Parameter Estimation in Computational Systems Biology Models: A Comparative Study of Initialization Methods in Global Optimization

Muhammad Akmal Remli[1]
Department of Data Science
Universiti Malaysia Kelantan, City Campus, Pengkalan Chepa, 16100
Kota Bharu, Kelantan, Malaysia

Nor-Syahidatul N.Ismail[2], Noor Azida Sahabudin[3]
Faculty of Computing, College of Computing and Applied Science
Universiti Malaysia Pahang
Pekan 26600, Pahang, Malaysia

Nor Bakiah Abd Warif [4]
Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia
Parit Raja 86400, Johor, Malaysia

***Abstract*—This paper compares different initialization methods and investigates their performance and effects on estimating kinetic parameters' value in models of biological systems. Estimating parameters values is difficult and time-consuming process due to their highly nonlinear and huge number of kinetic parameters involved. Global optimization method based on an enhanced scatter search (ESS) algorithm is a suitable choice to address this issue. However, despite its resounding success, the performance of ESS may decrease in solving high dimension problem. In this work, several choices of initialization methods are compared and experimental results indicated that the algorithm is sensitive to the initial value of kinetic parameters. Statistical results revealed that uniformly distributed random number generator (RNG) and controlled randomization (CR) that being used in ESS may lead to poor algorithm performance. In addition, the different initialization methods also influenced model accuracy. Our proposed methodology shows that initialization based on opposition-based learning scheme have shown 10% better accuracy in term of cost function.**

***Keywords*—Metaheuristic; opposition-based learning; kinetic parameters; initialization method; metabolic engineering**

I. Introduction

Kinetic models of living cells have drawn the attention of both practitioners and researchers in recent years [1]. Their applications are important in metabolic and bioprocess engineering as they facilitate scholars to better understand, accurately predict and consistently improve the desired products in systems biology [2,3]. The models are formulated by means of ordinary differential equations (ODEs) to mimic various functional behaviours such as glycolysis reactions via metabolic pathway and phosphorylationin signal transduction of human cells. Due to the highly nonlinear biological systems, building such model is considered both challenges and time-consuming [4].

One important aspect of model building is parameter estimation, which consists of finding the best possible value of kinetic parameters that produce best fit model to the experimental data. The goodness of fit can be measured by minimizing distance value in the simulated model and experimental data. Thus, searching best parameter values in kinetic model can be depicted as a nonlinear optimization problem [5] and this class of problem is difficult to be solved. In this view, various optimization algorithms have been proposed in parameter estimation and their findings revealed that local optimization often fails to obtain snear-optimal solution [6]. Although improvements such as iterated local search have been proposed, they still consume high computational cost. Consequently, global optimization which is based on metaheuristic is an ideal option to address this issue. Global methods are quite capable in parameter estimation problem as they are more likely to reach the global minimum compared to local methods.

Enhanced scatter search (ESS) is one of the metaheuristic algorithm which have recently shown to yield promising outcomes in biological problems [7,8]. The algorithm benefits from global exploration and local exploitation using various choices of local search. The balanced tradeoff between global and local methods in ESS has shown promising results in solving optimization problems. However, when dealing with high dimension problem involving hundreds of kinetic parameters, performance of most global methods including ESS are deteriorate. One of the most neglected mechanisms in global methods is the way they generate the initial solution which were commonly derived using random number generator (RNG). The initialization methods may influence the efficiency and performance of the optimization algorithm in terms of its probability in finding the global minima, convergence's rate and variance of statistical results [9]. To date, only a few works have been done for comparing initialization methods in optimization. So far, no comparative study with regard to initialization method has been done in large-scale parameter estimation problem, particularly in the biological domain. The limitation of existing work in the field of global optimization is they only rely on RNG for initialization and only focus on search operator or the way new solution are produced. The high complexity of the problem such as in biological domain or healthcare is challenging and applying optimization method must properly select the best initialization because it will influence the

output. Hence, this issue motivates this research to further investigate the effect of different initialization method.

This paper compares and investigates the effects of several initialization methods (also known as diversification generation method in ESS algorithm) from the context of parameter estimation in systems biology models. The evolutionary algorithm based on ESS is utilized in this study due to its efficiency and reliability in parameter estimation problem [7]. The paper is organized as follows: Section II explicates the problem statement in parameter estimations; Section III delineates ESS algorithm; Section IV introduces several initialization methods; Section V compares the methods and presents the discussion of their results and Section VI presents the conclusion of this study.

II. Problem Background

In a nonlinear kinetic model of biological systems, the parameter estimation problem deals with finding an unknown value of kinetic parameters to minimize a distance (objective or cost function) between simulated model and real data. The value of cost function determines the goodness of fit of the model. The observables, which are referred to as the output state variable, are experimentally measured. The cost function of this problem, which is also known as weighted nonlinear least squares J is defined as:

$$J = \sum_{exp=1}^{n_{exp}} \sum_{obs=1}^{n_{exp}^{obs}} \sum_{s=1}^{n_s^{exp,obs}} (ym_s^{exp,obs} - y_s^{exp,obs}(\boldsymbol{p}))^T W (ym_s^{exp,obs} - y_s^{exp,obs}(\boldsymbol{p})) \quad (1)$$

where n_{exp} is the number of experiments, n_{exp}^{obs} is the number of observables per experiment and $n_s^{exp,obs}$ is the number of samples per observable in each experiment. Time series experimental data is denoted as $ym_s^{exp,obs}$ and predicted model is denoted as $y_s^{exp,obs}(\boldsymbol{p})$. The kinetic parameters vector to be estimated is $\boldsymbol{p}$. The time span for observables is denoted as T and finally W represents the weight matrix to balance the contributions of the observables. Minimization of the above cost function is subject to the following constraints:

$$\dot{x} = f(x, \boldsymbol{p}, t) \quad (2)$$

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

$$y = g(x, \boldsymbol{p}, t) \quad (4)$$

$$\boldsymbol{p}^{lb} \leq \boldsymbol{p} \leq \boldsymbol{p}^{ub} \quad (5)$$

where derivative of $\dot{x}$ is the function f of ODEs model that describes the dynamics of biological systems, x_0 is the initial condition at time t_0, g is the observable functions and $\boldsymbol{p}^{lb}$ and $\boldsymbol{p}^{ub}$ are the lower bound and upper bound of the kinetic parameter vector $\boldsymbol{p}$ respectively. This nonlinear and multimodal problem consists of many local minima. Thus, the process of finding the global minima is both challenging and time-consuming.

III. Enhanced Scatter Search (ESS) Algorithm

An enhanced scatter search (ESS) is a metaheuristic that belongs to the family of evolutionary algorithms. This algorithm is similar with genetic algorithm (GA) with regards to maintaining and updating their population members and evaluating their cost function in an iterative cycle. However, unlike GA, ESS does not use crossover and mutation as their evolutionary operators. Instead, it uses the combination among members in a reference set (*RefSet*). In this study, four phases of ESS algorithm are used, namely: 1) initialization method, 2) *RefSet* update method, 3) *RefSet* member generation and combination method, and 4) hybrid of the local search method. More advance designs and their mechanism can be found in [10,11].

This algorithm starts with randomly generating m population of diverse vectors by means of initialization (diversification generation) method. The m size is ten times the problem size to ensure that the large initial solutions in the search space are widely sampled, thus increasing the chances of avoiding local minima. Although uniformly distributed random number is a popular method usually utilized to generate initial solutions in various optimization algorithms, there are other strategies that may provide better initial solutions. Therefore, we compared and investigate four different initialization methods which will be briefly discussed in Section IV.

After the diverse vectors are generated, each vector is evaluated and half of the *RefSet* members $b/2$ (b is the *RefSet* size) is formed. The diversification method produces high quality initial *RefSet* member. The remaining *RefSet* members are choosen from the *RefSet* by random cycle to complete a *RefSet*. Then, the subset generation produces pairs of members in *RefSet*. Let us consider members of a *RefSet*, x^i, to be combined with the rest of members in *RefSet*, $x^j, \forall i, j \in [1,2,..b], i \neq j$. The pairs of the combination ($combi_1$ and $combi_2$) are defined as follows:

$$combi_1 = x^i - m(1 + \gamma \cdot \delta) \quad (6)$$

$$combi_2 = x^i + m(1 - \gamma \cdot \delta) \quad (7)$$

where

$$m = \frac{x^j - x^i}{2} \quad (8)$$

$$\gamma = \begin{cases} 1 \; if \; i < j \\ -1 \; if \; j < i \end{cases} \quad (9)$$

and

$$\delta = \frac{|j-i|-1}{b-2} \quad (10)$$

Every pair of combination ($combi_1$ and $combi_2$) in the *RefSet* members is used to create new hyper-rectangles which are defined by their relative positions and distance and thus, resulting in a new solution within them. The hyper-rectangles based combination methods are applied and are defined in the following equation:

$$x^{new} = combi_1 + R \cdot (combi_2 - combi_1) \quad (11)$$

where x^{new} is new solution generated and R is the random number, $R \sim U([0,1])$. This combination strategy is similar to the mutation operator in differential evolution (DE) [12,13], which is effective in updating population members. In ESS,

the vectors of combination ($combi_1$ and $combi_2$) are systematically generated. They are not randomly generated, as practiced in DE. Using this combination strategy, every *RefSet* member generates a hyper-rectangle among the rest of *RefSet* member. The new number of solution produces $b-1$ solution for each *RefSet* member. After this strategy is implemented, a new solution (offspring) is generated with different distance and direction around their *RefSet* members (parents). In this case, if the offsprings have better (lesser) fitness value compared to their parents, the current solutions will be replaced. Otherwise, the same *RefSet* members will be used for the next iteration. In order to accelerate convergence, gradient local search is performed using Sequential Quadratic Programming (SQP). The algorithm is applied using *fmincon* solver in MATLAB. This solver minimizes the cost function using the results obtained in ESS using different vectors. If the solution obtained by *fmincon* outperforms the solution generated by ESS, the solution from *fmincon* will replace the current solution and it will in turn be added to *RefSet* members for further update. Otherwise, the solution from *fmincon* will be discarded. This process is repeated until the stopping criteria are met.

IV. Initialization Methods

We implement five initialization methods in the ESS algorithm in order to compare and investigate their effects on parameter estimation. The methods are random number generator (RNG), controlled randomization (CR), opposition-based learning (OBL), quasi-opposition learning (QOBL) and chaotic (Tent) map.

A. Random Number Generator (RNG)

The most commonly used initialization method in optimization algorithms is random number generator (RNG). RNG is defined as below: Let $X_i(x_{i,1}, x_{i,2}, \dots, x_{i,D})$ be the ith member of the population, each $x_{i,j}$ is generated between lower and upper bound (lb_i,ub_j). In summary, it generates uniformly distributed random numbers as in the following equation:

$$x_{i,j} = ub_j + R \cdot (lb_j - ub_j), j = 1, \dots, D \quad (12)$$

where R is the random numbers between 0 and 1. The vector of $X_{i,j}$ contains a list of random initial population generated between lower bound and upper bound $[lb, ub]$ for each of the variable.

B. Controlled Randomization (CR)

Unlike RNG, controlled randomization (CR) strategy generates the first five populations ($n = 5$) of equal size for each vector as in the following equation [14]:

$$x_{i,j} = \frac{R \cdot (npar) + i - 1}{n} \quad (13)$$

where $x_{i,j}$ is the vector of candidate solutions, R is the random numbers in the between 0 and1, and $npar$ is the number of kinetic parameters. After the first five vectors are generated, the remaining vectors are generated randomly and all initial solutions are put in the boundaries:

$$x_{new} = x_i \cdot (ub - lb) + lb \quad (14)$$

where lb and ub are lower and upper bounds, respectively. This strategy generates a set of diverse vectors which contain equal sizes of range in the first five vectors and other random vectors lie in sixth vector to m diverse vectors. It should be noted that ESS algorithm used CR strategy as its default initialization method [11].

C. Opposition-based Learning (OBL)

Opposition-based learning (OBL) is introduced in the field of computational intelligence [15]. This scheme is subsequently applied in optimization areas [16]. The basic idea of OBL is to generate a set of opposite numbers from first initial solutions generated by RNG, as follows: Let $x \in [lb, ub]$ is a random value. The opposition value of x is defined by:

$$\breve{x} = lb + ub - x \quad (15)$$

Based on Eq. (15), the opposite point for optimization in dimension space *D* is defined as follows:

Let $X_i(x_{i,1}, x_{i,2}, \dots, x_{i,D})$ be the i th member of the population and each member $x_{i,j}$ be bounded by (lb_i,ub_j) and $x_i \in [lb_i, ub_i], \forall i \in \{1,2,..D\}$. Thus, the opposite value of $\breve{X}_\iota = (\breve{x}_{i,1}, \breve{x}_{i,2}, .., \breve{x}_D)$ is defined as:

$$\breve{x}_{i,j} = lb_i +, ub_i - x_{i,j}, j = 1, \dots, D \quad (16)$$

Both X and $\breve{X}$ is merged. Now, let us assume $F(x)$ is the cost function in minimization problem. If cost function value of $f(\breve{x})$ is smaller than $f(x)$, $f(\breve{x}) < f(x)$, point X can be replaced with $\breve{X}$. Otherwise, point X will stay in the current population. All the population members will be evaluated and the initial population with fittest members among X and $\breve{X}$ is formed.

D. Quasi Opposition-based Learning (QOBL)

Another family of OBL is quasi-opposition learning (QOBL) which modified version of OBL that increases population uniformity [17]. Considering the opposite point in equation 16, the middle point $m_j = \{m_1, m_2, .., m_D\}$ is calculated as follows:

$$m_j = \frac{lb_j + ub_j}{2}, \forall j \in \{1,2,..D\}. \quad (17)$$

Then, quasi-opposite point $\breve{X}^Q = (x_1^Q, x_2^Q, .. x_n^Q)$ is selected randomly within the range of opposite points of $\breve{X}$ and middle point m:

$$\breve{x}_i^Q = \begin{cases} R \cdot (mj, \breve{x}_{i,j}) \; if \, x_{i,j} \le m_j \\ R \cdot (\breve{x}_{i,j}, mj) \; if \, x_{i,j} > m_j \end{cases} \quad (18)$$

where R is a random value drawn uniformly in the range of lower bound and upper bound. Like OBL, $\breve{X}^Q$ and X are merged and the best solution is chosen, that is, the fitness of the cost function.

E. Chaotic Map

Another alternative of uniformly distributed random numbers for diversification generation is chaotic map. This approach is based on the deterministic and chaotic systems and it is not necessarily random. In this paper, we investigate

one family of the chaotic map, which is Tent map [18] which is defined as:

$$x_{i,j}^{(k+1)} = \begin{cases} \frac{x_{i,j}^{(k)}}{0.7} x_{i,j} < 0.7 \\ \frac{10}{3}\left(1 - x_{i,j}^{(k)}\right) x_{i,j} \geq 0.7 \end{cases} \quad (19)$$

where $x_{i,j}^{(k+1)}$ is j th variable of i th individual in k th iteration. $x_{i,j}^{(k)}$ is the initial variable that is generated randomly using RNG. In this strategy, solutions which are generated from Tent map are not predictable and are highly sensitive to initial variables.

V. Result and Discussion

A large-scale model is used to test the different initialization methods in ESS algorithm. The model involves dynamic processes that reproduce the response to a pulse in extracellular glucose concentrations of central carbon metabolism (CCM) in *E. coli*. This model consists of 18 metabolites: 17 internal metabolites in cytosol and 1 extracellular metabolite (*glucose*) in extracellular compartment.

These metabolites consists of PEP, G6P, PYR, F6P, G1P, 6PG, FDP, GAP, CPEP, CG6P, CPYR, CF6P,GLCex, CG1P, CPG, CFDP, CGAP and Glucose. The model also contains 48 reactions coupled with 166 kinetic parameters. The mathematical formulation and description of this model can be found in [19]. Table I summarizes the characteristics of CCM E. coli model.

TABLE I. Characteristics of the Central Carbon Metabolism (CCM) in E.coli

Number of kinetic parameters	Dynamic metabolites	Observed metabolites	Noise level	Lower value	Upper value
116	18	9	Real	$0.1 \times p_{ori}$	$10 \times p_{ori}$

Note: For fair comparison, lower and upper bound are set as a function of p_{ori}, where p_{ori} is a set of kinetic parameters obtained from original publication. In this data, only observed metabolities are measured.

In order to obtain statistically significant result, we ran each initialization method discussed in Section IV, 20 times and reported the best, mean, and worst results; as well as average function evaluations, CPU time and standard deviation. Function evaluation for each run was limited to 100,000 (the stopping criteria) to let the algorithm obtain the best parameter values. The RefSet size used was 36, which is the recommended size in this problem. With the high number of function evaluations and hundreds of parameters, the minimization process is expected to consume very lengthy CPU time. To surmount this drawback, Parallel Computing Toolbox in MATLAB has been used and it expedited the computation by assigning each run to eight different processors (logical cores) simultaneously. In this strategy, eight computations for each method was run independently using parfor loop which is available from the abovementioned toolbox. It should be noted that a single run takes approximately 11 hours, so 20 runs take approximately 220 hour. Using the parallel strategy, 20 runs only take approximately 33 hour, which reduced 72.6% of CPU time needed. All methods were experimented on i7 CPU with 16GB RAM which implemented in MATLAB 2015.

Table II shows that the best (minimum) cost function was obtained from QOBL method with $J = 210.0511$. The second best value is 229.1855, which was obtained from CR. Only RNG, OBL and TENT produced cost function values which were slightly higher than the published benchmark value, 233.90. The results revealed that QOBL is the best method in finding global minimum. However, although QOBL presented the minimum value, its average standard deviation was relatively higher than RNG, OBL and TENT. RNG is the most consistent method followed by OBL, having 4.5955 and 4.7331 standard deviation each, respectively. In terms of search effort, RNG produced the lowest average of function evaluations (1.2327e+05) and also its CPU time is also the lowest with 3.8191e+04 seconds. It should be observed that QOBL is the best initialization method if we consider its ability in minimizing the cost function in large-scale parameter estimation problem.

TABLE II. Experimental Results Obtained from the 20 Runs Conducted using different Initialization Methods

Initialization method	Best value	Worst value	Mean value	Standard deviation	Function evaluation	CPU time (s)
RNG	234.6651	252.0459	245.7053	4.5955	1.2327e+05	3.8191e+04
CR (rerun)	229.1855	270.0339	245.5427	10.0586	1.2421e+05	4.3142e+04
OBL	234.5223	250.9120	243.9961	4.7331	1.2546e+05	4.1942e+04
QOBL	210.0511	255.0070	241.4211	9.3596	1.2830e+05	4.2574e+04
TENT	234.2876	259.8841	246.2248	6.4994	1.2533e+05	4.0755e+04

Note: The best (minimum) value of cost function (weighted nonlinear least square) is shown in shaded cell. CR (rerun) indicates our own experimental result (in case when comparing with publish result in the next subsection).

Additional information to compare the different initialization methods is given in Fig. 1 and Fig. 2. The figures depicts the best curves (with minimum cost function *J*) among the 20 runs obtained from RNG, CR, OBL, QOBL and Tent methods. The curves show that all methods are able to minimize the cost function at a similar rate in terms of function evaluations and CPU time. Note that we have set the same initial guess as the initial value for all methods. This gives fairer comparison and assumes that the search space is feasible. In Fig. 1, starting from the first fractions of 1,000 function evaluations, QOBL has better speed and found acceptable cost function value when it reached approximately 25,000 function evaluations. Meanwhile, CR has the slowest speed until it reached around 40,000 evaluations. All methods continued to progress they reached the final evaluations. In this case, QOBL found the best value of 210.051 at 123,979 function evaluations. At the end of the evaluations, all solutions were able to achieve equivalent solutions in terms of quality, although, the best (minimum) value was obtained by QOBL while the worst (maximum) value obtained by RNG. In

Fig. 2, default initialization method based on CR has obtained slow convergene rate compared to others. It can be noticed where CR obtained acceptable value of cost function when CPU time reached nearly 8 hours, while QOBL reached the acceptable value at nearly 1.3 hours. The results revealed that RNG which is mostly used initialization method and CR as default initialization method in ESS have obtained poor result. Meanwhile, initialization method based on QOBL is a better alternative which was not only able to speed up convergence, but also obtain optimal solution.

To evaluate the quality of the parameter estimates, we compared the best result (QOBL) with a published benchmark result [20]. One thing to note is benchmark result used conventional CR as their initialization methods for the ESS. Table III shows that our study produced the best cost function, J = 210.05 compared to the benchmark's J = 233.90. However, our work obtained a bigger number of function evaluations compared to CR, with a difference of 33,251 evaluations. In terms of CPU time, the benchmark also produced shorter time of approximately 3 hours for a single run. Due to different stopping criteria used (the benchmark study uses CPU time) and different hardware specifications, comparing QOBL and CR in terms of CPU time seems unfair. It should be noted that in terms of efficiency in finding global minimum, method used in this study produced better results compared to CR.

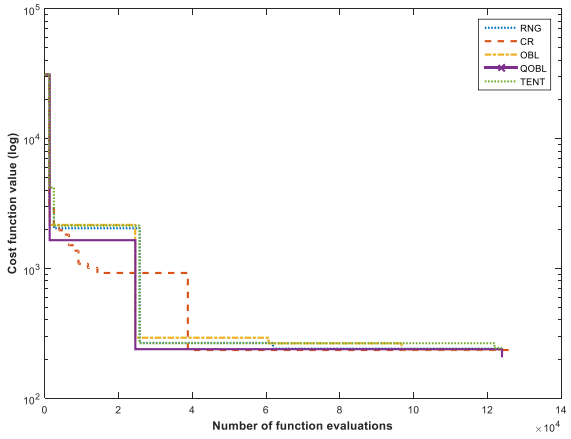

Fig. 1. Convergence of the Five Initialization Methods in Scatter Search.

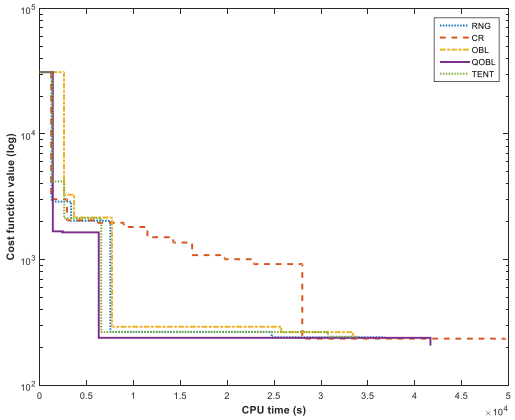

Fig. 2. Slow Convergence of the Five Initialization Methods in Scatter Search.

TABLE III. COMPARISON OF QOBL METHOD WITH PUBLISHED BENCHMARK IN SCATTER SEARCH

Initialization Methods	Best cost function J	Number of function evaluations	CPU Time (seconds)	ΣNRMSE
QOBL	210.05	12.3979e+04	4.1671e+04	2.3773
CR [20]	233.90	9.0728e+04	1.0800e+04	2.4921

Note: The best values are shown in shaded cell.

The decision variables (kinetic parameters) obtained from QOBL may provide the optimal model prediction since it produced the lowest cost function J and may produce best fit to experimental data. The goodness of fit can be measured by calculating root-mean-square-error ($RMSE$) for all metabolites. $RMSE$ is used to measure prediction error which is the different between experimental data and predicted model. The following equation defines $RMSE$,

$$RMSE = \sqrt{\frac{\sum_{exp=1}^{n_{exp}} \sum_{s=1}^{n_s^{exp,obs}} (ym_s^{exp,obs} - y_s^{exp,obs}(\boldsymbol{p}))^2}{n_{\epsilon xp} \cdot n_s^{exp,obs}}} \quad (20)$$

with the same notation defined in Eq. (1). In this case, normalized RMSE is used to cater for different magnitudes of observables. Each RMSE is divided by the range of value of observables as defined as,

$$NRMSE = \frac{RMSE}{\max(ym^{exp,obs}) - \min(ym^{exp,obs})} \quad (21)$$

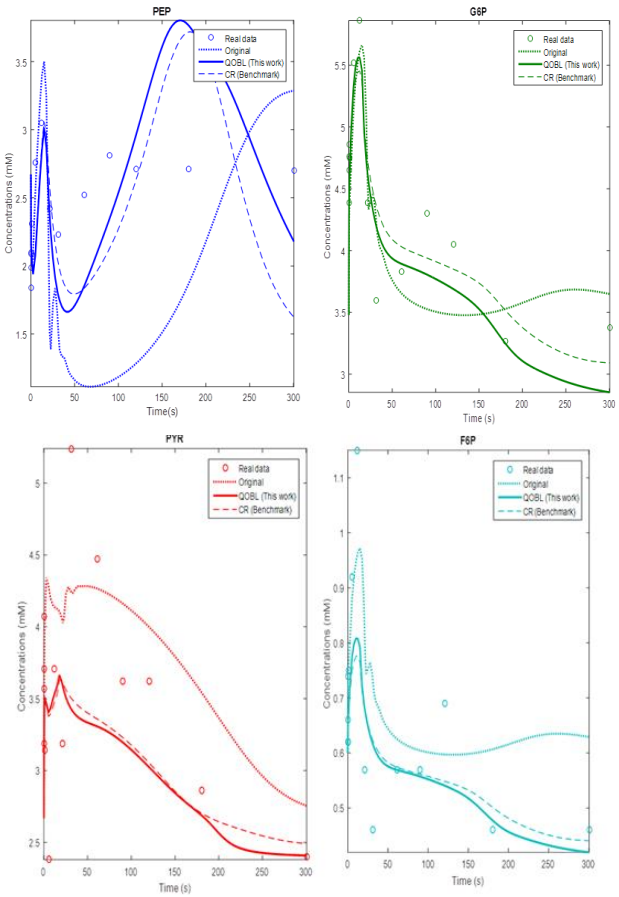

Fig. 3. Model Prediction over Experimental Data.

Thus, $\Sigma NRMSE$ is the sum of all $NRMSE$ for all observables. Table II shows our $\Sigma NRMSE$ is lower than the benchmark, means that it has better fit compared to the parameters obtained in the benchmark.

To measure the goodness of fit from parameter estimates with QOBL, we plot the model prediction over experimental data as shown in Fig. 3. For the sake of brevity, we plot only four out of nine metabolites. We chose metabolites which have very high nonlinear biological system, namely, PEP, G6P, PYR and F6P. The figure shows dynamic concentration change of extracellular glucose that responded to a pulse in central carbon metabolism. To compare the goodness of fit with other parameters, we also plot another fit based on parameters value from original published results and benchmark results. It should be observed that kinetic parameters retrieved from initialization methods based on QOBL represent a good fit between experimental data and predicted model.

VI. Conclusion

This paper studies different initialization methods and investigated their performance and effects on solving large-scale parameter estimation problem. We compared five initialization methods which are based on stochastic and randomization methods and implement them in ESS algorithm. Experimental results revealed that the choice of initialization (diversification generation) methods influenced the performance of the algorithm. The quality of solution, speed of convergence and statistical results were obtained with different characteristics derived from different initialization methods. Our statistical analyses revealed that the most popular initialization method, random number generator (RNG) performs poorly and there are significant better alternatives to this method, which have comparable computational requirements. In addition, the accuracy of model prediction also depends on the choice of initialization methods. Further investigation is needed to discover whether the same findings can be produced when different models and problems associated with parameter estimation are used. More intensive studies also need to be conducted on why some methods are generated more consistent performance in terms of statistical analysis and value of kinetic parameters in biological systems.

Acknowledgment

The authors would like to thank the Malaysian Ministry of Higher Education via the Fundamental Research Grant Scheme (FRGS), RACER/1/2019/ICT02/UMP//1 (University Reference RDU192601).

References

[1] Smallbone K, Mendes P. Large-Scale Metabolic Models: From Reconstruction to Differential Equations. Ind Biotechnol 2013;9:179–84. doi:10.1089/ind.2013.0003.

[2] Almquist J, Cvijovic M, Hatzimanikatis V, Nielsen J, Jirstrand M. Kinetic models in industrial biotechnology - Improving cell factory performance. Metab Eng 2014;24:38–60. doi:10.1016/j.ymben.2014.03.007.

[3] Hussain F, Jha SK, Jha S, Langmead CJ. Parameter discovery in stochastic biological models using simulated annealing and statistical model checking. Int J Bioinform Res Appl 2014;10:519–39. doi:10.1504/IJBRA.2014.062998.

[4] Link H, Christodoulou D, Sauer U. Advancing metabolic models with kinetic information. Curr Opin Biotechnol 2014;29:8–14. doi:10.1016/j.copbio.2014.01.015.

[5] Mendes P, Kell D. Non-linear optimization of biochemical pathways: applications to metabolic engineering and parameter estimation. Bioinformatics 1998;14:869–83. doi:10.1093/bioinformatics/14.10.869.

[6] Moles CG, Mendes P, Banga JR. Parameter estimation in biochemical pathways: a comparison of global optimization methods. Genome Res 2003;13:2467–74. doi:10.1101/gr.1262503.

[7] Egea J a., Martí R, Banga JR. An evolutionary method for complex-process optimization. Comput Oper Res 2010;37:315–24. doi:10.1016/j.cor.2009.05.003.

[8] Mansour N, Kehyayan C, Khachfe H. Scatter search algorithm for protein structure prediction. Int J Bioinform Res Appl 2009;5:501–15. doi:10.1504/IJBRA.2009.028679.

[9] Kazimipour B, Li X, Qin AK. Initialization methods for large scale global optimization. 2013 IEEE Congr. Evol. Comput., 2013, p. 2750–7. doi:10.1109/CEC.2013.6557902.

[10] Egea J, Balsa-Canto E. Dynamic optimization of nonlinear processes with an enhanced scatter search method. Ind Eng Chem Res 2009;48:4388–401.

[11] Egea JA, Henriques D, Cokelaer T, Villaverde AF, MacNamara A, Danciu D-P, et al. MEIGO: an open-source software suite based on metaheuristics for global optimization in systems biology and bioinformatics. BMC Bioinformatics 2014;15:136. doi:10.1186/1471-2105-15-136.

[12] Storn R, Price K. Differential Evolution -- A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. J Glob Optim 1997;11:341–59. doi:10.1023/A:1008202821328.

[13] Chong C, Mohamad M, Deris S, Shamsir M, Chai L, Choon Y. Parameter Estimation by Using an Improved Bee Memory Differential Evolution Algorithm (IBMDE) to Simulate Biochemical Pathways. Curr Bioinform 2014;9:65–75. doi:10.2174/15748936113080990007.

[14] Rodriguez-Fernandez M, Egea JA, Banga JR. Novel metaheuristic for parameter estimation in nonlinear dynamic biological systems. BMC Bioinformatics 2006;7:483. doi:10.1186/1471-2105-7-483.

[15] Tizhoosh HR. Opposition-Based Learning: A New Scheme for Machine Intelligence. Comput Intell Model Control Autom 2005 Int Conf Intell Agents, Web Technol Internet Commer Int Conf 2005;1:695–701. doi:10.1109/CIMCA.2005.1631345.

[16] Rahnamayan S, Tizhoosh HR, Salama MM. Opposition-based differential evolution. Stud Comput Intell 2008;143:155–71. doi:10.1007/978-3-540-68830-3_6.

[17] Rahnamayan S, Tizhoosh HR, Salama MMA. Quasi-oppositional differential evolution. 2007 IEEE Congr. Evol. Comput. CEC 2007, 2007, p. 2229–36. doi:10.1109/CEC.2007.4424748.

[18] Saremi S, Mirjalili S, Lewis A. Biogeography-based optimisation with chaos. Neural Comput Appl 2014;25:1077–97. doi:10.1007/s00521-014-1597-x.

[19] Chassagnole C, Noisommit-Rizzi N, Schmid JW, Mauch K, Reuss M. Dynamic modeling of the central carbon metabolism of Escherichia coli. Biotechnol Bioeng 2002;79:53–73. doi:10.1002/bit.10288.

[20] Villaverde AF, Henriques D, Smallbone K, Bongard S, Schmid J, Cicin-Sain D, et al. BioPreDyn-bench: a suite of benchmark problems for dynamic modelling in systems biology. BMC Syst Biol 2015;9:1–15. doi:10.1186/s12918-015-0144-4.