley Longman, London, (2006).
17. Holland J. H.: *Adaptation in Natural and Artificial Systems*, 2nd Edn. Cambridge, MIT Press, (1992).
18. Koumousis V. K., Katsaras C. P.: A sawtooth genetic algorithm combining the effects of variable population size and reinitialization to enhance performance, IEEE Transactions on Evolutionary Computation, Vol. 10, No. 1, pp. 1928, 2006.
19. Levisauskas D., Galvanauskas V., Henrich S., Wilhelm K., Volk N., Lubbert A.: Model-based optimization of viral capsid protein production in fed-batch culture of recombinant *Escherichia coli*, Bioprocess and Biosystems Engineering, Vol. 25, pp. 255-262, (2003).
20. Lobo F. G., Goldberg D. E.: The parameterless genetic algorithm in practice, Information SciencesInformatics and Computer Science, Vol. 167, No. 1-4, pp. 217232, (2004).
21. Lobo F. G., Lima C. F.: A review of adaptive population sizing schemes in genetic algorithms, In Proceedings of the Genetic and Evolutionary Computation Conference, pp. 228234, (2005).
22. Nowotniak R., Kucharski J.: GPU-based Tuning of Quantum-Inspired Genetic Algorithm for a Combinatorial Optimization Problem, In Proceedings of the XIV International Conference System Modeling and Control, ISSN 978-83-927875-1-8, (2011).
23. Paplinski J. P.: The Genetic Algorithm with Simplex Crossover for Identification of Time Delays, *Intelligent Information Systems*, pp. 337-346, (2010).
24. Pelikan M., Goldberg D. E., Cantu-Paz E.: Bayesian optimization algorithm, population sizing, and time to convergence, Illinois Genetic Algorithms Laboratory, University of Illinois, Tech. Rep., (2000).

25. Reeves C. R.: Using Genetic Algorithms With Small Populations, In Proceedings of the Fifth International Conference on Genetic Algorithms, pp. 92-99, 1993.
26. Roeva O.: Improvement of Genetic Algorithm Performance for Identification of Cultivation Process Models, Advanced Topics on Evolutionary Computing, Book Series: Artificial Intelligence Series-WSEAS, pp. 34-39, (2008).
27. Roeva O., Slavov Ts.: Fed-batch Cultivation Control based on Genetic Algorithm PID Controller Tuning, Lecture Notes on Computer Science, Springer-Verlag Berlin Heidelberg, Vol. 6046, pp. 289-296, (2011).
28. Roeva, Fidanova S.: Chapter 13. A Comparison of Genetic Algorithms and Ant Colony Optimization for Modeling of *E. coli* Cultivation Process, In book Real-World Application of Genetic Algorithms, In Tech, pp. 261-282, (2012).
29. Saremi A.,ElMekkawy T. Y., Wang G. G.: Tuning the Parameters of a Memetic Algorithm to Solve Vehicle Routing Problem with Backhauls Using Design of Experiments, International Journal of Operations Research, Vol. 4, No. 4, pp. 206-219, (2007).
30. Piszcz A., Soule T.: Genetic programming: Optimal population sizes for varying complexity problems, In Proceedings of the Genetic and Evolutionary Computation Conference, pp. 953954, (2006).
31. Zelic B., Vasic-Racki D., Wandrey C., Takors R.: Modeling of the pyruvate production with *Escherichia coli* in a fed-batch bioreactor, Bioprocess and Biosystems Engineering, Vol. 26, pp. 249-258, (2004).