Comparison of different algorithms for simultaneous estimation of multiple parameters in kinetic metabolic models

Syed Murtuza Baker*, Kai Schallau, Björn H. Junker

Systems Biology Group, Leibniz Institute of Plant Genetics and Crop Plant Research, Gatersleben, Germany.

Summary

Computational models in systems biology are usually characterized by a lack of reliable parameter values. This is especially true for kinetic metabolic models. Experimental data can be used to estimate these missing parameters. Different optimization techniques have been explored to solve this challenging task but none has proved to be superior to the other. In this paper we review the problem of parameter estimation in kinetic models. We focus on the suitability of four commonly used optimization techniques of parameter estimation in biochemical pathways and make a comparison between those methods. The suitability of each technique is evaluated based on the ability of converging to a solution within a reasonable amount of time. As most local optimization methods fail to arrive at a satisfactory solution we only considered the global optimization techniques. A case study of the upper part of Glycolysis consisting 15 parameters is taken as the benchmark model for evaluating these methods.

1 Introduction

The main goal of Systems Biology is an integrated description of the functionality of living organisms. One important aspect are metabolic networks, which are extremely complex, highly interconnected and regulate vital cellular processes for all living organisms. To understand this complex behavior it is helpful to translate a metabolic network into a dynamic model with a rate law for each enzymatic reaction. These rate laws are defined as mathematical expressions that heavily depend on the underlying mechanism of the enzymatic reactions and can become quite complex with a large quantity of parameters. Therefore system-level computational approaches are required to model and understand these mechanisms. To model the system as accurately as possible, it is important to have a complete and accurate set of parameters which characterize the system. However it is not always possible to measure these parameters in wet lab experiments due to high demands on cost and time. Furthermore there are certain paramters for which there are no appropriate measurement methods yet established. As a result different *insilico* methods have been proposed for parameter estimation which greatly reduce the effort and cost of biological experiments.

Parameter estimation is considered to be an inverse problem because the methods calibrate the model parameters to reproduce the experimental results in the best possible way. This is an inverse approach as it solves problems by minimizing a cost function which quantifies the value of the difference between the model-simulated data and the measurement data which

*To whom correspondence should be addressed. E-mail:baker@ipk-gatersleben.de

it already have. Different methods, both global and local, are proposed in this regard. Both types of algorithms have advantages and disadvantages. Local optimization methods tend to converge quickly whereas global methods might take time. Local optimization methods have a tendency to get stuck in local optima whereas global optimization methods ensure the global optimum. The latter is quite important for biological systems since in most cases there is no clear indication about the order of magnitude of these parameters.

The main focus of this paper is to study and compare different parameter estimation algorithms, namely Evolutionary Programming, Genetic Algorithm, Simulated Annealing and Particle Swarm Optimization. A small exemplary model of the upper part of Glycolysis with 15 parameters was implemented as the case study of a non-linear dynamic model. The results of different parameter estimation methods applied in this case study are then analyzed and compared.

In the following sections, the problem statement will be illustrated before describing the basic concepts of global optimization methods. The next selection deals with the implementation of the model, and the results obtained from the algorithms will be discussed. Finally, we will conclude with an outline of future work.

2 Problem Statement

Estimating parameters in a nonlinear dynamic model is complex due to its nonlinearity for which no general solution exists [12]. Biological models are mostly dynamic in nature and highly nonlinear. Because of this parameter estimation of biological models are dependent on optimization techniques. Most current methods formulate this problem as a nonlinear dynamic optimization problem with differential algebraic constraints and a measure of the distance between model prediction and experimental data that is used as the objective function. The task is to minimize this objective function over a time series of data points. The objective function is formulated as below

$$J = \int_0^t (y_{mes}(t) - y_{pre}(\theta, t))^T W(t)(y_{mes}(t) - y_{pre}(\theta, t))\mathrm{d}t \tag{1}$$

where the algorithm tries to find the vector θ that minimizes the cost function J. θ is the vector of parameters in the optimization problem. Here $y_{mes}(t)$ is the vector of experimental measurement values of the state at time t, $y_{pre}(\theta, t)$ is the vector computed values of the state at time t. W(t) is a weighting matrix. Because of the nonlinear and dynamic nature of the system this optimization is typically multimodal (nonconvex). So if local methods are used to find a solution of the problem it is very likely that these methods will get stuck in local minima [9]. Thus global optimization methods are best suited to solve this class of problems.

3 Global Optimization Methods

Global optimization methods can be classified as deterministic [3, 5] and stochastic [1, 4]. Stochastic global optimization methods depend on probabilistic approaches. Because of random nature of these approaches they cannot guarantee a convergence to the global solution. On

the other hand deterministic methods do guarantee a global maximum, however, these methods cannot solve problems with certainty in finite time. Though cannot guarantee, stochastic methods can locate the vicinity of global optima quite efficiently most of the time and importantly it will find it with modest computational time. For this reason stochastic global optimization methods are typically used for paramter estimation.

There are a large number of stochastic global optimization methods. In this paper we discuss four of them and compare the performance of these algorithms for evaluating the example model used as the benchmark for testing these methods.

Evolutionary Computation (EC) is a very popular class of stochastic global optimization method. So this class of algorithms is based on the ideas of biological evolution, in particular the mechanisms of reproduction, mutation and survival of the fittest [2]. Just like biological evolution, the evolutionary computational methods generate better solutions by creating new generations from the one that were closest to the solution in the previous generation. Two algorithms in this class are very popular: Evolutionary Programming and Genetic Algorithms.

Evolutionary Programming (EP) is one of the most popular optimization techniques in the class of Evolutionary Computation. Like all Evolutionary Computation algorithms the mechanism in this algorithm inspired by biological evolution. Since its introduction it has been applied in different fields of optimization. It starts with a random initialization of the population of solution. For each individual of the population it then generates a mutation. After that it calculates the fitness of each individual. Based on this fitness value it keeps the best half while ignoring the rest. In this way the population size is maintained. Using this new population as the starting point the algorithm repeats this process until the solution is reached.

Genetic Algorithms (GA) are another very popular subclass of Evolutionary Computation. It is quite similar to Evolutionary Programming differing in that GA uses both crossover and mutation, with crossover as the primary search operator, while EP uses only mutation.

The Simulated Annealing (SA) method derives its name from annealing in metallurgy which is a technique involving the heating and controlled cooling of a material to increase the energy of its crystals to allow uphill move. Based on the analogy to this physical process each step of SA chooses the current solution by a random nearby solution. In the SA method, each point of the search space corresponds to a state of some physical system, and the objective function is similar to the internal energy of the system in that state. The method eliminates the problem of getting stuck in local minima by allowing 'uphill' moves [8].

The Particle Swarm Optimization (PSO) method was proposed by Kennedy and Eberhart year 1995 [10]. It is inspired by social behavior and movement dynamics of insects, birds and fishes. Its performance is comparable to that of GA. The swarm is typically modeled by particles that have a position and a velocity in multidimensional space. These particles roam through the hyperspace and have two essential reasoning capabilities: their memory of their own best position and information of their neighbour best. Members of a swarm communicate good positions to each other and adjust their own position and velocity based on these good positions. It is a very efficient global search algorithm because it is derivative-free and insensitive to the scaling onto design variables [10].

4 Model Implementation

For the sake of a fair comparison, all algorithms were tested using the same model in the same platform. The case study model was implemented in Copasi [6] which is one of the most popular software application for simulation and analysis of biochemical networks. The main reason for choosing this software is that all of the algorithms that were compared in this paper have already been implemented. Thus it is comparatively easy to model biochemical pathways and simulate a model without any programming effort. Furthermore the command prompt version of the software can be extended by scripting languages like Perl to run the simulation multiple times in order to predict the statistics of an estimation algorithm. One drawback to Copasi is that it is not easily extensible to include new optimization algorithms.

The upper part of Glycolysis was used as the case study model. This pathway is taken from Yeast as described by Hyne et. al. [7]. All parameter values are also taken from the same paper. The schematic diagram of the model is given in Fig.1.

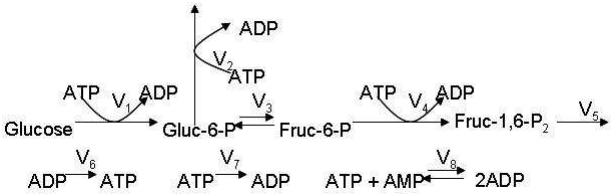


Figure 1: Schematic diagram of the case study model

The ordinary differential equations (ODEs) of this nonlinear dynamic model are:

$$\frac{d}{dt}Gluc6P = \frac{V_{\max,1}ATP}{K_{ATP,1}+ATP} - \mathrm{K_2ATP.Gluc6P} - \frac{\frac{V^f_{\max,3}}{K_{Gluc6P,3}}Gluc6P - \frac{V^r_{\max,3}}{K_{Fruc6P,3}}Fruc6P}{1+\frac{Gluc6P}{K_{Gluc6P,3}}+\frac{Fruc6P}{K_{Fruc6P,3}}}$$

$$\frac{d}{dt}Fruc6P = \frac{\frac{V^f_{\max,3}}{K_{Gluc6P,3}}Gluc6P - \frac{V^r_{\max,3}}{K_{Fruc6P,3}}Fruc6P}{1+\frac{Gluc6P}{K_{Gluc6P,3}}+\frac{Fruc6P}{K_{Fruc6P,3}}} - \frac{V_{\max,4}(Fruc6P)^2}{K_{Fruc6P,4}\left(1+\kappa\left(\frac{ATP}{ADP}\right)^2\right)+(Fruc6P)^2}$$

$$\frac{d}{dt}Fruc1,6P_2 = \frac{V_{\max,4}(Fruc6P)^2}{K_{Fruc6P,4}\left(1+\kappa\left(\frac{ATP}{ADP}\right)^2\right)+(Fruc6P)^2} - K_5Fruc1,6P_2$$

$$\frac{d}{dt}ATP = -\frac{V_{\max,1}ATP}{K_{ATP,1}+ATP} - K_2ATP.Gluc6P - \frac{V_{\max,4}(Fruc6P)^2}{K_{Fruc6P,4}\left(1+\kappa\left(\frac{ATP}{ADP}\right)^2\right)+(Fruc6P)^2} + \mathrm{K_6ADP - K_7ATP - K_{8,f}ATP.AMI} - K_{8,r}(Fruc6P)^2$$

$$\frac{d}{dt}ADP = \frac{V_{\max,1}ATP}{K_{ATP,1}+ATP} + \mathrm{K_2ATP.Gluc6I} + \frac{V_{\max,4}(Fruc6P)^2}{K_{Fruc6P,4}\left(1+\kappa\left(\frac{ATP}{ADP}\right)^2\right)+(Fruc6P)^2} - \mathrm{K_6ADP + K_7ATP + 2.(K_{8,f}ATP.AMP} - K_{8,r}(Fruc6P)^2)$$

$$\frac{d}{dt}AMP = -K_{8,f}ATP.AMP - K_{8,r}(Fruc6P)^2$$

where Glucose, Gluc-6-P, Fruc-6-P, Fruc-1,6-P_2, ADP, ATP, AMP are the six metabolites which are considered to be the variables. All the V_m and K_m are the parameters. There are 15 parameters in the model. Among those 15 parameters 4 parameters were estimated based on time course data. The general form of the Copasi objective function is:

$$S(P) = \sum_{i=1}^{n} \omega_i (x_i - y_i(P))^2 \tag{2}$$

which is minimized during paramter estimation. Equation 2 is the discrete form of the objective function presented in equation 1. These two equations are the same when n $\rightarrow \infty$. This is the same as equation 1 but just a different mathematical reformulation. Where $y_i(P)$ are the simulated data corresponding to the experimental data x_i, and ω_i is a weight used to normalize the objective function. The weights are calculated according to the formula $\omega_i = \frac{1}{|x_i|}$.

Table 1 describes the settings of the parameters for each of the algorithm.

Evolutionary Programming	**Simulated Annealing**
Number of Generations: 200	Start Temperature: 1
Population Size: 20	Cooling Factor: 0.85
Random Number Generator: 1	Tolerance: $1E^{-06}$
Seed: 0	Random Number Generator: 1
	Seed: 0
Genetic Algorithm	**Particle Swarm**
Number of Generations: 200	Iteration Limit: 2000
Population Size: 20	Swarm Size: 50
Random Number Generator: 1	Std. Deviation: $1E^{-06}$
Seed: 0	Random Generator: 1
	Seed: 0

Table 1: Search parameters used in different algorithms

5 Results and Discussion

For a fair comparison all of the computations were performed using a Intel Quad Core (2.66 GHz) platform running Windows XP. The method parameters were kept at the default values that COPASI starts with. This was made to ensure an equal starting point for the comparisons. Parameter values of all the algorithms can be adjusted differently depending on the model. One parameter set for a specific model might not perform well for other models. Keeping this in mind and also the complexity of adjusting these parameter values, we decided to keep the method parameters at their initial values to ensure easy comparability of the results. The values of the objective function are taken from 1.8 second before the algorithm finishes. Table 2 highlights the results obtained from running the computations 100 times for each algorithm. The four parameters that are estimated are k_2, $V^f_{max,3}$, $V_{max,4}$ and k_{8r}. The actual values of these four parameters are 2.26, 140.282, 44.7287 and 133.33 respectively [7]. The time series data is generated by integrating the ODEs over a time period of 100 arbitrary time steps. This time series data is later considered as a measurement data during parameter estimation.

From Table 2 it can be seen that among the two algorithms derived from EC algorithms the performance of EP is a little better than that of GA. But the mean values of both algorithms are far away from the actual value. This might be due to a known limitation of such algorithms for getting stuck in local minima. The median value of the algorithms reaches quite close to the actual value, which makes these algorithms still applicable to the parameter estimation problems. Table 3 depicts the maximum and minimum value of the objective function for each of the algorithms among the 100 runs. These values also validate the superiority of evolutionary programming over genetic algorithms as the objective function for evolutionary programming is (J_{max}= 119.569 and J_{min} = 9.6802E^{-10}) and for genetic algorithm it is (J_{max}= 129.588 and J_{min} = 0.003527).

Algorithm Name	Parameter Name	Mean	Standard Deviation	Median	CPU Time (second)
EP	k_2	71.67417	304.4333	2.26002	1104.615
	$V^f_{max,3}$	3263.349	13767.35	140.248	
	$V_{max,4}$	295.7599	1338.407	44.75795	
	k_{8r}	124.3258	31.42429	133.3105	
GA	k_2	81.57572	288.9584	2.25719	1053.612
	$V^f_{max,3}$	3616.948	13354.14	139.826	
	$V_{max,4}$	643.5873	2149.941	44.9594	
	k_{8r}	125.8438	30.14491	132.806	
SA	k_2	2.25	2.13E^{-05}	2.26	58924.72
	$V^f_{max,3}$	140.2818	0.00141	140.282	
	$V_{max,4}$	44.7287	0.000258	44.7287	
	k_{8r}	133.3299	0.001191	133.33	
PSO	k_2	2.26	2.677E^{-15}	2.26	6319.488
	$V^f_{max,3}$	140.282	2.57E^{-13}	140.282	
	$V_{max,4}$	44.7287	5E^{-14}	44.7287	
	k_{8r}	133.33	5.71E^{-14}	133.33	

Table 2: Results obtained by repeating the computation of the four algorithms on the case study model 100 times (CPU time is the total for 100 runs). Statistics for the data are calculated from these 100 runs.

Algorithm Name	Objective function value	
	Maximum	Minimum
EP	119.569	9.6802E^{-10}
GA	129.588	0.003527
SA	2.93E-08	7.83E^{-10}
PSO	5.06E-10	5.06E^{-10}

Table 3: Maximum and Minimum values of objective functions of the algorithms obtained after running the computation of each of the four algorithms 100 times. PSO has the best value in both minimum and maximum as they both become to be the same.

However the best result is obtained from PSO with the lowest objective function value (J_{max}= 5.06E^{-10} and J_{min} = 5.06E^{-10}). The mean and the median of the parameters are exactly the same as that of the actual values with low standard deviations. But the computational time

for PSO is 5-6 times longer than that of the EC algorithms. It takes 6319.488 seconds for the total of 100 runs. The performance of SA (J_{max}= 2.93E^{-08} and J_{min} = 58924.72) considering the accuracy is near to that of PSO . However it has the highest running time among all the algorithms (58924.72 seconds). SA may be more effective to find an acceptably good solution in a fixed amount of time, rather than the best possible solution. This is illustrated in Fig. 2 and Fig. 3. SA reaches to an acceptable value of the objective function very quickly but they take very long to converge to the best solution. Fig. 2 shows the value of the objective function for each of the algorithms after running the computation for only 5 seconds.

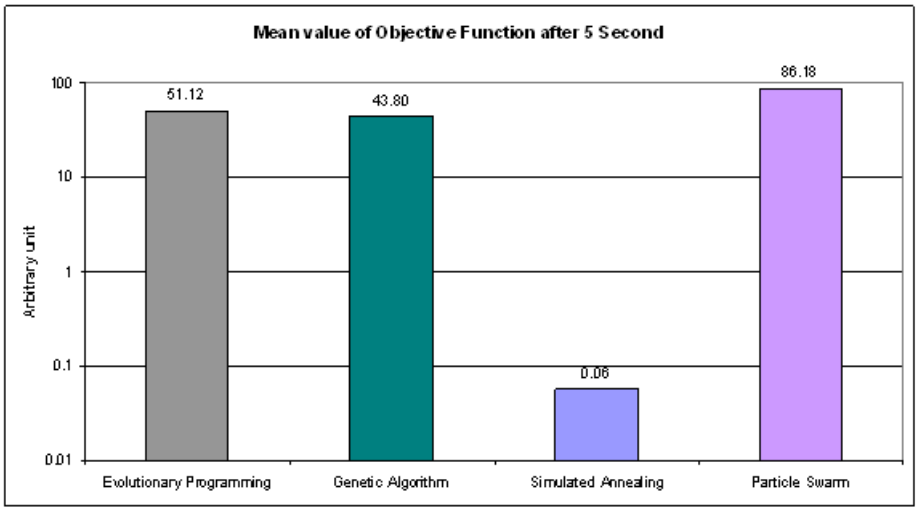
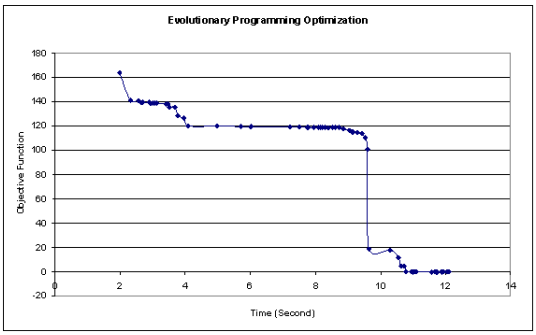


Figure 2: Mean value of the objective function after running each of the algorithm for 5 seconds. Though SA takes considerable longer time to reach the best objective function value, its converge rate for the first few steps is quite fast as it reaches to the lowest objective function value comparing the other algorithms.

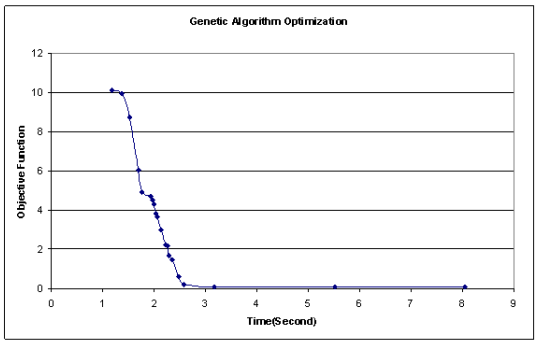
Fig.3 describes the objective function for all the algorithms. The figure shows that the objective function values as well as time needed to complete the process for each algorithm vary. As a result the scales for both abscissa and ordinate also differ. Fig. 3(a) describes the Evolutionary Programming. The algorithm takes about 12 seconds to find the optimized value. The starting value of the objective function is quite high but optimizes to a value of 0.006996 within quite short time. Fig. 3(b) depicts the activity of GA where the slope for the objective function is quite stiff as it decreases rapidly and reaches to a nearly optimized value within 3 second of the start. It further estimates the objective function value and stops on 0.066554 at 8.04 sec. Fig. 3(c) portrays SA. Though it reaches an objective function value less than 1 within 2.3 sec., it takes considerable long time (598.198 sec.) to reach at its optimized value. This could be considered as a limitation of this algorithm. Finally Fig. 3(d) shows that PSO is the best performing algorithm among these four with the test model data. It has the smallest objective function value of 5.06E^{-10} and it finds it within a reasonable time (42.2188 sec).

Considering accuracy PSO and SA outperforms EP and GA. But the computational time of the EC algorithms is much lower than that of either PSO or SA.

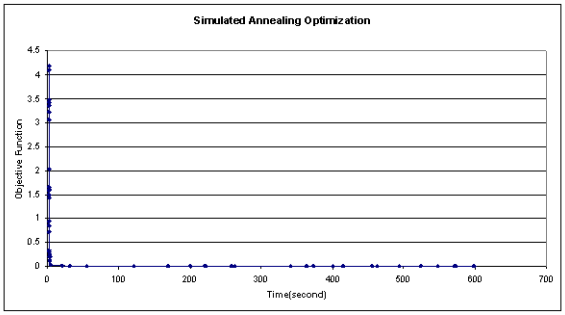
Our work confirms previous work by Mendes *et al*. [11]. Among the algorithms they tested, Simulated Annealing gives the best fit of the solution but it took considerable longer than other algorithms. They did not check the suitability of Particle Swarm Optimization. Moles *et al.*



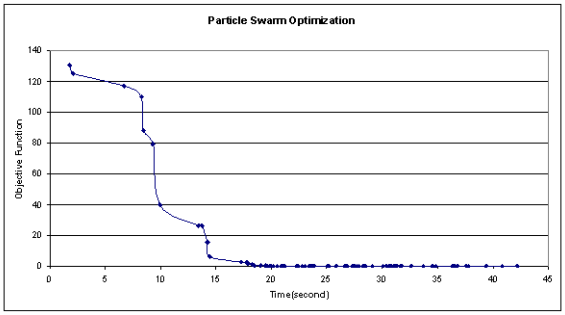
(a) Evolutionary Programming



(b) Genetic Algorithm



(c) Simulated Annealing



(d) Particle Swarm Optimization

Figure 3: Plot of the objective function with time for each algorithm. For each algorithm, the illustration plot is for the best solution obtained for that respective algorithm in the 100 repetition (for the best solution obtained from running each of the algorithms 100 times)

came to the conclusion that Evolutionary Strategy gives the best possible solution [9]. They mentioned in the paper that although pure genetic algorithms are by far the most popular evolutionary computing algorithm, methods derived from evolutionary strategy are more efficient and robust. They compared two methods derived from evolutionary strategies named SRES (Evolutionary Strategy using Stochastic Ranking) and CMA-ES(Covariance Matrix Adaptation-Evolutionary Strategy) and showed that SRES was the better of the two. But they excluded simulated annealing from the comparison citing poor performance with respect to their selected evolutionary strategies, the development which was their main motivation. They also did not mention anything specific about Particle Swarm Optimization.

6 Conclusion

In this paper several very popular global optimization methods have been analyzed. Although the stochastic global optimization methods perform better, their accuracy is relatively low. Some of those have the advantage of a short runtime but lack accuracy, whereas other are more accurate but take considerable longer time. What is most desired is a balance between the two; accuracy and computational time. However, it is well known that many stochastic methods lend themselves to parallelization very easily. In that case, provided suitable hardware, these

algorithms can be run within much shorter time. Other approaches such as nonlinear filtering might give better estimations within less running time. In our future work we will consider a hybrid approach. This will take the advantage of each mechanism to offset weekness that each of them have individually shown. In this hybrid approach we will start with SA which rapidly converges to a low objective function region and then will use GA to converge to the optimized value.

References

[1] Ali, M.M., Storey, C., and Törn, A ., Application of stochastic global optimization algorithms to practical problems. *J. Optim. Theory Appl.*, **95**, 545-563 (1997)

[2] Eiben A.E. and Smith J.E. , Introduction to Evolutionary Computing, Springer (2003)

[3] Grossmann, I.E., Global optimization in engineering design. Kluwer Academic Publishers, Dordrecht, The Netherlands (1996)

[4] Guus, C., Boender, E. and Romeijn, H.E., Stochastic methods. In Handbook of global optimization (eds.R. Horst and P.M. Pardalos) Kluwer Academic Publishers, Dordrecht, The Netherlands (1995)

[5] Horst, R. and Tuy, H., Global optimization: Deterministic approaches. Springer-verlag, Berlin (1990)

[6] Hoops, S., Sahle, S., Gauges, R., Lee, C., Pahle, J., Simus, N., Singhal, M., Xu, L., Mendes, P., and Kummer, U., COPASI - a COmplex PAthway SImulator. *BMC Bioinformatics*, **22**, 3067-3074 (2006)

[7] Hynne, F., Dano, S. and Sorensen, P.G. Fullscale model of glycolysis in *Saccharomyces cerevisiae, Biophys. Chem.*, **94**, 121-163 (2001).

[8] Kirkpatrick, S., Gelatt C., Vecchi M., Optimization by Simulated Annealing *Science. New Series*, **220**, 671-680 (1983)

[9] Moles C., Mendes P. and Banga J., Parameter Estimation in Biochemical Pathways: A Comparison of Global Optimization Methods. *Genome Res.*, **13**, 2467-2474 (2003)

[10] Russel C. Eberhart and James Kennedy. A new optimizer using particle swarm theory. *In Proceedings of the Sixth International Symposium on Micro Machine and Human Science MHS95, IEEE Press*, 3943 (October 1995)

[11] Pedro Mendes and Douglas B. Kell, Non-linear optimization of biochemical pathways: applications to metabolic engineering and parameter estimation. *BMC Bioinformatics*, **14**, 869-883 (1998)

[12] Rodriguez-Fernandez M., Egea J. and Banga J., Novel meta heuristic for parameter estimation in nonlinear dynamic biological systems. *BMC Bioinformatics*, **7**, 483 (2006)