Expert Systems With Applications 116 (2019) 131–146

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Cooperative enhanced scatter search with opposition-based learning schemes for parameter estimation in high dimensional kinetic models of biological systems

Muhammad Akmal Remli[a], Mohd Saberi Mohamad[b,c,*], Safaai Deris[b,c], Azurah A Samah[d], Sigeru Omatu[e], Juan Manuel Corchado[f]

[a] Faculty of Computer Systems & Software Engineering, Universiti Malaysia Pahang, Kuantan, Pahang 26300, Malaysia
[b] Institute For Artificial Intelligence and Big Data, Universiti Malaysia Kelantan, City Campus, Pengkalan Chepa, Kota Bharu, Kelantan 16100, Malaysia
[c] Faculty of Bioengineering and Technology, Universiti Malaysia Kelantan, Jeli Campus, Lock Bag 100, Jeli, Kelantan 17600, Malaysia
[d] Artificial Intelligence and Bioinformatics Research Group, School of Computing, Universiti Teknologi Malaysia, Skudai, Johor 81310, Malaysia
[e] Department of Electronics, Information and Communication Engineering, Osaka Institute of Technology, Osaka 535-8585, Japan
[f] Biomedical Research Institute of Salamanca/BISITE Research Group, University of Salamanca, Calle Espejo s/n, Salamanca 37008, Spain

ARTICLE INFO

Article history:
Received 17 January 2018
Revised 8 September 2018
Accepted 8 September 2018
Available online 11 September 2018

Keywords:
Parameter estimation
Metabolic engineering
Kinetic model
Opposition-based learning
Global optimization
Cooperative metaheuristic

ABSTRACT

Industrial bioprocesses development nowadays is concerned with producing chemicals using yeast, bacteria and therapeutic proteins in mammalian cells. This involves the utilization of microorganism cells as factories and re-engineering them *in silico*. The tools that could facilitate this process are known as the kinetic models. Kinetic models of cellular metabolism are important in assisting researchers to understand the rational design of biological systems, predicting metabolites production, and improving bio-products development. However, the most challenging task in model development is parameter estimation, which is the process of identifying an unknown value of model parameters which provides the best fit between the model output and a set of experimental data. Due to the increased complexity and high dimensionality of the models, which are extremely nonlinear and contain large numbers of kinetic parameters, parameter estimation is known to be difficult and time-consuming. This study proposes a cooperative enhanced scatter search with opposition-based learning schemes (CeSSOL) for parameter estimation in large-scale biology models. The method was executed in parallel with the proposed cooperative mechanism in order to exchange information (kinetic parameters) between individual threads. Each thread consists of different parameters settings that enhance the systemic properties in obtaining the global minimum. The performance of the proposed method was assessed against two large-scale microorganisms models using mammalian and bacteria cells. The results revealed that the proposed method recorded faster computation time compared to other methods. The study has also demonstrated that the proposed method can be used to provide more accurate and faster estimation of kinetic models, indicating the potential benefits of utilizing this method for expert systems of industrial biotechnology.

1. Introduction

Pharmaceutical products and dairy foods are mostly produced during the fermentation process in a fed-batch culture (Park, Kim, Lee, & Lee, 2011). This process is a part of metabolic engineering, which is a practice in industrial biotechnology (Robles-Rodriguez et al., 2017). In a bioreactor, fed-batch culture converts substrates (i.e. glucose) to desired metabolites, notably lactic acid (Oh et al., 2011; Zain, bin, Kanesan, Kendall, & Chuah, 2018). The host cell plays an important role in this process, which acts as a factory to produce the desired metabolites (products) under the controlled condition. Bacterium cells with high growth rate such as *Escherichia coli* (*E. coli*) and *Saccharomyces cerevisiae* (*S. cerevisiae*) are widely used as cell factories in scientific research as well as in producing food and dairy (Dobson et al., 2010; Park et al., 2011). Moreover, mammalian cells such as Chinese hamster ovary (CHO) has been used to manufacture therapeutic proteins such as insulin for diabetes treatment (Ahn & Antoniewicz, 2012).

* Corresponding author at: Institute for Artificial Intelligence and Big Data, Universiti Malaysia Kelantan, City Campus, Pengkalan Chepa, Kota Bharu, Kelantan 16100, Malaysia.
E-mail addresses: akmalremli@ump.edu.my (M.A. Remli), saberi@umk.edu.my (M.S. Mohamad), safaai@umk.edu.my (S. Deris), azurah@utm.my (A. A Samah), omatu@rsh.oit.ac.jp (S. Omatu), corchado@usal.es (J.M. Corchado).

https://doi.org/10.1016/j.eswa.2018.09.020
0957-4174/

Nowadays, computational modeling in systems biology plays an important role in assisting metabolic engineering through *in silico* approach. The biological processes of a cell could be formulated via a mathematical model using ordinary differential equations (ODEs) to mimic the behavior of a real cell. Thus, by utilizing and simulating the model, biologists can understand, predict and improve the products in order to meet the industrial demands (Cvijovic, Bordel, & Nielsen, 2011). One type of model which is capable of executing this task is the kinetic model (Billa et al., 2017; Saa & Nielsen, 2016). Mechanistic and dynamic details of a kinetic model signify various beneficial information such as biochemical reactions, cell compartments, and rate of metabolites concentration (Miskovic, Tokic, Fengos, & Hatzimanikatis, 2015). The development of kinetic models provides a novel paradigm in the rational design of cell factory which could be beneficial in various biotechnology applications (Jahan, Maeda, Matsuoka, Sugimoto, & Kurata, 2016).

Recently, research efforts have been focused on building large-scale kinetic models. These large models could be employed to predict near-optimal alterations required to improve desired products and simultaneously maintain other functions in the host in a minimal but essential level. This provides benefits in terms of rational design in cell factories. For instance, current kinetic modeling practice involves scaling-up a specific metabolic process such as glycolysis from pathway level to genome level. With such effort, the interaction of biochemical reactions in this process could be investigated at the genome level. This proves an opportunity for biologists to clone the glycolysis pathway to a new host and observe its effects (Gábor, Villaverde, & Banga, 2017; Smallbone & Mendes, 2013). In the near future, the ultimate goal of large-scale kinetic models is to create whole-cell models which provide dramatic impact in the field of systems biology and metabolic engineering (Karr, Takahashi, & Funahashi, 2015; Macklin, Ruggero, & Covert, 2014).

However, one of the most daunting tasks in the large-scale model building is parameter estimation, commonly known as system identification or model calibration (Gottu Mukkula & Paulen, 2017). This task consists of applying global optimization methods in order to explore unknown values of model parameters that provide the best fit between model prediction and a set of experimental data. Kinetic models with hundreds to thousands of kinetic parameters cause a serious challenge in parameter estimation due to the high dimension of search space that is required to be explored. Thus, in high dimensional kinetic models (such models may contain hundreds to thousands of parameters), the performance of most optimization methods, especially those using sequential metaheuristics, tend to deteriorate and is deemed expensive in terms of computational cost. Therefore, in order to tackle this issue (especially to reduce the CPU time), parallelization using cooperative search could be performed (Li & Wang, 2014; Penas, Gonzalez, & Egea, 2017; Villaverde, Egea, & Banga, 2012).

The cooperative search strategy is one class of metaheuristic optimization technique that aims to speed up computational times, convergence rate, and produce a robust algorithm (Alba, 2005; Crainic & Toulouse, 2010; El-Abd & Kamel, 2005; Nedjah et al., 2016; Ngo, Sadollah, & Kim, 2016). The main benefit of this technique is that it reduces the computational effort when dealing with large-scale data (Martin, Ouelhadj, Smet, Vanden Berghe, & Ozcan, 2013). The idea behind this strategy is that algorithms will exchange and share information between search agents (global and local searches). Algorithms with different settings will run in parallel and the results will be gathered and exchanged in order to produce a better solution. Cooperative search is one of the techniques in parallel metaheuristic methods (Crainic & Gendreau, 2002; Crainic, Gendreau, Hansen, & Mladenović, 2004; Cruz & Pelta, 2009; El-Abd & Kamel, 2005; Talbi & Bachelet, 2006).

One of the prominent methods that have been proposed when dealing with large-scale models is cooperative enhanced scatter search (CeSS) proposed by Villaverde et al. (2012). CeSS is the parallel version of enhanced scatter search (eSS) (Egea et al., 2014; Egea, Martí, & Banga, 2010). The method benefits from multiple threads of eSS that could run in parallel. This method exchanges information between threads in fix time intervals. Each thread consists of different parameter settings that lead to different performance and search pattern among them. Thus, it acts like a macroscopic behavior due to the interaction that occurs between individual threads, while this strategy influences the systemic properties in each eSS thread. The method is able to reduce computation time and is efficient in searching different regions of the search space. However, the proposed method faces serious issues in terms of efficiency and scalability when dealing with the large-scale model.

Differential evolution (DE) is widely used and studied in parameter estimation of biological systems (Chong et al., 2014; Zúñiga, López Cruz, & García, 2014). However, this method results in high computation time due to the high dimensional dataset used. One variant of the parallel method using DE was proposed to overcome excessive computational cost on large datasets (Penas, Banga, Gonzalez, & Doallo, 2015). The method takes advantages of asynchronous parallel implementation and hybrid global and local search. Moreover, three heuristics local searches were also used for cooperation toward global minimum including local solver, tabu search, and logarithmic search In order to test this method, the experiment was performed using cluster CPU employing 16 nodes multicores processor. The result indicated that the method could significantly reduce the computation effort required. The main weaknesses of the method are that it is time consuming and challenging to tune the control parameters when it involves a large number of processors and multiple local searches.

Recently, a variant of the cooperative search method known as the self-adaptive cooperative enhanced scatter search (saCeSS) was proposed which extends CeSS (Penas et al., 2017). It provides several novel mechanisms in terms of asynchronous cooperative and self-tuning strategies. This promising method employs parallel implementation using coarse-grained and fine-grained parallelization. The method could be performed on High Performance Computing (HPC) systems including clusters of multicore nodes based on Message Passing Interface (MPI). The strength of this method is that it is able to significantly reduce the computational time in comparison to previous methods (from days to minutes, as reported in several cases). However, this method requires expensive computational resources to achieve significant result via the use of multicore cluster with a large number of nodes. This factor limits the usage of this method especially in solving large-scale kinetic models and whole-cell models.

In view of expert systems, cooperative search metaheuristic serves as an intelligence process to search near-optimal values of kinetic parameters in the model. The process consists of generating initial random solutions as inputs which were iteratively improved using cooperative mechanism and parallel implementation of global optimization in order to produce the outputs (near-optimal values of kinetic parameters). The parameter value obtained in that process is plugged into the ODE model for the simulation purpose, which is the ultimate goal of modeling. By means of an established (accurate) model, the simulation can be prepared by fitting the model with experimental data. From this process, the model can be seen as a cell factory tool that can answer various biological hypothesis and as an alternative way to conduct a wet laboratory experiment. The model can be utilized to predict, evaluate, and explore different scenarios of biological processes. For instance, the level of metabolite concentration (in millimolar) can be predicted based on fermentation process duration without the necessity of execution of wet lab experiment. This allows biolo-

gists to design multiple experiments, conduct significant modifications and improve production, quality, and bioproduct process design (Almquist, Cvijovic, Hatzimanikatis, Nielsen, & Jirstrand, 2014).

In this study, a cooperative enhanced scatter search with opposition-based learning schemes (CeSSOL) has been proposed for parameter estimation in high dimensional kinetic models of biological systems. The proposed method is capable of reducing the computation time as well as providing a robust method in terms of solution quality (minimum value of the objective function). It is expected that the performance of the proposed method is better in dealing with high dimension parameter estimation problems when the low-cost computational resource is used (multicores on a single workstation). Specifically, the main contributions of this study are:

(1) In the proposed CeSSOL, a recent variant of eSS, known as SSCOL (scatter search with combined opposition-based learning) (Remli, Deris, Mohamad, Omatu, & Corchado, 2017) has been utilized as the individual thread which is executed in parallel using a small number of processors. SSCOL has been successfully applied in parameter estimation and large-scale global optimization problems. In SSCOL, different types of opposition-based learning schemes have been proposed at different stages of search processes. For instance, in diversification generation method, quasi-opposition based learning has been introduced for enhanced initialization. A similar scheme was applied in solution combination method of SSCOL in order to enhance the reduced parameter space for rapid convergence. In addition, quasi reflection intensification was introduced in order to improve the intensification phase. In previous work, SSCOL was proposed and implemented in a serial run and expensive computation time. In contrast, the proposed method is implemented in a parallel manner using similar CPU resources as in SSCOL case which resulted in faster estimation to be potentially utilized in industrial biotechnology applications.
(2) In this study, an information exchange strategy between individual SSCOL thread based on good and diverse solutions was proposed that is capable of efficiently searching broad region of solution space. In previous work (Villaverde et al., 2012), the kinetic parameters values to be shared among threads were based on the best solution found and the rest of the solutions (*RefSet*). To the best of our knowledge, this has limited the diversity of the search region and tends to stuck in the local minima. In this study, two solutions found (good and diverse) were utilized among all threads to be applied as initial solutions along with new initial random values generated in the initialization stage. These solutions were then improved by SSCOL. The proposed strategy is expected to search a wider area of fitness landscape that can provide a more accurate estimation. Finally, the proposed CeSSOL method outperforms the state-of-the-art 8-SSCOL (a parallel version of SSCOL method) on both solution quality and computation time.

This paper was organized as follows. Section 2 formulates the parameter estimation problem, which consists of minimizing nonlinear least squares function with several constraints. Section 3 describes the recent method, SSCOL, which was proposed to solve parameter estimation problems. Section 4 introduces the proposed method in this paper, stated as cooperative enhanced scatter search with opposition-based learning (CeSSOL). Section 5 demonstrates the experimental setup including dataset used and other methods for comparison. Section 6 discusses the result obtained from the computational experiment and comparison made with previous works. Finally, Section 7 concludes the findings and future works of the research.

2. Problem formulation

Parameter estimation in metabolic engineering application consists of the process to identify the unknown value of parameters in kinetic models of biological systems. The parameters determine the accuracy of model prediction where they provide the best fit to the experimental data. This problem is also formulated as a global optimization problem. Optimization methods can be utilized in order to minimize the distance between predicted model and experiment data, which is formulated as nonlinear least squares (NLS) function (Moles, Mendes, & Banga, 2003; Villaverde et al., 2015):

$$J = \sum_{o=1}^{n_o} \frac{(ym_o - ym_o(p))^2}{(\sigma^o)^2} \tag{1}$$

where J is the objective function, n_o is the number of measured states (metabolite products), ym is the measured states, p is the kinetic parameters for estimation, and σ is the weight to balance the contribution of a different order of magnitude in metabolite products.

Minimization process of the NLS is subject to the following constraints:

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

$$x(t_0) = x_0 \tag{3}$$

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

$$p^L \leq p \leq p^U \tag{5}$$

where x is the state variable and f is the function describing systems dynamics in the nonlinear biochemical process model. The initial condition (concentrations) of x at time zero t_0 was denoted as x_0 while g is an observation function and p is the kinetic parameters in the range of lower bound p^L and upper bound p^U. Biological processes that occur in microorganism cells such as metabolism and enzymatic reactions are highly nonlinear. ODEs models which represent these processes are known to be complex with many interdependencies involved between dynamic state variables (metabolites) and higher order interrelationship among kinetic parameters (Li & Vu, 2013). Highly nonlinear systems in ODEs cause multimodality in optimization landscape where local minima exist (Moles et al., 2003; Villaverde et al., 2012). This phenomenon causes most optimization methods to easily converge in local minima and reduces the predictive ability of the model. Thus, a highly nonlinear model poses a critical challenge in obtaining near-optimal model parameter values.

3. Enhanced scatter search with combined opposition-based learning (SSCOL)

SSCOL is a recent variant of scatter search (eSS) (Egea et al., 2010; Riahi, Khorramizadeh, Hakim Newton, & Sattar, 2017) that was proposed in order to solve large-scale and challenging parameter estimation problems (Remli, Mohamad, Deris, Napis, Sinnott, & Sjaugi, 2017). It is the extension of eSS (Egea & Balsa-Canto, 2009; Egea et al., 2010) which has been improved in terms of a combination of opposition-based learning schemes. The schemes are introduced in several parts of eSS which made it efficient, particularly for improving the speed of convergence, when dealing with high dimensional problems. This method performs extensive exploration and intensification in the search space. Contrasting to the original eSS, SSCOL employs quasi-opposition based learning scheme in the *RefSet* formation. Since the *RefSet* size is small, only half of the original *Refset* is subject to evaluation in order to generate a

Table 1
Algorithm 1. Scatter Search with Combined Opposition-based Learning Schemes (SSCOL).

Input : Experimental data, model parameter values (randomly generated within bounds)
Output : Near-optimal model parameter values
1: Generate initial *RefSet* with high quality and quasi-opposite members
2: **repeat**
3: Sort *RefSet* by quality $[x^1, x^2, \ldots, x^{dim\_refset}]$ so that
$f(x^i) \leq f(x^j)$ where $i, j \in [1, 2, \ldots, dim\_refset]$ and $i < j$
4: **if** $max(abs(\frac{x^i - x^j}{x^j})) \leq \epsilon$ with $i < j$ **then**
5: Replace x^j by a random solution
6: **end if**
7: **for** $i = 1$ *to* dim_refset **do**
8: Combine x^i with the rest of population members to generate a set of *dim_refset* new members, offspring^i using quasi-reflection combination
9: $x^i_{off} =$ best solutions in **offspring**^{i}
10: **if** x^i_{off} outperforms x^i **then**
11: Apply quasi-reflection intensification and perform quasi-reflection with probability
12: **end if**
13: **end for**
14: Update best solution found x_{best} and its objective function value f_{best}
15: Perform a local search from x_{best} to obtain x^* based on competitive ranking initial selection and balance between quality and diversity
16: **if** $f(x^*) < f(x_{best})$
17: Update x_{best}, f_{best}
18: **end if**
19: **until** stopping criterion is met

Note: The introduced opposition-based learning schemes are shown in lines 1, 8 and 11.

quasi-opposite solution. Hence, the method is able to provide high-quality *RefSet* with fewer function evaluations. Moreover, the original combination method in eSS has been modified to efficiently accelerate the search process in high dimensional search space. The modification employs quasi-reflection combination to quickly reach a high-quality solution, which leads to faster convergence. Another additional feature in SSCOL is the introduction of jumping rate j_r to perform intensification using quasi-reflection. The feature improved the intensification phase by generating a quasi-reflection solution to accelerate search process. These three modifications, which were based on a combination of opposition-based learning schemes, significantly improved the performance of original eSS in terms of convergence speed and solution quality. Further details regarding this optimization method can be found in Remli et al. (2017). Table 1 depicts the SSCOL algorithm.

SSCOL is considered as a better metaheuristic method compared to the original eSS. However, when dealing with a large number of parameters the computational cost is still excessive due to the high dimensional (hundreds of kinetic parameters) search space that is required to be explored.

4. Cooperative enhanced scatter search with opposition-based learning (the proposed method)

In this study, an improved cooperative metaheuristic method known as CeSSOL (cooperative enhanced scatter search with opposition-based learning) is proposed. This method shares the idea of cooperative enhanced scatter search (CeSS) (Villaverde et al., 2012) but is different in terms of three fundamental modifications. Firstly, this method employs SSCOL for parallel threads, which has been proven to solve parameter estimation problems. Secondly, the information exchange strategy in this method utilizes good and diverse solutions as well as newly generated random solutions in order to form a *RefSet* for each thread. The strategy is not restricted to all members of *RefSet* for information exchange which is driven by the best solution found, and the *RefSet* that contains information regarding the diversity of solutions. Thirdly, this method employs a number of function evaluations for stopping condition in each thread instead of CPU time. The main strategy in CeSSOL is to execute an optimization process via several threads η in parallel execution. Each thread implements an optimization method using SSCOL. During the optimization process, information (solutions or kinetic parameters) in *RefSet* ref_j are exchanged between threads for every cooperative iteration co_{iter}. Thus, those *RefSet* which has high quality solutions will be used in order to further guide the search process and thus enhancing the systemic properties in every SSCOL. It should be noted that the cooperative search method does not solely speed up the computational time, but it is expected that it could produce a robust result in solving challenging optimization problems. Fig. 1 and Table 2 demonstrate the flowchart and pseudocode of the proposed CeSSOL, respectively. The final output of this method is a near-optimal set of kinetic parameters which can be found in the supplementary material.

4.1. Schematic representation of CeSSOL

To reap full benefit from the implementation of a cooperative search method, multicores processor is required. In order to run multiple parallel SSCOL threads in the MATLAB environment, *jPar* library which consists of three components was adopted (Karbowski, Majchrowski, Trojanek, Pokorski, & Załuga, 2015):

1) Registration server
2) Solvers (Slave threads)
3) Client (Master thread)

The first process involves starting the *registration server* using a single Java executable *JAR* file through the command line. The registration server is responsible for managing a set of solvers (threads). Then, several MATLAB instances were launched. One of the instances was used for the *jPar* console (as Master thread or central processor) and the rest were used for *solvers* (Slave threads). The Slave threads were started in the same directory and once they have been started, they will wait for input to be processed. Each slave thread blocks the current MATLAB session until a new task is available, while the Master thread acts as a client or central processor which manages all tasks including dividing the data into chunks and distributing parameter setting for all threads.

Fig. 2 depicts the schematic representation of CeSSOL. Every thread has one MATLAB session running SSCOL method while thread *0* acts as a central processor. Information exchange strategy between cooperative iteration enables each thread to benefit from

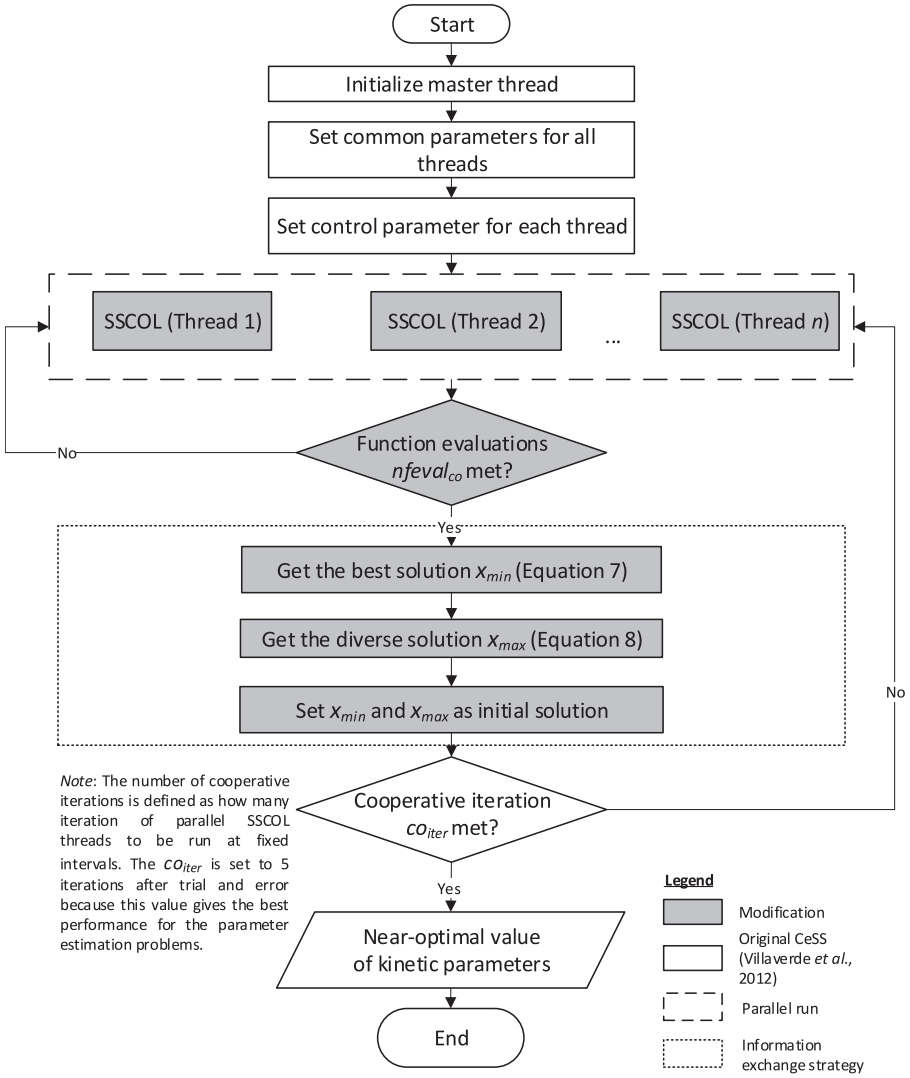

Fig. 1. Flowchart of the proposed CeSSOL method.

Table 2
Algorithm 2. Cooperative Enhanced Scatter Search with Opposition-based Learning (CeSSOL).

Input : Experimental data and randomly generated kinetic parameters values
Output : Near-optimal values of kinetic parameters

```
Initialization of Master (Processor 0) and Slave (Processor j) threads
Set parameters that are common for all threads
Set control parameters for each thread
Initialize global reference set (RefSet) array: Global_ref = [ ]
for i = 1 to co_iter do
    for j = 1 to η do
        Slave thread j (parallel processing): run optimization using SSCOL (Algorithm 1)
    (Algorithm 1)
        if nfeval_co then
            if ref_j ∉ Global_ref then
                Global_ref = [Global_ref, ref_j]
            end if
            Assign good and diverse solutions as initial solutions for next co_iter in all threads (Algorithm 3)
        in all threads (Algorithm 3)
        end if
    end for
end for
Final solution = best solution in Global_ref
```

Note: $nfeval_{co}$ is the number of function evaluations for all threads. This stopping condition is for all threads based on the predefined value from Eq. (6).
ref_j is the *RefSet* that contains unique solutions from each thread *j*.
co_{iter} is the number of cooperative iteration and this value can be set based on trial and error. Starting from cooperative iteration 2, all threads receive two solutions (best and diverse) as initial solutions while the rest of the solutions is completed by using a random number. From that, the new *RefSet* with the fixed size is created.

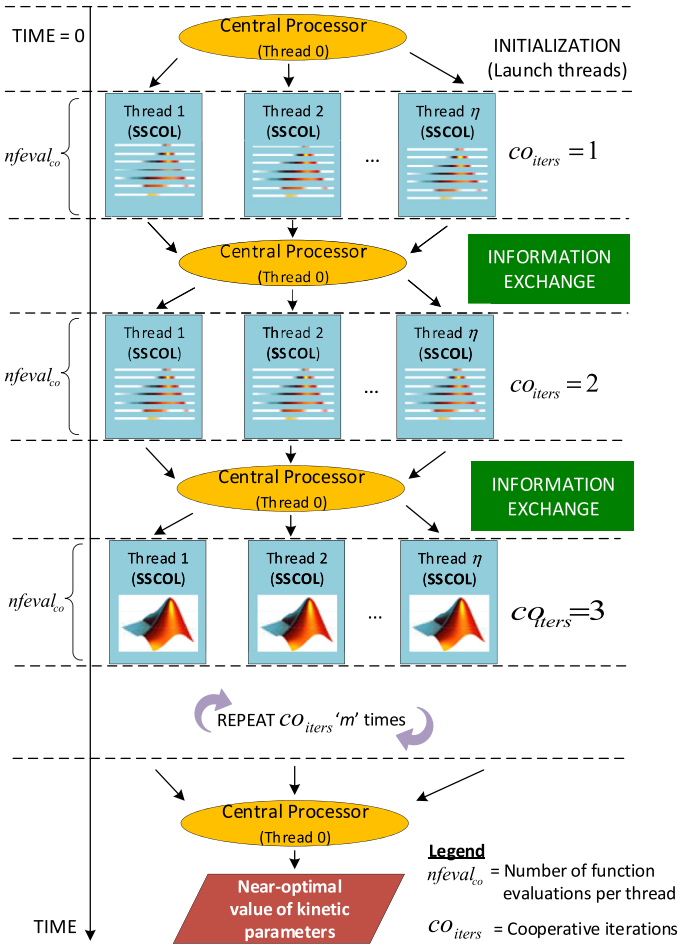

Fig. 2. Schematic representation of CeSSOL (adopted from Villaverde et al., 2012). The number of cooperative iteration co_{iter} is defined by the number of iteration of all the SSCOL threads to be run at fix intervals.

the solutions gathered by other threads. It should be noted that every SSCOL thread has a different parameter setting to vary the search behavior among them. Thus, the proposed method changes the systemic properties for all SSCOL threads in order to find global minimum, especially for challenging optimization and parameter estimation problems. Additional details on parameter setting in all SSCOL threads will be discussed in the next subsection.

4.2. Common parameters for all threads

All SSCOL threads share common parameters that are number of diverse solutions (*ndiverse*), *RefSet* size, number of function evaluations, jumping rate, and local search. The *ndiverse* for each SSCOL thread is set to the recommended size as described in the previous work (Remli et al., 2017). The size that is ten times bigger than the size of kinetic parameters is employed in order to ensure that the initial solutions are sampled in the broad area thus increasing the chance of obtaining global minimum. In order to reduce the time required for evaluating objective function, the *RefSet* size must be set to a lower number. In this method, the *RefSet* size was set manually by trial and error method after the initial exploratory run was performed.

In terms of function evaluations, every thread has the same number of function evaluation, which was used as the stopping condition for each SSCOL thread. Thus, this study defines a new tunable parameter of function evaluations for each cooperative thread $nfeval_{co}$ as:

$$nfeval_{co} = \frac{nfeval_{se}}{co_{iter}} \tag{6}$$

where $nfeval_{se}$ is the maximum number of function evaluations used for a sequential run of SSCOL, and co_{iter} is the number of cooperative iteration. It should be noted that previous works adopted the CPU time of each thread as the time between information exchanges. However, due to different hardware specifications, a number of function evaluations will be able to provide an unbiased comparison between different methods. Table 3 depicts a common parameter setting for all SSCOL threads.

In Table 3, *ndiverse*, jumping rate J_r, and local search are set with the default setting as in previous work (Remli et al., 2017). Meanwhile, the value for *RefSet* size is much lower ($dim\_refset = 10$) than in the previous work ($dim\_refset = 36$). This value was obtained after trial and error was performed.

Table 3
Parameter setting for all common SSCOL threads.

Parameters	Values
Number of diverse solutions (*ndiverse*)	1,170 for CHO and 1,160 for CCM
RefSet size (*dim_refset*)	10
Number of function evaluations for every thread per iteration	24,000 for CHO and 18,000 for CCM
Jumping rate J_r	0.3
Local search	*fmincon*

Note: Number of function evaluations for every thread per iteration is set according to Eq. (6). CHO is the Chinese hamster ovary cell, while CCM is central carbon metabolism. Both datasets were used to test and evaluate the proposed method. More description of this data can be observed in the experimental setup section.

Table 4
Control parameters for each thread.

Parameter	Description
local.n2	Number of iterations between two local searches
balance	Balance between diversification and intensification of initial points for local search

4.3. Control parameters for each thread

Instead of setting the same control parameters for all threads, each thread could be set with specific parameters which would result in different performance and search characteristics. This different parameter setting could alter the search processes due to the random nature of SSCOL method. In this study, the following parameter could be set with a different value for each *balance* and *local.n2* thread. Table 4 depicts the control parameters and their descriptions for each thread in CeSSOL.

Table 5 demonstrates the value of parameter settings for each thread. These parameters were set after several initial runs (trial and error) have been performed. The parameter settings for SSCOL influence its performance in solving optimization and parameter estimation problems. Since the fitness landscape - either rugged or smooth surface - of the problems are usually unknown (except for large-scale global optimization benchmark functions), it is essential to possess various settings for every thread. 'Aggressive' threads have a small value of *balance* and *local.n2* parameters, which are suitable for a smooth surface. They focus on intensification by launching local search frequently. 'Conservative' threads, on the other hand, focus on searching broad area of search space and spending more time on solution combination. In addition, 'Conservative' threads have a large value of both parameters and are suit-

Table 5
Parameter settings for each thread (thread 1–8).

Thread	Balance	Local.n2
Thread 1 (SSCOL 1)	0	1
Thread 2 (SSCOL 2)	0	4
Thread 3 (SSCOL 3)	0	8
Thread 4 (SSCOL 4)	0	10
Thread 5 (SSCOL 5)	0.25	20
Thread 6 (SSCOL 6)	0.25	30
Thread 7 (SSCOL 7)	0.25	50
Thread 8 (SSCOL 8)	0.5	7

Note: These parameters are set after initial experiments have been performed. These values gave the best performance for each thread.

Table 6
Algorithm 3 Pseudo-code of the proposed information exchange strategy based on good and diverse solutions.

Input : All *RefSet* members for every thread
Output : Two *RefSet* members (good and diverse solutions)
1: Merge all *RefSet* obtained from every thread
2: Get the good solution in *RefSet* (x_{min}) using **Eq. (7)**
3: **for** all *RefSet* members **do**
4: Compute sum difference between good solution (x_{min}) and all *RefSet* (x_i) using **Eq. (9)**
5: **end for**
6: Get maximum difference of solution x_{max} using **Eq. (8)**

able for the rugged surface. The combination of 'aggressive' and 'conservative' threads is known to be beneficial for solving difficult and challenging parameter estimation problems.

4.4. Information exchange strategy

Information exchange plays an important role in cooperative search. Parallel run of several SSCOL threads produces the output (*RefSet*) based on fixed number of function evaluations and these *RefSet* were employed as information to be exchanged with other threads. In this work, the information exchange is based on good and diverse solutions along with random initial solutions. Two solutions from merged *RefSet* were used as the initial solution for the next cooperative iteration. The first element is the good solution value, which in this study, is the lowest value (x_{min}), that is defined as:

$$x_{min} = min(RefSet) \tag{7}$$

where *RefSet* contains a set of candidate solutions obtained from several SSCOL threads. The second element is a diverse solution value (x_{max}), which was selected based on the largest distance between a good solution and the rest of candidate solutions, defined as:

$$x_{max} = max(d) \tag{8}$$

where

$$d = abs \sum_{i=1}^{n} x_{min} - x_i \tag{9}$$

abs is absolute value and x_i is the remaining candidate solutions in *RefSet*. Thus, good and diverse solutions were used to guide all SSCOL threads for the remaining cooperative iterations. The algorithm for information exchange proposed in this research has been demonstrated in Table 6.

In Algorithm 3, information exchange occurs when all threads have reached the number of function evaluations in each cooperative iteration. Once all threads have completed the tasks, all *RefSet* members from parallel threads are merged. Two solutions, which is the best solution x_{min} and diverse solution x_{min} from the merged *RefSet* is then used as the initial solutions for the next cooperative iteration. From here, every SSCOL will use good and diverse solutions as well as the random number as initial solutions before generating a *RefSet*. For instance, if the size of the initial solutions is 100, the size will be then increased to 102. Next, the quality of initial solutions is sorted before forming a small *RefSet* (e.g. 10). The *RefSet* consists of high quality solutions which were selected from the initial solutions. Thus, there is no increase in the size of the *RefSet* for every thread. Recent work (Villaverde et al., 2012) employed all *RefSet* members that were obtained from parallel threads as initial solutions. In addition, their information exchange strategy was also based on best solutions found and the rest of the *RefSet*. However, their method prevents the initial random number to be generated as initial solutions for the rest of the iteration. The main drawback of their strategy is limited randomness, which prevents the exploration of other diverse solutions. This study employs only two elements to be exchanged and combines them with random values for each cooperative iteration. This strategy is expected to increase the diversity when combined with opposition-based learning schemes which is introduced in SSCOL.

Table 7
Number of threads and cooperative iterations in CeSSOL.

Parameters	Values
Number of threads η	8
Number of cooperative iteration co_{iter}	5

4.5. Number of threads and cooperative iteration in CeSSOL

In CeSSOL, the number of threads depends on the availability of the physical multicore processors. This aspect influences the maximum number of parallel threads which could be launched. This study uses 8 available threads, which is the maximum processing power that the CPU can support. In the parallel environment, when cluster nodes or HPC is used, the processing speed is expected to increase due to the availability of a large number of processors. However, the synchronization of the threads by the proposed method may decrease its scalability. Thus, the increase in speed will not be directly proportional (linear) with the arbitrarily large number of threads (processors). This could occur due to two reasons. First is the communication overhead caused by synchronization between threads. Some of the threads will complete the search process earlier than other threads due to the different parameter settings set for each thread. This will waste the CPU resources, as it will be idle while waiting for other threads. Second is the risk of overlapping by carrying out a similar search in a large number of different threads. This occurs due to the relatively small size of the *RefSet* (Villaverde et al., 2012). Hence, when the number of threads is very large, the resulting speed is smaller than the increase in computational effort. Table 7 displays the parameter values of the number of threads η and number of cooperative iteration co_{iter}. After initial trial and error, the co_{iter} is set to 5, since this value provides the best performance for solving large-scale parameter estimation problems. With co_{iter} defined, the number of function evaluations $nfeval_{co}$ in Eq. (6) can be computed.

5. Experimental setup

5.1. Dataset

Two sets of large-scale kinetic model of biological systems were used namely Chinese hamster ovary (CHO) cells and central carbon metabolism (CCM) in *E. coli*. Both data are standard benchmark data published to test the efficiency of parameter estimation methods. CHO cells consist of 117 kinetic parameters, while CCM

Table 8
Parameter setting in SSCOL and eSS(rerun).

Parameters	Values
Number of diverse solutions (initial solutions)	1170 for CHO and 1160 for CCM
RefSet size	36
Local.n2	10
Balance	0.5

Reference: Remli et al. (2017).

cells consist of 116 parameters. These large number of kinetic parameters cause a serious challenge in terms of computation time in parameter estimation methods (Villaverde et al., 2015).

5.2. Method comparison

The proposed CeSSOL method was executed 20 times as proposed by the benchmark criteria. It is not easy to make a fair comparison between the parallel method (CeSSOL) and sequential methods. By means of the main goals of minimizing computation time and improving results obtained in Remli et al. (2017), the proposed method was compared to a previous study that used a sequential method including DE, PSO, eSS, SSCOL, and another modified eSS namely quasi-opposition enhanced scatter search (QeSS). QeSS is another variant of eSS that modifies the diversification method in the original eSS. The modification is performed by employing quasi-opposition based learning to create an initial solution using diversification generation method. Mainly, QeSS's feature consists of a multi-step diversification method to generate initial random solutions and simultaneously produce quasi-opposite of the initial solutions. Both solutions were compared and the fittest one (in terms of lowest NSL value) was selected as a member of *RefSet*.

In order to make a comparison with sequential methods, the same total number of function evaluations was used, as in previous work. Here, the total number of function evaluations was divided by a number of cooperative iterations. For example, in CHO dataset, the total number of function evaluations of the sequential method reported in previous work was 120,000. Thus, in every cooperative thread, only 24,000 evaluations were performed. Total 5 cooperative iteration was then used to complete 120,000 evaluations. This configuration has been mentioned in Table 3. Thus, the total evaluations for the proposed method were only counted for the threads that were having a good solution compared to other threads in each iteration. This comparison has several disadvantages including a different number of cores and different parameter settings used by the individual sequential and parallel methods, which might result in an unfair comparison. The parameter setting of the sequential methods (eSS and SSCOL) has been shown in Table 8 (Remli et al., 2017).

Therefore, in order to ensure that the best comparison was achieved between these methods, the 8-SSCOL method was implemented which uses 8 cores to run SSCOL in parallel without cooperation. Each core in 8-SSCOL utilized different parameter settings, as practiced by CeSSOL. The difference between CeSSOL and 8-SSCOL is the lack of cooperation among 8-SSCOL threads. The 8-SSCOL method is a sequential method that runs in an embarrassingly parallel fashion. The best result and fastest execution time obtained from the 8 parallel non-cooperative threads of 8-SSCOL was then used to be compared with the proposed method. In addition, the obtained results were compared with recent studies that used original eSS (Villaverde et al., 2015) and saCeSS (Penas et al., 2017). Dell Precision T1700 workstation with Intel Core i7 3.6 Ghz was utilized to carry out the experiments. It consists of 4 physical and 8 logical processors (multithreading technology). Thus, 8 logical processors were used in parallel in this work using MATLAB 2015a and *jPar* library.

6. Result and discussion

6.1. Chinese Hamster Ovary (CHO)

In CHO data, the experimental results report the best, worst, average NLS values and standard deviation over 20 runs, as depicted in Table 9. Overall, the table demonstrates that CeSSOL obtained the best value of 32.696 compared to 8-SSCOL (331.39), SSCOL (34.169), QeSS (35.184), and eSS (36.705). According to this table, CeSSOL also recorded the most consistent and stable results, where it obtained the lowest worst value (48.668) as well as lowest average value (35.982), and lowest standard deviation (3.4732). Based on the results, all methods except CeSSOL and 8-SSCOL have large standard deviations, which indicates that CHO has high multimodality. However, CeSSOL managed to overcome this challenge by means of parallel and cooperative strategy. In addition, CeSSOL has more consistent results for each run and produced the lowest standard deviation (3.4732) compared to other methods. This is due to the advantages of parallel and cooperative search that utilizes the benefit of eight threads and exchange strategy mechanism to explore global minimum.

Additional information for this comparison has been presented in the convergence graph in Fig. 3. In order to perform a fair comparison, same initial guess (randomly generated) was set for all methods, resulting to the same initial NLS value. Based on the figure, it can be noticed that CeSSOL has better convergence speed, followed by 8-SSCOL, SSCOL, and QeSS. In this data, CeSSOL converges to the near-optimal solution when the evaluations reached approximately 100,000. Meanwhile, SSCOL converges to the near-optimal solution when its evaluations reached approximately 64,000. However, after 64,000 evaluations, SSCOL solution stopped improving. The same case was recorded for QeSS, where it converges very early, around 45,000 function evaluations with no further improvement obtained until 120,000 evaluations.

In the proposed method, the best run (minimum value of NLS) from each cooperative iteration was selected as the best convergence and the value was used as the initial solution for the rest of the iteration. Meanwhile, the largest NLS value was selected as the diverse solution. Both solutions (diverse and good) were used as the initial solution to modify systemic properties for all SSCOL threads which could speed up the convergence rate as well.

In terms of computation time, Table 10 depicts the results for average CPU time (seconds) and speedup ratio for the proposed method compared to sequential 8-SSCOL. The table indicates that CeSSOL obtained the highest speedup with 3007.2 seconds (approximately 50 minutes) average CPU time compared to 8-SSCOL that took nearly two hours. The obtained results revealed that CeSSOL is able to reduce computational time compared to original eSS and other methods (from hours to minutes) with speedup ratio of 2.31. This is due to the fact that CeSSOL uses a low number of *RefSet* size and different parameter settings in multiple SSCOL threads which leads to faster evaluation and requires less time to achieve a satisfactory solution. Apart from that, the use of parallel execution that utilized eight available SSCOL threads also resulted in faster computation time.

Next, significance test using Wilcoxon signed ranks test, Friedman test, and nWins procedure were conducted to examine the significant difference between CeSSOL with other methods as well as ranking the methods globally. Table 11 demonstrates the results of the pairwise Wilcoxon signed rank test based on NLS value over 20 runs. Overall, CeSSOL recorded significant result compared to other methods. This is indicated by the lowest p-value obtained (p-value < 0.05) with the level of significance $\alpha = 0.05$ when com-

Table 9
Experimental results for CHO cells over 20 runs.

Method	Nonlinear least squares (NLS)			Standard deviation
	Best value	Worst value	Average value	
DE[+]	$5.3075 \bullet 10^4$	$1.7121 \bullet 10^5$	$9.2626 \bullet 10^4$	$3.3562 \bullet 10^4$
PSO[+]	$2.2378 \bullet 10^2$	$6.5946 \bullet 10^3$	$1.4175 \bullet 10^3$	$1.5119 \bullet 10^3$
eSS (rerun)[+]	$3.6705 \bullet 10^1$	$2.0747 \bullet 10^2$	$9.5537 \bullet 10^1$	$5.2290 \bullet 10^1$
QeSS[+]	$3.5184 \bullet 10^1$	$1.6583 \bullet 10^2$	$8.0930 \bullet 10^1$	$4.8376 \bullet 10^1$
SSCOL[+]	$3.4169 \bullet 10^1$	$1.5499 \bullet 10^2$	$7.6727 \bullet 10^1$	$4.1455 \bullet 10^1$
8-SSCOL	$3.3139 \bullet 10^1$	$6.5189 \bullet 10^2$	$4.0045 \bullet 10^1$	$7.4135 \bullet 10^0$
CeSSOL[*]	$3.2696 \bullet 10^1$	$4.8668 \bullet 10^1$	$3.5982 \bullet 10^1$	$3.4732 \bullet 10^0$

[*] The proposed method in this work. Shaded cell represents the best overall result.
[+] Results taken from Remli et al. (2017). 'eSS (rerun)' is the result from Remli et al. (2017) (to differentiate from eSS result of the original publication).

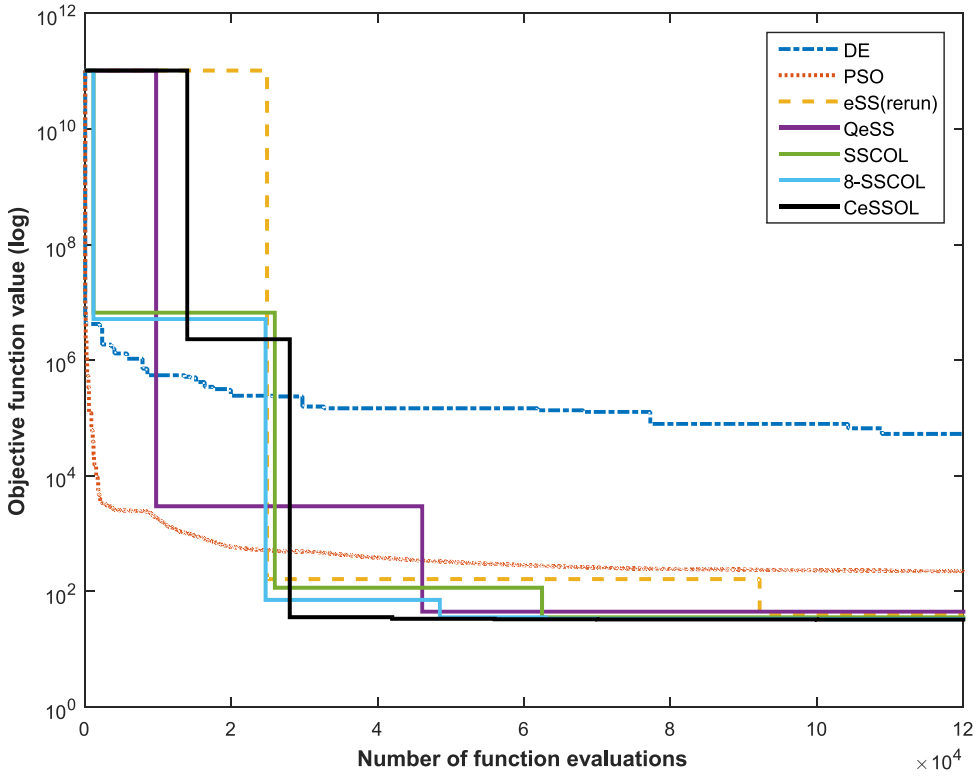

Fig. 3. Convergence curves of best runs for CHO cells.

Table 10
Computation time and speedup results for CHO cells over 20 runs.

Method	Average CPU time (seconds)	Speedup Ratio
8-SSCOL	$6.9401 \bullet 10^3$	1.00
CeSSOL[*]	$3.0072 \bullet 10^3$	2.31

[*] The proposed method in this work. Shaded cell represents the lowest computation time with the highest speedup using 8 threads (processors) with cooperation.

Table 11
Results of the Wilcoxon signed ranks test for CHO based on NLS value.

Comparison	R+	R-	*p*-value
CeSSOL vs DE	210	0	0.000089
CeSSOL vs PSO	210	0	0.000089
CeSSOL vs eSS (rerun)	206	4	0.000163
CeSSOL vs QeSS	209	1	0.000103
CeSSOL vs SSCOL	203	7	0.000254
CeSSOL vs 8-SSCOL	162	48	0.033340

Note: R^+ represents the sum of ranks (one method outperformed the others) and R^- represents the sum of ranks for the opposite.

pared to other methods via pairwise comparison. The null hypothesis H_0, where there is no statistically significant difference between CeSSOL and other methods (DE, PSO, eSS (rerun), QeSS, 8-SSCOL and SSCOL), was rejected and alternate hypothesis H_1, which is the opposite of H_0, was accepted.

Based on Table 11, the nWins procedure was conducted. The best methods which produced larger R^+ than R^- values and *p*-values smaller than 0.05 were granted +1. The losing methods (R^- larger than R^+) with *p*-values smaller than 0.05 were granted −1. Furthermore, the Friedman test was also used to rank all methods. The result of this analysis has been presented in Table 12. Overall, CeSSOL performed excellently compared to other methods with Friedman and nWins values of 1.55 and 6, respectively.

Additionally, results obtained from this work were compared with a recent study that used eSS (Villaverde et al., 2015) and a recent publication on saCeSS (Penas et al., 2017) for the same dataset, as shown in Table 13. It can be seen that CeSSOL obtained a better objective function value J_f (32.696) compared to other methods. In terms of computational cost, specifically for

Table 12
Global ranking of all methods for CHO.

Method	Friedman ranking	nWins
CeSSOL*	1.55	6
8-SSCOL	2.05	5
SSCOL	3.60	0
QeSS	3.65	0
eSS (rerun)	4.15	0
PSO	6.00	−5
DE	7.00	−6

* The proposed method in this work. The lowest mean rank value is the best method and the highest mean rank value is the worst method based on Friedman ranking. Shaded row represents the best overall result based on Friedman ranking and nWins.

computational time, saCeSS has far better CPU time, where it only consumes 343 seconds and obtains 55.000 J_f. Two factors have contributed to the better computation time for the reported works; first is the different stopping condition used and the second is the different hardware platform utilized. In the first study (Villaverde et al., 2015), the stopping condition of eSS is based on CPU time, while saCeSS used value to reach (VTR) as its stopping condition. Meanwhile, this study employed a number of function evaluations as its stopping condition, which leads to different performance behavior.

It should be noted that the metrics used for NLS J_f and NRMSE are different. Thus, it could lead to different estimation behavior. For instance, from the CHO result presented in Table 9, it can be seen that the lowest objective function J_f value (32.696) is obtained from the parameter estimated using CeSSOL. However, the $\Sigma NRMSE_f$ of CeSSOL (2.8056) is slightly higher than $\Sigma NRMSE_f$ of eSS (re-run) (2.7951), although J_f from CeSSOL is lower (better) than J_f from eSS (rerun). Since the lowest J_f is expected to produce the lowest $\Sigma NRMSE_f$, there are several possible explanations that caused this contradiction. First, this data have un-identifiability issue (Almquist et al., 2014; Raue et al., 2009; Raue, Karlsson, Saccomani, Jirstrand, & Timmer, 2014). Second, due to the nature of these large-scale kinetic models (model with hundreds of kinetic parameters), near-optimal solutions have resulted to over-fitting in artificial noise. These issues resulted in different behavior for J_f and $\Sigma NRMSE_f$ in CHO.

Regarding the issue of un-identifiability, lack of identifiability indicates that there are some possible model parameters with different values that could obtain the same agreement to the model prediction in ODEs (model output) (Gábor et al., 2017). Thus, due to this issue, it was observed that for the case of CHO, the lowest J_f does not necessarily provide the lowest $\Sigma NRMSE_f$. Identifiability analysis of this model has been conducted and it was found that some parameters had little influence on the model output (Villaverde et al., 2015). In order to illustrate the identifiability issue on CHO data, Fig. 4 illustrates the diagonal plots between nominal parameters (model parameters that generated the data) and near-optimal parameters which are obtained from CeSSOL. Plots indicated that the difference between nominal and optimal values was large. In other words, nominal and near-optimal values of parameters were quite different; however, they resulted in the same prediction as in the diagonal plot shown in Fig. 5.

Regarding the overfitting issue, the lowest J_f obtained by the proposed method provided a better fit to the experimental data (with artificial noise) rather than the one obtained with the nominal parameters used to generate the data. This occurs due to the presence of artificial noise in CHO, the near-optimal solutions do not solely provide the best fits for ODEs, but they also try to produce a fit for artificial noise (which cannot be obtained from nominal parameters) (Villaverde et al., 2015). Although the reported results in the literature are better for computational time (due to different stopping condition and platforms), in terms of solution quality (lowest value of nonlinear least squares), this work obtained the lowest value compared to the results reported in Table 13. Hence, it is expected that the proposed method will produce a better model fit.

In order to assess the quality of the fitted models, normalized root mean square error (NRMSE) was used. It is a standard measure for goodness of model fit. The NRMSE values for each metabolite have been tabulated in Table 14. Lowest NRMSE indicates the best fit for each metabolite. Based on the table, the proposed method in this study produced the lowest NRMSE (best fit) for L-Lactate, as illustrated in Fig. 6.

6.2. Central Carbon Metabolism (CCM) of E. coli

The computational cost for CCM of *E. coli* is more expensive compared to CHO. Due to the high complexity and high dimensionality of this data, the process to evaluate NLS is costly where it consumes long CPU time. Table 5 depicts the experimental results where it reports the best, worst, and average values obtained as well as their standard deviation over 20 runs. Overall, the results revealed that the proposed method (CeSSOL) obtained excellent results with the best minimum and average values of the objective function (208.58 and 229.77). Similar pattern was observed in CHO data, as seen in the CCM data where the lowest NLS value was obtained from CeSSOL, followed by SSCOL and QeSS. However, in terms of consistency, there is not much difference in the standard deviation values of different runs. In addition, CeSSOL pro-

Table 13
Comparison of best results between the proposed method and previous works for CHO.

Method	CPU time (s)	J_f	J_{nom}	$\Sigma NRMSE_f$	$\Sigma NRMSE_{nom}$
eSS	$3.6000 \bullet 10^3$	$4.5718 \bullet 10^1$	$3.9068 \bullet 10^1$	2.8010	2.8273
SaCeSS	$3.4300 \bullet 10^2$	$5.5000 \bullet 10^1$	Na	Na	Na
DE+	$2.2049 \bullet 10^4$	$5.3075\ 10^4$	$3.9068 \bullet 10^1$	4.3453	2.8273
PSO+	$3.9899 \bullet 10^3$	$2.2379\ 10^2$	$3.9068 \bullet 10^1$	2.9410	2.8273
eSS (re-run)+	$8.7175 \bullet 10^3$	$3.6705\ 10^1$	$3.9068 \bullet 10^1$	2.7951	2.8273
QeSS+	$6.8297 \bullet 10^3$	$3.5514\ 10^1$	$3.9068 \bullet 10^1$	2.8226	2.8273
SSCOL+	$6.7116 \bullet 10^3$	$3.4169 \bullet 10^1$	$3.9068 \bullet 10^1$	2.8048	2.8273
8-SSCOL	$6.2303 \bullet 10^3$	$3.3139 \bullet 10^1$	$3.9068 \bullet 10^1$	2.8111	2.8273
CeSSOL*	$2.9880 \bullet 10^3$	$3.2696 \bullet 10^1$	$3.9068 \bullet 10^1$	2.8056	2.8273

* The proposed method in this research.eSS result is obtained from Villaverde et al. (2015), while SaCeSS is obtained from Penas et al. (2017).

+ Results taken from Remli et al. (2017).NRMSE = Normalized root-mean-square-error.The best (minimum) number of function evaluations, the best (minimum) CPU time (s), the best (minimum) objective function (nonlinear least squares) values J_f and the best sum of $\Sigma NRMSE_f$ are indicated in shaded cells.J_{nom}= Objective function values obtained from nominal parameters p_{nom} $\Sigma NRMSE_{nom}$= Sum of NRMSE with nominal parameters.

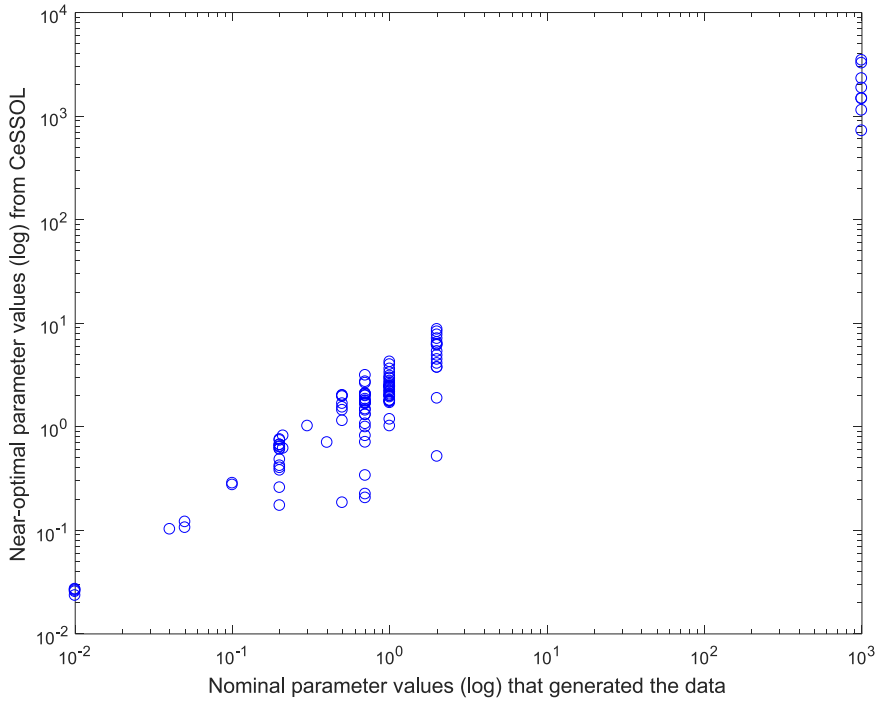

Fig. 4. Difference between the nominal parameters (log) and near-optimal parameters (log) in CHO.

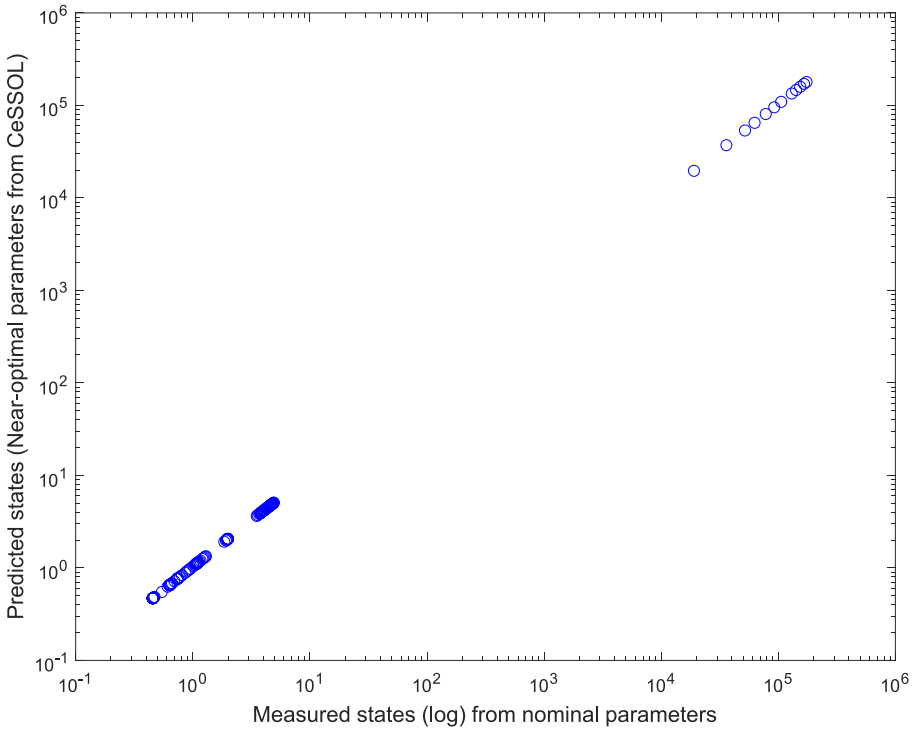

Fig. 5. Difference between the measured states with nominal parameters and predicted states with near-optimal parameters in CHO.

duced the lowest standard deviation (10.150), followed by SSCOL (11.891), and QeSS (17.463). This indicated that for parameter estimation, CCM data is less challenging compared to CHO, where all the competitive methods (except DE) were able to find satisfactory solutions.

The convergence curve of the best run for this data has been shown in Fig. 7. Except for DE, all methods produced a high speed of convergence in the early stage of evaluation. However, after 70,000 evaluations, CeSSOL produced the best objective value followed by 8-SSCOL, SSCOL, QeSS, eSS (rerun), PSO, and DE. In view of computation time, Table 16 represents the average computation time and speedup ratio for CCM data. Similar to the previous data, CeSSOL was able to reduce the computation time, taking nearly two and half hours compared to 8-SSCOL which consumed nearly six hours. This result indicates that CeSSOL was able to obtain better average CPU time compared to other methods.

Based on the results of the nonparametric tests presented in Tables 17 and 18, the proposed method yields to significant im-

Table 14
NRMSE values for all parameters estimated in CHO.

Metabolite	NRMSE value						
	DE[+]	PSO[+]	eSS (rerun)[+]	QeSS	SSCOL[+]	8-SSCOL	CeSSOL[*]
Product Protein	0.1625	0.13132	0.1318	0.13138	0.13176	0.1318	0.13172
L-Methionine	0.27479	0.2587	0.25385	0.25372	0.25325	0.2543	0.25437
L-Leucine	0.1799	0.18079	0.181	0.18079	0.18096	0.1810	0.18099
L-Lactate	0.051246	0.048955	0.046447	0.046562	0.046397	0.0464	0.046315
beta-D-Glucose	0.14469	0.14636	0.14412	0.14413	0.14432	0.1446	0.14466
L-Aspartate	0.32515	0.089176	0.086051	0.086009	0.08528	0.0851	0.08522
L-Malate	0.28523	0.24388	0.24943	0.24874	0.24939	0.2495	0.2495
Pyruvate	0.24223	0.25596	0.2563	0.25805	0.25743	0.2579	0.25751
Oxaloacetate	0.52836	0.54308	0.39091	0.4166	0.39758	0.4035	0.39919
ATP (Cytosol)	0.27957	0.28514	0.2851	0.28525	0.28514	0.2851	0.28518
ATP (Mitochondria)	0.42778	0.25551	0.25054	0.25478	0.25342	0.2527	0.25466
ADP (Mitochondria)	1.1554	0.24221	0.25909	0.26445	0.26273	0.2620	0.26208
ADP (Cytosol)	0.2884	0.25988	0.26047	0.25215	0.25716	0.2572	0.2542

[*] The proposed method in this research. Shaded cells represent the lowest (best) NRMSE values for each metabolite.
[+] Results taken from Remli et al. (2017).

Table 15
Experimental results for CCM of *E. coli* over 20 runs.

Method	Objective function (Nonlinear least squares)			Standard deviation
	Best value	Worst value	Average value	
DE[+]	$4.1203 \bullet 10^2$	$6.1407 \bullet 10^2$	$5.1741 \bullet 10^2$	$4.5680 \bullet 10^1$
PSO[+]	$2.3787 \bullet 10^2$	$4.1090 \bullet 10^3$	$1.1403 \bullet 10^3$	$1.5838 \bullet 10^3$
eSS (rerun)[+]	$2.2918 \bullet 10^2$	$2.7007 \bullet 10^2$	$2.4843 \bullet 10^2$	$1.0465 \bullet 10^1$
QeSS[+]	$2.3129 \bullet 10^2$	$2.9771 \bullet 10^2$	$2.5188 \bullet 10^2$	$1.7463 \bullet 10^1$
SSCOL[+]	$2.0992 \bullet 10^2$	$2.5642 \bullet 10^2$	$2.4107 \bullet 10^2$	$1.1891 \bullet 10^1$
8-SSCOL	$2.2980 \bullet 10^2$	$2.4541 \bullet 10^2$	$2.3688 \bullet 10^2$	$5.6121 \bullet 10^0$
CeSSOL[*]	$2.0858 \bullet 10^2$	$2.4555 \bullet 10^2$	$2.2977 \bullet 10^2$	$1.0150 \bullet 10^1$

[*] The proposed method in this research. Shaded cell represents the best overall result.
[+] Results adopted from Remli et al. (2017).

Table 16
Computation time and speedup results for CCM over 20 runs.

Method	Average CPU time (seconds)	Speedup Ratio
8-SSCOL	$2.1071 \bullet 10^4$	1.00
CeSSOL[*]	$8.5844 \bullet 10^3$	2.45

[*] The proposed method in this research. Shaded cell represents the lowest computation times with the highest speedup using 8 threads (processors).

Table 17
Results of the Wilcoxon signed ranks test for CCM based on NLS value.

Comparison	R+	R-	*p*-value
CeSSOL vs DE	210	0	0.000089
CeSSOL vs PSO	209	1	0.000103
CeSSOL vs eSS (rerun)	209	1	0.000103
CeSSOL vs QeSS	203	7	0.000254
CeSSOL vs SSCOL	184	26	0.003185
CeSSOL vs 8-SSCOL	182	28	0.004045

Note: R^+ represents the sum of ranks (one method outperformed the others) and R^- represents the sum of ranks for the opposite.

Table 18
Global ranking of all methods for CCM.

Method	Friedman ranking	nWins
CeSSOL[*]	1.55	6
8-SSCOL	2.50	4
SSCOL	3.40	1
QeSS	4.20	0
eSS (rerun)	4.25	−1
PSO	5.35	−5
DE	6.75	−5

[*] The proposed method in this work. The lowest mean rank value is the best method and the highest mean rank value is the worst method, based on Friedman ranking. Shaded row represents the best overall result based on Friedman ranking and nWins.

provements over other methods. This was demonstrated by the result of Wilcoxon pairwise comparison that recorded *p*-value lower than 0.05. On the other hand, the proposed method also ranked first in terms of Friedman ranking and nWins procedure. By means of a parallel strategy, the proposed method could improve parameter estimation result in terms of its ability to minimize NLS values, convergence speed, and computation time.

Table 19 demonstrates a comparison between this study and two other studies which utilized eSS (Villaverde et al., 2015) and saCeSS (Penas et al., 2017). The table indicates that the proposed method in this study produced superior results in terms of the lower objective function value J_f and lower value of $\Sigma NRMSE_f$. In terms of CPU time, saCeSS obtained the lowest time (1694 seconds). From the comparison, it can be noticed that this study obtained the lowest $\Sigma NRMSE_f$ value compared to the benchmark and original $\Sigma NRMSE_{ex}$, indicating that the parameters estimated by this study provides the best possible fit to the real experimental data. The NRMSE values for each metabolite have been shown in Table 20, where the lowest NRMSE indicates the best fit for each metabolite. Based on the table, the proposed method in this study produced the lowest NRMSE for GAP. Specifically, model parameter values obtained from CeSSOL produced the best fit for the GAP in CCM, as shown in Fig. 8. On the other hand, parameter values obtained from SSCOL produced the best fit for PEP, G6P, and GLCex. Other than that, the lowest NRMSE for other metabolites were obtained by other methods. Values of kinetic parameters obtained from all methods employed in this experiment including DE, PSO, eSS (rerun), QeSS, SSCOL, 8-SSCOL, and CeSSOL have been provided in supplementary material.

It was interesting to observe that the proposed method does not yield the lowest (best) values for all metabolites in CHO and CCM data. In some metabolites, parameters estimated from other methods provided slightly better NRMSE values, except for DE

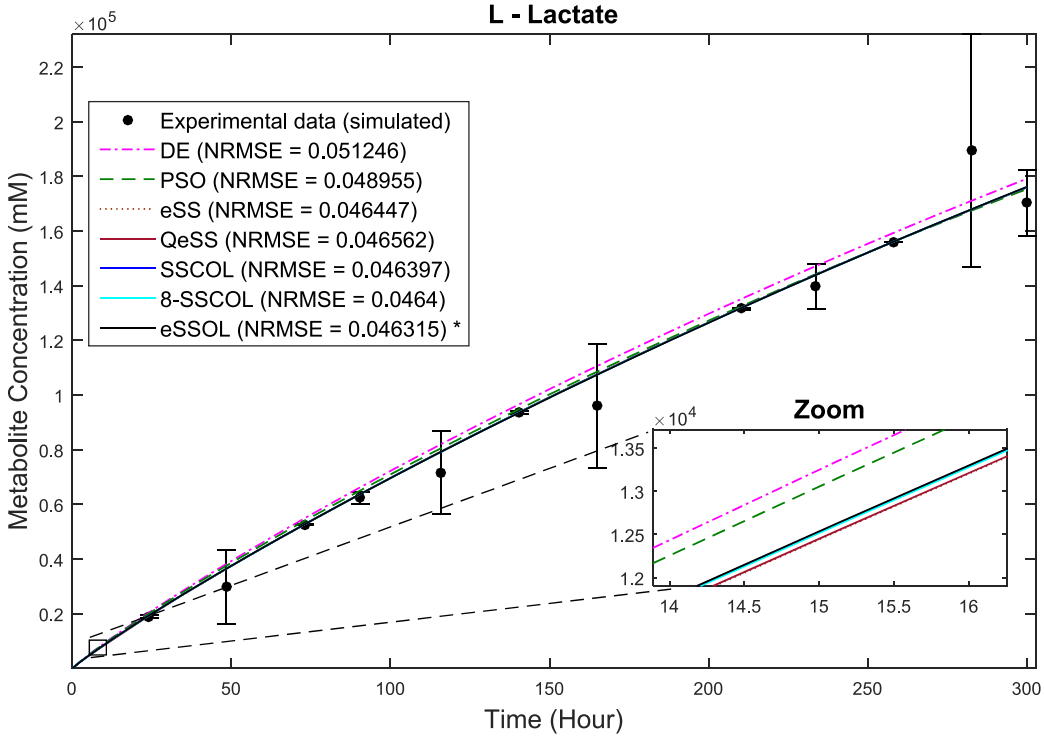

Fig. 6. Fitted model of L-Lactate metabolite. Experimental data with artificial noise is shown as bars. The fittest model was obtained from CeSSOL.

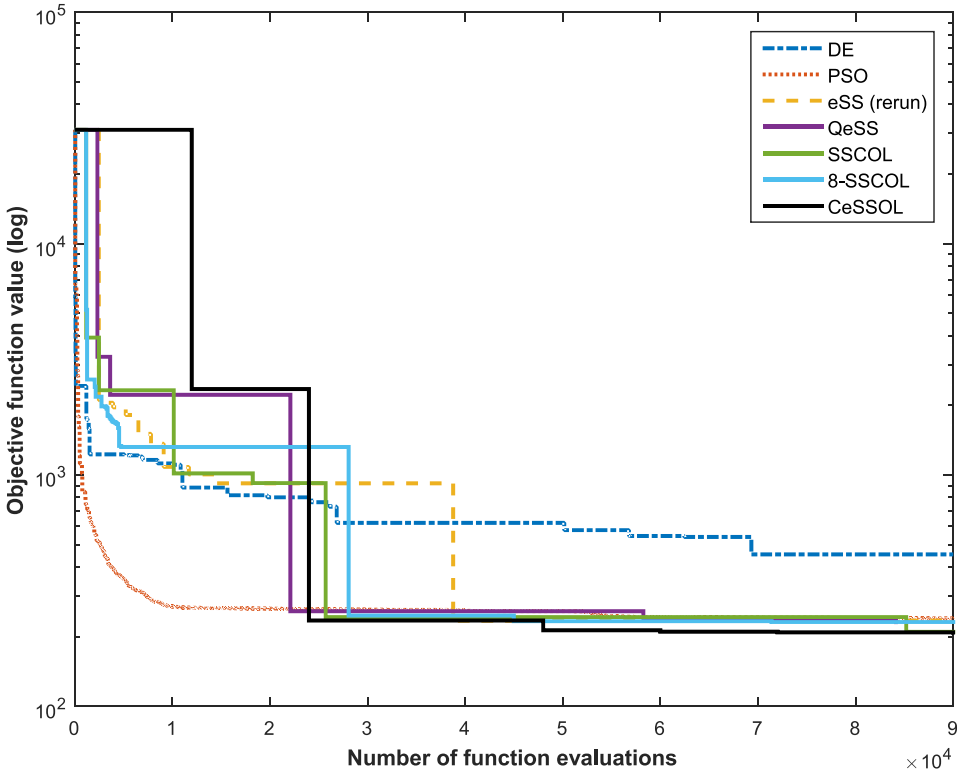

Fig. 7. Convergence curves of best run for CCM of *E. coli*.

in which it resulted to poor fit in most of the metabolites. Accordingly, it is believed that the minimum value of the objective function J_f does not guarantee the lowest NRMSE for all metabolites. This indicates that although the minimum objective function values were obtained by CeSSOL, it did not produce the lowest NRMSE for all metabolites for both data (CHO and CCM) compared to other methods. This situation occurs due to the higher order interrelationship between kinetic parameters in ODEs and interdependency between metabolites (state variables). Thus, due to the stochastic (random) nature of the problems and several metabolites (13 in CHO and 9 in CCM) considered for estimation, some methods may produce the lowest NRMSE values for partic-

Table 19
Comparison of best results between the proposed method and previous works for CCM of *E. coli*.

Method	CPU time (s)	J_f	J_{ex}	$\Sigma NRMSE_f$	$\Sigma NRMSE_{ex}$
eSS	$1.0800 \bullet 10^4$	$2.3390 \bullet 10^2$	$3.1087 \bullet 10^4$	2.8010	3.6065
saCeSS	$1.6940 \bullet 10^3$	$2.5000 \bullet 10^2$	Na	Na	Na
DE+	$5.2229 \bullet 10^4$	4.1203 10^2	$3.1087 \bullet 10^4$	3.2341	3.6065
PSO+	$2.3248 \bullet 10^4$	2.3786 10^2	$3.1087 \bullet 10^4$	2.6138	3.6065
eSS (re-run)+	$4.3591 \bullet 10^4$	2.2918 10^2	$3.1087 \bullet 10^4$	2.4349	3.6065
QeSS+	$2.7770 \bullet 10^4$	2.3129 10^2	$3.1087 \bullet 10^4$	2.4677	3.6065
SSCOL+	$2.7605 \bullet 10^4$	$2.0992 \bullet 10^2$	$3.1087 \bullet 10^4$	2.3879	3.6065
8-SSCOL	$2.4271 \bullet 10^4$	$2.2980 \bullet 10^2$	$3.1087 \bullet 10^4$	2.4131	3.6065
CeSSOL*	$1.3996 \bullet 10^4$	$2.0858 \bullet 10^2$	$3.1087 \bullet 10^4$	2.3804	3.6065

Note: eSS result is obtained from Villaverde et al. (2015), while saCeSS obtained from Penas et al. (2017).
+ Results taken from Remli et al. (2017).The best (minimum) number of function evaluations, the best (minimum) CPU time (s), the best (minimum) objective function (nonlinear least squares) values J_f and the best $\Sigma NRMSE_f$ are indicated in shaded cells.J_{ex}= Objective function values obtained from original the parameters reported in publication.$\Sigma NRMSE_{ex}$= Sum of normalized root-mean-square-error with original parameters reported in publication.

Table 20
NRMSE values of all metabolites in CCM of *E. coli*.

Metabolite	NRSME value						
	DE+	PSO+	eSS (rerun)+	QeSS+	SSCOL+	8-SSCOL	CeSSOL*
PEP	0.6446	0.4430	0.4162	0.4155	0.3714	0.4080	0.4031
G6P	0.3124	0.1489	0.1388	0.1284	0.1234	0.1419	0.1276
PYR	0.3323	0.2440	0.2521	0.2555	0.2631	0.2547	0.2597
F6P	0.1701	0.2072	0.1823	0.1890	0.1780	0.1783	0.1789
ex. GLC	0.1912	0.2045	0.1906	0.1904	0.1836	0.1865	0.1854
GLP	0.2112	0.2439	0.1751	0.1868	0.1976	0.1634	0.1924
6PG	0.2698	0.1667	0.1058	0.1215	0.1187	0.0961	0.1097
FDP	0.2951	0.2104	0.2496	0.2464	0.2391	0.2583	0.2402
GAP	0.8073	0.7453	0.7244	0.7342	0.7129	0.7260	0.6835

* The proposed method in this research. Shaded cells represent the lowest (best) NRMSE value for each metabolite.
+ Results taken from Remli et al. (2017).

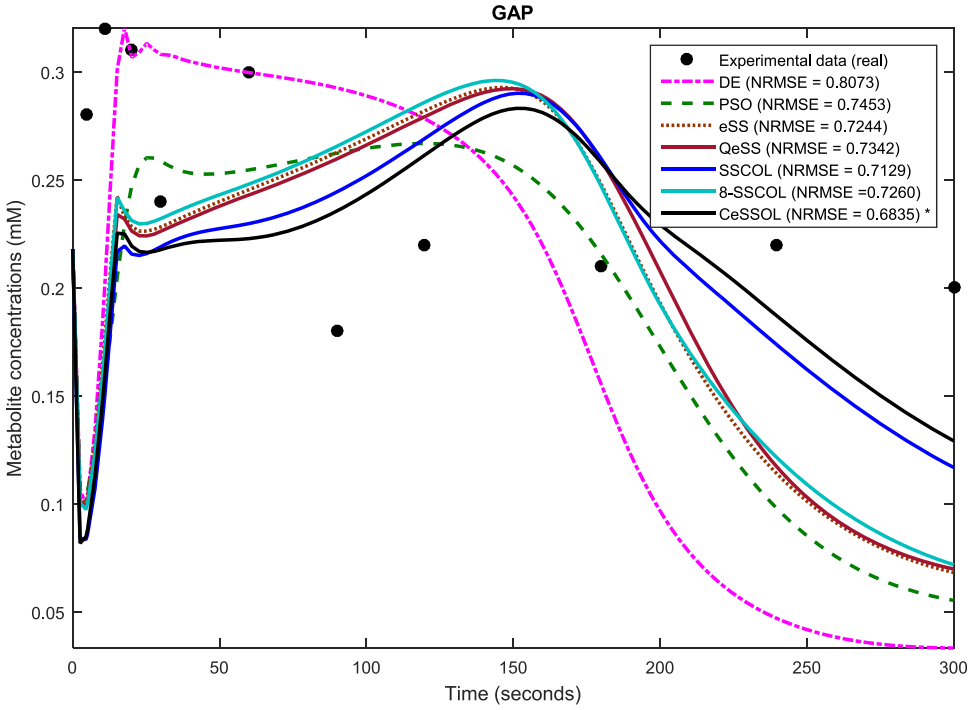

Fig. 8. Fitted model of GAP metabolite. The lowest NRMSE is obtained from CeSSOL.

ular metabolites, although they do not produce lowest objective functions J_f which contain the sum of NLS values for all metabolites.

The cooperative method proposed in this study aims to reduce the computation time as well as maintaining the minimum (best) solution quality. Based on the obtained NLS values, CeSSOL has managed to produce the lowest value compared to other methods including eSS and eSSCOL for both CHO and CCM data. The advantages of information sharing by the proposed method have resulted in improved systemic properties in all sequential CeSSOL threads,

and at the same time, improved the values of the objective function and recorded faster computation time. Providing high-quality kinetic parameter values to be used to guide the search process in all parallel threads by using a combination of opposition-based learning schemes is the important factor that contributed to the lowest (improved) NRMSE for L-Lactate (in CHO cells) and GAP (in CCM). The method is able to explore different regions of parameter space based on the proposed cooperative mechanism that shares the best value and diverse (worst) value among eSSCOL threads. The proposed method provides a better alternative to search for unknown parameter values in a large-scale kinetic model of biological systems. Based on the experimental outcomes, the parameter values obtained from the proposed methods produced the best-fitted model prediction of two biological models which have been widely utilized in industrial biotechnology research. The fitted model that contains metabolites concentrations could be used in simulation and prediction of biological systems. The usage of these models will save both time and resources.

7. Conclusion

Preparing a predictive model of highly nonlinear metabolic processes of microorganism cells is known to be challenging and time-consuming. The challenging issue that arises in such model building is parameter estimation that involves the process of searching for near-optimal values of kinetic parameters. The values of parameters influence the accuracy of model prediction, where they determine the distance between model prediction and experimental data. This issue received great attention in both systems biology and metabolic engineering fields, especially involving large-scale models. Metaheuristic methods based on sequential global optimization are suitable candidates to surmount this issue. However, their main drawback is in terms of performance deterioration due to high computational cost (in terms of function evaluations and CPU time) when solving high-dimensional data.

In this study, an improved cooperative metaheuristic method with information exchange strategy was proposed in order to solve the stated issues. The proposed method was based on parallel and cooperative search. Parallel metaheuristic method based on scatter search with combined opposition-based learning (SSCOL) was executed in multicores CPU on a single machine. Each thread utilized a small number of *RefSet* to reduce the computational effort, which could speed up the search process. It was observed that the proposed method was able to reduce computation time by two times, compared to the sequential method. Moreover, the obtained results revealed that the method does not solely result in faster computation time but is also efficient in discovering near-optimal solutions. This is due to the fact that the proposed information exchange strategy enhances the systemic properties and search pattern in each SSCOL thread. The exchange strategy was based on good and diverse solutions obtained from previously known solutions. These two solutions with the combination of opposition-based learnings schemes were used to guide the overall minimization process for all cooperative threads. However, the main drawback of the proposed method is its inefficient resource (CPU) usage since not all threads were used to obtain the good and diverse solutions (only two threads out of eight were used in information exchange). Furthermore, the information exchange in this method leads to an unbalanced scenario where the threads have to wait for all threads to finish their tasks before new tasks can be launched, which decreased the threads' scalability. As future work, further study will be performed on the following:

(i) Additional information exchange strategies will be proposed and analyzed. Current work solely utilized two solutions (best and diverse) among the threads to be imported as initial solutions. It led to the waste of CPU usage, while some of the other threads did not produce the best and diverse solutions. Therefore, the performance and systemic behavior of the proposed method is expected to be influenced when the best and diverse solutions for each thread is imported as an initial solution in information exchange strategy.

(ii) Implementation of cooperative search using parallel distributed computing including a graphic processing unit (GPU), high-performance computing (HPC) and computer clusters (at least local cluster) are the promising direction in this research. This can be beneficial if the data is huge (thousands of kinetic parameters), especially if it involves a full genome metabolic model.

(iii) Prior to performing parameter estimation in large-scale models, it is crucial to conduct an identifiability analysis of model parameters. This could be performed using several statistical techniques such as the Fisher Information Matrix in order to rank the kinetic parameters based on their sensitivity. Identifiability problem occurs when the portion of kinetic parameters has little influence on the model output. This analysis is important in order to group the parameters based on sensitivity ranks and could reduce the model complexity.

(iv) Additionally, the robustness of the proposed method can be investigated by applying the method in other challenging and computational expensive domains such as in large-scale global optimization (LSGO) problem.

Acknowledgment

We would like to thank the Malaysian Ministry of Higher Education and Universiti Teknologi Malaysia for their support via the Fundamental Research Grant Scheme (grant number: R.J130000.7828.4F886) and Flagship Grant Scheme (grant number: Q.J130000.2428.03G57).

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.eswa.2018.09.020.

References

Ahn, W. S., & Antoniewicz, M. R. (2012). Towards dynamic metabolic flux analysis in CHO cell cultures. *Biotechnology Journal, 7*(1), 61–74. http://doi.org/10.1002/biot.201100052.

Alba, E., & Hoboken, N. J. (2005). *Parallel metaheuristics: a new class of algorithms* (1st ed). John Wiley and Sons *http://doi.org/10.1002/0471739383*.

Almquist, J., Cvijovic, M., Hatzimanikatis, V., Nielsen, J., & Jirstrand, M. (2014). Kinetic models in industrial biotechnology - Improving cell factory performance. *Metabolic Engineering, 24*, 38–60. http://doi.org/10.1016/j.ymben.2014.03.007.

Billa, T., Horton, S. R., Sahasrabudhe, M., Saravanan, C., Hou, Z., Agarwal, P., et al. (2017). Enhancing the value of detailed kinetic models through the development of interrogative software applications. *Computers and Chemical Engineering, 106*, 512–528. http://doi.org/10.1016/j.compchemeng.2017.07.009.

Chong, C. K., Mohamad, M. S., Deris, S., Shamsir, M. S., Chai, L. E., & Choon, Y. W. (2014). Parameter estimation by using an improved bee memory differential evolution algorithm (IBMDE) to simulate biochemical pathways. *Current Bioinformatics, 9*(1), 65–75. http://doi.org/10.2174/15748936113080990007.

Crainic, T. G., & Gendreau, M. (2002). Cooperative parallel tabu search for capacitated network design. *Journal of Heuristics, 8*(6), 601–627. http://doi.org/10.1023/A:1020325926188.

Crainic, T. G., Gendreau, M., Hansen, P., & Mladenović, N. (2004). Cooperative parallel variable neighborhood search for the p-median. *Journal of Heuristics, 10*(3), 293–314. http://doi.org/10.1023/B:HEUR.0000026897.40171.1a.

Crainic, T. G., & Toulouse, M. (2010). Parallel strategies for meta-heuristics. In F. Glover, & G. A. Kochenberger (Eds.), *Handbook of metaheuristics* (pp. 475–513). Boston, MA.: Springer. http://doi.org/10.1007/0-306-48056-5_17.

Cruz, C., & Pelta, D. (2009). Soft computing and cooperative strategies for optimization. *Applied Soft Computing Journal, 9*(1), 30–38. http://doi.org/10.1016/j.asoc.2007.12.007.

Cvijovic, M., Bordel, S., & Nielsen, J. (2011). Mathematical models of cell factories: Moving towards the core of industrial biotechnology. *Microbial Biotechnology, 4*(5), 572–584. http://doi.org/10.1111/j.1751-7915.2010.00233.x.

Dobson, P. D., Smallbone, K., Jameson, D., Simeonidis, E., Lanthaler, K., Pir, P., et al. (2010). Further developments towards a genome-scale metabolic model of yeast. *BMC Systems Biology, 4*, 145. http://doi.org/10.1186/1752-0509-4-145.

Egea, J. A., & Balsa-Canto, E. (2009). Dynamic optimization of nonlinear processes with an enhanced scatter search method. *Industrial & Engineering Chemical Research, 48*(9), 4388–4401.

Egea, J. A., Henriques, D., Cokelaer, T., Villaverde, A. F., MacNamara, A., Danciu, D.-P., et al. (2014). MEIGO: An open-source software suite based on metaheuristics for global optimization in systems biology and bioinformatics. *BMC Bioinformatics, 15*, 136. http://doi.org/10.1186/1471-2105-15-136.

Egea, J. A., Martí, R., & Banga, J. R. (2010). An evolutionary method for complex-process optimization. *Computers and Operations Research, 37*(2), 315–324. http://doi.org/10.1016/j.cor.2009.05.003.

El-Abd, M., & Kamel, M. (2005). A taxonomy of cooperative search algorithms. In M. J. Blesa, C. Blum, A. Roli, & M. Sampels (Eds.). In *Hybrid metaheuristics: 3636* (pp. 32–41). BerlinHeidelberg: Springer. http://doi.org/10.1007/11546245_4.

Gábor, A., Villaverde, A. F., & Banga, J. R. (2017). Parameter identifiability analysis and visualization in large-scale kinetic models of biosystems. *BMC Systems Biology, 11*(1), 54. http://doi.org/10.1186/s12918-017-0428-y.

Gottu Mukkula, A. R., & Paulen, R. (2017). Model-based design of optimal experiments for nonlinear systems in the context of guaranteed parameter estimation. *Computers & Chemical Engineering, 99*, 198–213. http://doi.org/10.1016/j.compchemeng.2017.01.029.

Jahan, N., Maeda, K., Matsuoka, Y., Sugimoto, Y., & Kurata, H. (2016). Development of an accurate kinetic model for the central carbon metabolism of Escherichia coli. *Microbial Cell Factories, 15*(1), 112–130. http://doi.org/10.1186/s12934-016-0511-x.

Karbowski, A., Majchrowski, M., Trojanek, P., Pokorski, T., & Załuga, D. (2015). jPar – a simple, free and lightweight tool for parallelizing Matlab calculations on multicores and in clusters. In *Federated conference on computer science and information systems: 6* (pp. 91–96). PTI. http://doi.org/10.15439/2015F233 .

Karr, J. R., Takahashi, K., & Funahashi, A. (2015). The principles of whole-cell modeling. *Current Opinion in Microbiology, 27*, 18–24. http://doi.org/10.1016/j.mib.2015.06.004.

Li, G., & Wang, Q. (2014). A cooperative harmony search algorithm for function optimization. *Mathematical Problems in Engineering, 2014*(1), 1–13. http://doi.org/10.1155/2014/587820.

Li, P., & Vu, Q. D. (2013). Identification of parameter correlations for parameter estimation in dynamic biological models. *BMC Systems Biology, 7*. http://doi.org/10.1186/1752-0509-7-91.

Macklin, D. N., Ruggero, N. A., & Covert, M. W. (2014). The future of whole-cell modeling. *Current Opinion in Biotechnology, 28*, 111–115. http://doi.org/10.1016/j.copbio.2014.01.012.

Martin, S., Ouelhadj, D., Smet, P., Vanden Berghe, G., & Ozcan, E. (2013). Cooperative search for fair nurse rosters. *Expert Systems with Applications, 40*(16), 6674–6683. http://doi.org/10.1016/j.eswa.2013.06.019.

Miskovic, L., Tokic, M., Fengos, G., & Hatzimanikatis, V. (2015). Rites of passage: Requirements and standards for building kinetic models of metabolic phenotypes. *Current Opinion in Biotechnology, 36*, 146–153. http://doi.org/10.1016/j.copbio.2015.08.019.

Mohd Zain, M. Z., Kanesan, J., Kendall, G., & Chuah, J. H. (2018). Optimization of fed-batch fermentation processes using the backtracking search algorithm. *Expert Systems with Applications, 91*, 286–297. http://doi.org/10.1016/j.eswa.2017.07.034.

Moles, C. G., Mendes, P., & Banga, J. R. (2003). Parameter estimation in biochemical pathways: A comparison of global optimization methods. *Genome Research, 13*(11), 2467–2474. http://doi.org/10.1101/gr.1262503.

Nedjah, N., Calazan, R., de, M., Mourelle, L., de, M., & Wang, C. (2016). Parallel implementations of the cooperative particle swarm optimization on many-core and multi-core architectures. *International Journal of Parallel Programming, 44*(6), 1173–1199. http://doi.org/10.1007/s10766-015-0368-3.

Ngo, T. T., Sadollah, A., & Kim, J. H. (2016). A cooperative particle swarm optimizer with stochastic movements for computationally expensive numerical optimization problems. *Journal of Computational Science, 13*, 68–82. http://doi.org/10.1016/j.jocs.2016.01.004.

Oh, E., Lu, M., Park, C., Oh, H. Bin, Lee, S. Y., & Lee, J. (2011). Dynamic modeling of lactic acid fermentation metabolism with Lactococcus lactis. *Journal of Microbiology and Biotechnology, 21*(2), 162–169. http://doi.org/10.4014/jmb.1007.07066.

Park, J. H., Kim, T. Y., Lee, K. H., & Lee, S. Y. (2011). Fed-batch culture of Escherichia coli for L-valine production based on in silico flux response analysis. *Biotechnology and Bioengineering, 108*(4), 934–946. http://doi.org/10.1002/bit.22995.

Penas, D. R., Banga, J. R., Gonzalez, P., & Doallo, R. (2015). Enhanced parallel differential evolution algorithm for problems in computational systems biology. *Applied Soft Computing, 33*, 86–99. http://doi.org/10.1016/j.asoc.2015.04.025.

Penas, D. R., Gonzalez, P., & Egea, J. A. (2017). Parameter estimation in large-scale systems biology models : A parallel and self-adaptive cooperative strategy. *BMC Bioinformatics, 18*, 52. http://doi.org/10.1186/s12859-016-1452-4.

Raue, A., Karlsson, J., Saccomani, M. P., Jirstrand, M., & Timmer, J. (2014). Comparison of approaches for parameter identifiability analysis of biological systems. *Bioinformatics, 30*(10), 1440–1448. http://doi.org/10.1093/bioinformatics/btu006.

Raue, A., Kreutz, C., Maiwald, T., Bachmann, J., Schilling, M., Klingmüller, U., et al. (2009). Structural and practical identifiability analysis of partially observed dynamical models by exploiting the profile likelihood. *Bioinformatics, 25*(15), 1923–1929. http://doi.org/10.1093/bioinformatics/btp358.

Remli, M. A., Deris, S., Mohamad, M. S., Omatu, S., & Corchado, J. M. (2017a). An enhanced scatter search with combined opposition-based learning for parameter estimation in large-scale kinetic models of biochemical systems. *Engineering Applications of Artificial Intelligence, 62*, 164–180. April http://doi.org/10.1016/j.engappai.2017.04.004 .

Remli, M. A., Mohamad, M. S., Deris, S., Napis, S., Sinnott, R., & Sjaugi, M. F. (2017b). Metaheuristic optimization for parameter estimation in kinetic models of biological systems - recent development and future direction. *Current Bioinformatics, 12*(00), 286–295. http://doi.org/10.2174/1574893611666161018142 8.

Riahi, V., Khorramizadeh, M., Hakim Newton, M. A., & Sattar, A. (2017). Scatter search for mixed blocking flowshop scheduling. *Expert Systems with Applications, 79*, 20–32. http://doi.org/10.1016/j.eswa.2017.02.027.

Robles-Rodriguez, C. E., Bideaux, C., Guillouet, S. E., Gorret, N., Cescut, J., Uribelarrea, J. L., et al. (2017). Dynamic metabolic modeling of lipid accumulation and citric acid production by Yarrowia lipolytica. *Computers and Chemical Engineering, 100*, 139–152. http://doi.org/10.1016/j.compchemeng.2017.02.013.

Saa, P. A., & Nielsen, L. K. (2016). Construction of feasible and accurate kinetic models of metabolism: A Bayesian approach. *Scientific Reports, 6*(1), 29635. http://doi.org/10.1038/srep29635.

Smallbone, K., & Mendes, P. (2013). Large-scale metabolic models: From reconstruction to differential equations. *Industrial Biotechnology, 9*(4), 179–184. http://doi.org/10.1089/ind.2013.0003.

Talbi, E. G., & Bachelet, V. (2006). COSEARCH: A parallel cooperative metaheuristic. *Journal of Mathematical Modelling and Algorithms, 5*(1), 5–22. http://doi.org/10.1007/s10852-005-9029-7.

Villaverde, A. F., Egea, J. A., & Banga, J. R. (2012). A cooperative strategy for parameter estimation in large scale systems biology models. *BMC Systems Biology, 6*, 75.

Villaverde, A. F., Henriques, D., Smallbone, K., Bongard, S., Schmid, J., Cicin-Sain, D., et al. (2015). BioPreDyn-bench: A suite of benchmark problems for dynamic modelling in systems biology. *BMC Systems Biology, 9*, 8. http://doi.org/10.1186/s12918-015-0144-4.

Zúñiga, E. C. T., López Cruz, I. L., & García, A. R. (2014). Parameter estimation for crop growth model using evolutionary and bio-inspired algorithms. *Applied Soft Computing, 23*, 474–482. http://doi.org/10.1016/j.asoc.2014.06.023.