Article

Estimation of Small-Scale Kinetic Parameters of *Escherichia coli* (*E. coli*) Model by Enhanced Segment Particle Swarm Optimization Algorithm ESe-PSO

Mohammed Adam Kunna Azrag [1,*], Jasni Mohamad Zain [1], Tuty Asmawaty Abdul Kadir [2], Marina Yusoff [1], Aqeel Sakhy Jaber [3], Hybat Salih Mohamed Abdlrhman [4], Yasmeen Hafiz Zaki Ahmed [4] and Mohamed Saad Bala Husain [4]

[1] Institute for Big Data Analytics & Artificial Intelligence (IBDAAI), Komplek AI-Khawarizmi, Universiti Teknologi Mara, Shah Alam 40450, Selangor, Malaysia
[2] Faculty of Computing, Universiti Malaysia Pahang, Pekan 26600, Pahang, Malaysia
[3] Department of Electrical Power Engineering Techniques, Al-Ma'moon University College, Baghdad 10013, Iraq
[4] Faculty of Chemical Engineering, Universiti Malaysia Pahang, Gambang 26300, Pahang, Malaysia
* Correspondence: adamkunna@uitm.edu.my

Abstract: The ability to create "structured models" of biological simulations is becoming more and more commonplace. Although computer simulations can be used to estimate the model, they are restricted by the lack of experimentally available parameter values, which must be approximated. In this study, an Enhanced Segment Particle Swarm Optimization (ESe-PSO) algorithm that can estimate the values of small-scale kinetic parameters is described and applied to *E. coli's* main metabolic network as a model system. The glycolysis, phosphotransferase system, pentose phosphate, the TCA cycle, gluconeogenesis, glyoxylate pathways, and acetate formation pathways of *Escherichia coli* are represented by the Differential Algebraic Equations (DAE) system for the metabolic network. However, this algorithm uses segments to organize particle movements and the dynamic inertia weight (ω) to increase the algorithm's exploration and exploitation potential. As an alternative to the state-of-the-art algorithm, this adjustment improves estimation accuracy. The numerical findings indicate a good agreement between the observed and predicted data. In this regard, the result of the ESe-PSO algorithm achieved superior accuracy compared with the Segment Particle Swarm Optimization (Se-PSO), Particle Swarm Optimization (PSO), Genetic Algorithm (GA), and Differential Evolution (DE) algorithms. As a result of this innovative approach, it was concluded that small-scale and even entire cell kinetic model parameters can be developed.

Keywords: kinetic parameters; simulation; estimation; algorithm; *E. coli*

check for updates

Citation: Azrag, M.A.K.; Zain, J.M.; Kadir, T.A.A.; Yusoff, M.; Jaber, A.S.; Abdlrhman, H.S.M.; Ahmed, Y.H.Z.; Husain, M.S.B. Estimation of Small-Scale Kinetic Parameters of *Escherichia coli* (*E. coli*) Model by Enhanced Segment Particle Swarm Optimization Algorithm ESe-PSO. *Processes* **2023**, *11*, 126. https://doi.org/10.3390/pr11010126

Academic Editor: Andrzej Kasperski

Received: 26 September 2022
Revised: 31 October 2022
Accepted: 13 November 2022
Published: 1 January 2023

1. Introduction

Computing simulations and optimization are important subjects in systems biology and bioinformatics, where they play an important role in mathematical approaches to the reverse engineering of biological systems and managing uncertainty in that context. The large amount of computation required for the simulation, calibration, and analysis of models has prompted a number of researchers to propose various parallelization schemes in an attempt to accelerate these activities [1]. The development of dynamic (kinetic) models at a smaller scale has been the focus of recent research, with the eventual goal of producing whole-cell models. Model calibration has gained a lot of interest, especially with reference to global optimization metaheuristics and hybrid approaches [2].

Thus, kinetic models are being developed so that the dynamics of biological processes can be described in a quantitative manner. These models consider the stoichiometry of reactions and the kinetic expressions associated with each enzyme. In vitro experiments are used to determine the kinetic properties of enzymes by exposing isolated enzymes to optimal conditions. In vitro experiments are used to determine the kinetic properties of enzymes by exposing isolated enzymes to optimal conditions. These conditions are not the same as those found around enzymes inside living cells; thus, when compared to in vivo-measured concentrations, the use of in vitro parameters in kinetic models can result in incorrect predictions of intracellular metabolite concentrations [3]. The dynamics of cell metabolism can be fine-tuned by perturbing a culture and measuring fluxes, enzyme levels, and intra- and extracellular metabolite concentrations as a function of time. Advances in experimental techniques have paved the way for developing dynamic models for metabolic networks that can estimate microbial behavior [4].

Various *Escherichia coli* (*E. coli*) models were created and simulated in order to better understand the model's behavior and produce specific products such as those reported in [5–9]. In [5], the researchers simulate and estimate the kinetic parameters of two pathways while ignoring the other pathways (gluconeogenesis and glyoxylate). In [6], glutamine/aspartate metabolism and fructose consumption are incorporated into the model by utilizing the (Pts) system. However, the model does not consider the entire pathway model or *E. coli* cell growth. The researchers in [7] generate the experimental time courses of extracellular glucose and biomass in the *E. coli* model but do not consider the overall production of the pathways. As reported in [8], Kotte used the Monod equations to simulate glucose uptake without estimating the specific growth rate based on a molecular process in *E. coli*; therein, he did not consider estimating the entire *E. coli* main metabolic pathway. Neglecting the entire main metabolic pathway may result in an incorrect prediction of the simulation result. As a result, small-scale kinetic parameters must be comprehensively investigated. The study of a complete model can only be comprehensively achieved by including the entire cell system. This is because the feedback loop of several metabolites, such as PEP, OAA, PYR, and others, may affect other enzymes in the main metabolic pathway, causing the concentration of other metabolites to change dynamically over time [10,11].

Recently, three different strategies were compared for estimating the kinetic parameters of a dynamic model of central carbon metabolism in *Escherichia coli*, i.e., the modified simplex method [12,13], simulated annealing, and differential evolution. According to the authors, differential evolution produced the best results. Moreover, the researchers in [14] revised the central carbon metabolism model and estimated kinetic parameters for the glycolytic enzymes from [5]. The parameter estimation problem was solved using MATLAB and a weighted least squares objective function.

As a result, researchers are increasingly using metaheuristic optimization [15] algorithm methods to estimate the kinetic parameters of the *E. coli* model and other biological models because of the difficulties in calculating kinetic parameters. Several of these metaheuristic methods have been utilized to estimate the kinetic parameters utilized in [16–19], and some of these algorithms used experimental data derived from [5,20]. Furthermore, there are hundreds or even thousands of parameters in biological kinetic models that make parameter estimates difficult because of the large search space that must be investigated. High-dimensional kinetic models (with hundreds or thousands of different kinetic parameters) are difficult to compute and, as a result, the above algorithms' performance is negatively affected, resulting in reduced accuracy [21].

Differential Evolution (DE) is an example of a method that is commonly used and explored for parameter estimation in metabolic models. The DE method's key shortcoming is its time consumption, which makes it difficult for the DE algorithm to set its parameters when a high number of processors and numerous local searches are involved. Evolutionary algorithms, such as the Genetic Algorithm (GA), share many of the same principles as DE. This methodology is often used to estimate metabolic model parameters. When compared to DE, Particle Swarm Optimization (PSO), and other methods, this algorithm's key flaw is the time it takes to compute [22,23].

Many other fields, including those referenced in [24–26], have used the PSO algorithm since its inception in 1995. Birds and fish finding their food were modeled using the method described in [27]. They share information with each other as they proceed through the search process, and then hunt for their destination randomly and autonomously. Therefore, the search space is filled with different courses and directions for the particles to follow.

Additionally, the Enhanced Segment Particle Swarm Optimization (ESe-PSO) method was conceived and developed based on the Segment Particle Swarm Optimization (Se-PSO) algorithm with the aim of performing deep searches while maintaining the accuracy of Se-PSO. This progression is predicated on the understanding of Se-PSO's local and global point initialization [2]. Segmentation separates the particles into groups in this scenario, allowing them to work together toward the ideal solution. This approach was modified to identify small-scale kinetic parameters in an *E. coli* model [9] and governor–turbine models in a single-area power plant [28,29]. As a result of the linearity of the model's linear inertia weight (ω) [28], the inertia weight has an effect on the particles' exploration and exploitation. As a result, extensive exploration is required at the start and minimal exploitation near the end of the algorithm's execution. This is performed to avoid the local optima trap and thus increase the efficiency of estimating kinetic parameters with the goal of minimizing model distances in a reasonable amount of time.

In this regard, the model under study [9] was formulated to simulate the main metabolic pathway of *E. coli*. This model has six pathways, including glycolysis, pentose phosphate, the TCA cycle, gluconeogenesis, and glyoxylate, in addition to acetate formation. This model was chosen in this study rather than the aforementioned models due to its ability to simulate the main metabolic pathways of *E. coli* with small-scale kinetic parameters. Moreover, as a result of the lack of real experimental data, the nonlinearity of the model, and the small-scale kinetic parameters, the model response metabolites must be investigated depending on estimating small-scale kinetic parameters [30]. The experimental dataset used in this study [20] consists of many metabolites and is used in many studies for kinetic parameters, such as [16,17,21,31]. The purpose of this work was to adopt the ESe-PSO algorithm to estimate small-scale kinetic parameters. The sensitive kinetics were obtained for estimation purposes [4]. The remainder of the paper is organized as follows. The introduction is presented in Section 1. The problem statement is outlined in Section 2. The approach is described in Section 3. Section 4 discusses the outcome. In the final section, the conclusion is summarized. As a result, we believe that this novel technique can help to estimate small-scale dynamic models in systems biology.

2. Problem Formulation

Kinetic metabolic models are part of the enzymatic equation for ODE functions. The models' behavior and procedure design can be hampered by erroneous metabolic kinetic models. As a result of the nonlinearity of the model, obtaining accurate results is a challenging task because the kinetics are often gathered from multiple laboratories and under differing conditions [32–34].

In this regard, the main kinetic metabolic model of *E. coli* simulated in [9] contains large-kinetic parameters distributed across six pathways. This model had a great impact on *E. coli* model simulations in terms of understanding and simulating its behavior. However, the researchers stated that the model requires further investigations focused on its kinetic parameter estimations and responses and that further comparisons with real experimental data with small-scale kinetic parameters are necessary. Thus, further study to this end is of significant importance. However, the process of parameter estimation involves looking for the parameter values in a mathematical model (formulated using ODE) that best suit the experimental data. This can be accomplished by minimizing the scalar distance between the model prediction and experimental data, considering experimental errors. This problem can be divided into three categories: multimodal, continuous, and single-objective optimization. The objective function of kinetic parameter estimation considered in this work is described as follows:

$$f = \left| \left(y^{r,1} - y^{s,1}\right) + \left(y^{r,2} - y^{s,2}\right) + \ldots + \left(y^{r,m} - y^{s,m}\right) \right| \tag{1}$$

where f is the objective function and $y^{r,m}$ is the actual model output r resulting from m metabolites. The $y^{s,m}$ term is the simulation model output s for m metabolites.

Because biological problems are nonlinear, estimating the best parameters for this problem is difficult because many local minima exist. Most optimization algorithms become easily trapped in local minima as a result, resulting in a slow convergence speed.

Furthermore, the kinetic parameters that need to be estimated using the methods in Section 3 are implemented.

3. Materials and Methods

This section describes the model structure and the algorithms of DE, GA, PSO, Se-PSO, and Ese-PSO in order to estimate small-kinetic parameters.

3.1. *The Structure of the* E. coli *Kinetic Model*

The dynamic model of the main metabolic pathway of *E. coli* formulated in [9] was used as a benchmark and described in Figure 1. This model, which consists of glycolysis, pentose phosphate, the TCA cycle, gluconeogenesis, and glyoxylate pathways, in addition to acetate formation with the phosphotransferase system. This model has 23 metabolites, 28 enzymatic reactions, 10 co-factors (e.g., adenosine triphosphate (ATP), coenzyme A (COA), nicotinamide adenine dinucleotide phosphate (NADPH)), and 172 kinetic parameters. Equation (2) gives the rate at which the concentration of the metabolite in the considered model changes.

$$\frac{dC_i}{dt} = \sum_j R_{ij} v_j - \mu C_i \tag{2}$$

where C_i is the concentration of metabolite i, R_{ij} is the stoichiometric coefficient of metabolite i in the reaction j, v_j is the rate of the reaction j, and C_i is the growth rate on the dilution effect $\mu = 0.1\ h^{-1}$ due to the increase in cell volume as the cell grows [9].

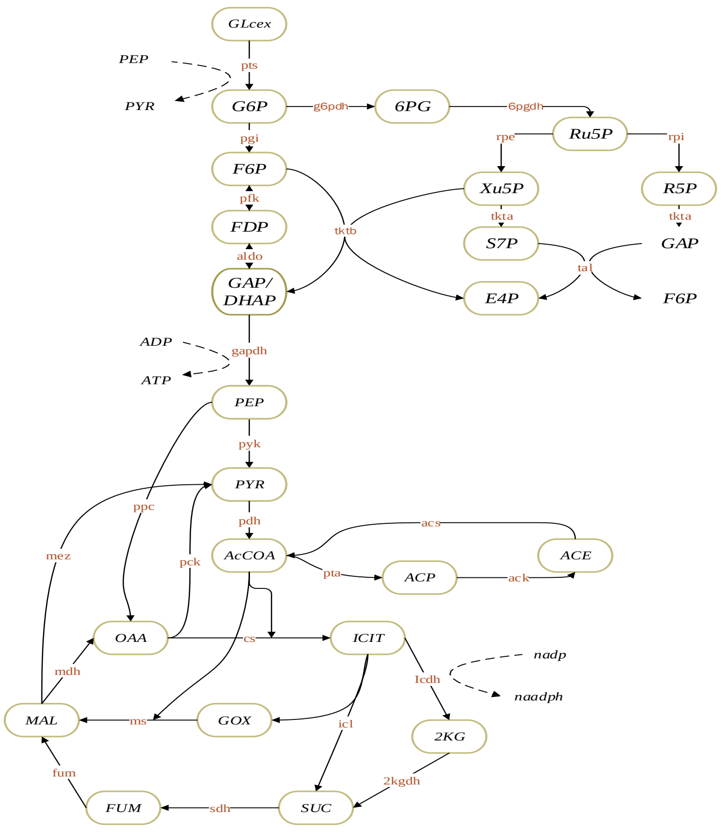

Figure 1. The main metabolic model of *E. coli*.

The model mass balance equations in Table 1 and the kinetic rate equations in Table 2 for the investigated model that was described in Figure 1 are all listed below.

Table 1. The mass balance equation.

Metabolites	**Mass Balance Description**
Cell (X)	$\frac{d[X]}{dt} = \mu[X]$
Extra Glucose (GLC^{ex})	$\frac{d[GLC^{ex}]}{dt} = -v_{PTS}[X]$
Glucose-6-phosphate ($G6P$)	$\frac{d[G6P]}{dt} = v_{PTS} - v_{PGI} - v_{G6PDH} - \mu[G6P]$
Fructose 6-phosphate ($F6P$)	$\frac{d[F6P]}{dt} = v_{PGI} - v_{PFK} + v_{TKTB} + v_{TAL} - \mu[F6P]$
Fructose 1,6-Phosphate (FDP)	$\frac{d[FDP]}{dt} = v_{PFK} - v_{ALDO} - \mu[FDP]$
Glyceraldehyde 3-phosphate (GAP)	$\frac{d[GAP]}{dt} = 2v_{ALDO} - v_{GAPDH} + v_{TKTA} + v_{TKTB} - v_{TAL} - \mu[GAP]$
Phosphoenol-pyruvate (PEP)	$\frac{d[PEP]}{dt} = v_{GAPDH} + v_{PCK} - v_{PTS} - v_{PYK} - v_{PPC} - \mu[PEP]$
Pyruvate (PYR)	$\frac{d[PYR]}{dt} = v_{PYK} + v_{PTS} + v_{MEZ} - v_{PDH} - \mu[PYR]$
Acetyl-CoA ($AcCoA$)	$\frac{d[AcCoA]}{dt} = v_{PDH} + v_{ACS} + v_{CS} - v_{PTA} - \mu[AcCoA]$
Isocitrate ($ICIT$)	$\frac{d[ICIT]}{dt} = v_{CS} - v_{ICDH} - v_{ICL} - \mu[ICIT]$
2-Keto-D-gluconate ($2KG$)	$\frac{d[2KG]}{dt} = v_{ICDH} - v_{2KGDH} - \mu[2KG]$
Succinate (SUC)	$\frac{d[SUC]}{dt} = v_{2KGDH} + v_{ICL} - v_{SDH} - \mu[SUC]$
Fumarate (FUM)	$\frac{d[FUM]}{dt} = v_{SDH} - v_{FUM} - \mu[FUM]$
Malate (MAL)	$\frac{d[MAL]}{dt} = v_{FUM} + v_{MS} - v_{MDH} - v_{MEZ} - \mu[MAL]$
Oxaloacetate (OAA)	$\frac{d[OAA]}{dt} = v_{MDH} + v_{PPC} - v_{CS} - v_{PCK} - \mu[OAA]$
Glyoxylate (GOX)	$\frac{d[GOX]}{dt} = v_{ICL} - v_{MS} - \mu[GOX]$
Acetyl phosphate (ACP)	$\frac{d[ACP]}{dt} = v_{PTA} - v_{ACK} - \mu[ACP]$
Acetate (ACE^{ex})	$\frac{d[ACE^{ex}]}{dt} = (v_{ACK} - v_{ACS})\mu[X]$
6-Phosphogluconolactone ($6PG$)	$\frac{d[6PG]}{dt} = v_{G6PDH} - v_{6PGDH} - \mu[6PG]$
Ribose 5-phosphate ($Ru5P$)	$\frac{d[RU5P]}{dt} = v_{6PGDH} - v_{RPE} - v_{RPI} - \mu[Ru5P]$
Ribulose 5-phosphoenolpyruvate ($R5P$)	$\frac{d[R5P]}{dt} = v_{RPI} - v_{TKTA} - \mu[R5P]$
Xylulose 5-phosphate ($Xu5P$)	$\frac{d[Xu5P]}{dt} = v_{RPE} - v_{TKTA} - v_{TKTB} - \mu[Xu5P]$
Sedoheptulose 7-phosphate ($S7P$)	$\frac{d[S7P]}{dt} = v_{TKTA} - v_{TAL} - \mu[S7P]$
Erythrose 4-phosphate ($E4P$)	$\frac{d[E4P]}{dt} = v_{TAL} - v_{TKTB} - \mu[E4P]$

Table 2. The kinetic rate equations.

Reactions	**Kinetic Equation**
Cell growth (X)	$\begin{cases} \mu_m\left(1-\frac{[X]}{X_m}\right)\left(\frac{[GLc^{ex}]}{K_s+[GLc^{ex}]}\right)k_{ATP}v_{ATP}(.), ([GLc^{ex}]>0) \\ \frac{\mu_{mA}[Ace^{ex}]}{K_{sA}+[Ace^{ex}]}\ k_{ATP}v_{ATP}(.), ([GLc^{ex}] \leq 1\ and [Ace^{ex}]>0) \end{cases}$
Phosphotransferase systems (PTS)	$\frac{v_{PTS}^{max}[GLc^{ex}]\frac{[PEP]}{PYR}}{\left(K_{a1}+K_{a2}\frac{[PEP]}{[PYR]}+K_{a3}[GLc^{ex}]+[GLc^{ex}]\frac{[PEP]}{[PYR]}\right)\left(1+\frac{[G6P]^n G6P}{K_{G6P}}\right)}$
Phosphoglucose isomerase/phosphoglucoisomerase (PGI)	$\frac{v_{PGI}^{max}\left([G6P]-\frac{[F6P]}{K_{eq}}\right)}{K_{G6P}\left(1+\frac{[F6P]}{K_{F6P}\left(1+\frac{[F6P]}{K_{6pginh}^{F6P}}\right)}+\frac{[6PG]}{K_{6pginh}^{G6P}}\right)+G6P}$

Table 2. *Cont.*

Reactions	Kinetic Equation
Phosphofructokinase (PFK)	$\frac{v_{PFK}^{max} K_{ATP}\ [F6P]}{K_{(ATP\ ,ADP)}\left([F6P]+K_z^{F6P}\frac{K_{b(ADP,AMP)+\frac{[PEP]}{K_{PEP}}}}{K_{a(ADP,AMP)}}\right)\left(1+\frac{L_{pfk}}{\left(1+[F6P]\left(\frac{K_{a(ADP,AMP)}}{K_s^{F6P}\left(K_{b(ADP,AMP)}+\frac{[PEP]}{K_{PEP}}\right)}\right)\right)^{n_{PFK}}}\right)}$
Aldolase (Aldo)	$\frac{v_{ALDO}^{max}\left([FDP]-\frac{[DHAP][GAP]}{K_{eq}}\right)}{\left(K_{FDP}+[FDP]+\frac{K_{GAP}[DAHP]}{\left[K_{eq}\ V_{bif}\right]}+\frac{K_{DHAP}[GAP]}{\left[K_{eq}\ V_{bif}\right]}+\frac{[FDP][GAP]}{K_{inh}^{PEP}}+\frac{[DHAP][GAP]}{K_{eq}V_{bif}}\right)}$
Glyceraldehyde 3-phosphate dehydrogenase (GAPDH)	$\frac{v_{GAPDH}^{max}\left([GAP]-\frac{[PEP][NADH]}{K_{eq}[NAD]}\right)}{\left(K_{GAP}\left(1+\frac{[PEP]}{K_{PGP}}\right)+[GAP]\right)\left(\frac{K_{NAD}}{NAD}\left(1+\frac{[NADH]}{K_{NADH}}\right)+1\right)}$
Pyruvate kinase (PYK)	$\frac{v_{PYK}^{max}\ [PEP]\left(\frac{PEP}{K_{PEP}}+1\right)^{n_{pyk}-1}[ADP]}{K_{PEP}\left(L_{PYK}\left(\frac{1+\frac{[ATP]}{K_{ATP}}}{\frac{[FDP]}{K_{FDP}}+\frac{[AMP]}{K_{AMP}}+1}\right)^{n_{pyk}}+\left(\frac{[PEP]}{K_{PEP}}+1\right)^{n_{pyk}}\right)\ ([ADP]+K_{ADP})}$
Phosphoenolpyruvate carboxylase (Ppc)	$\frac{K_1+K_2[AcCOA]+K_3[FDP]+K_4[AcCOA][FDP]}{1+K_5[AcCOA]+K_6[FDP]}\left(\frac{[PEP]}{K_m+[PEP]}\right)$
Glucose-6-phosphate dehydrogenase (G6PDH)	$\frac{v_{G6PDH}^{max}[G6P][NADP]}{\left([G6P]+K_{g6p}\right)\left(1+\frac{[NADPH]}{K_{ndph}}\right)\left(K_{nadp}\left(1+\frac{[NADPH]}{K_{nadph}}\right)+NADP\right)}$
Hydroxyprostaglandin dehydrogenase (PGDH)	$\frac{v_{PGDH}^{max}[6PG][NADP]}{\left([6PG]+K_{6pg}\right)\left([NADP]+K_{nadp}\left(1+\frac{[NADPH]}{K_{nadph}}\right)\left(1+\frac{[ATP]}{K_{atp}}\right)\right)}$
Ribulose-phosphate 3-epimerase (Rpe)	$v_{Rpe}^{max}\left([Ru5P]-\frac{[R5P]}{K_{eq}^{Rpe}}\right)$
Ribose-5-phosphate isomerase (Rpi)	$v_{Rpi}^{max}\left([Ru5P]-\frac{[R5P]}{K_{eq}^{Rpi}}\right)$
Transketolase 1 (TktA)	$v_{TKtA}^{max}\left([R5P][Xu5P]-\frac{[S7P][GAP]}{K_{eq}^{TKtA}}\right)$
Transketolase 2 (TktB)	$v_{TKtB}^{max}\left([Xu5P][E4]-\frac{[F6P][GAP]}{K_{eq}^{TKtB}}\right)$
Tyrosine ammonia lyase (Tal)	$v_{TaL}^{max}\left([GAP][S7P]-\frac{[E4P][F6P]}{K_{eq}^{TKtB}}\right)$
Phosphoenolpyruvate carboxykinase (PcK)	$v_{PcK}^{max}\left(\frac{[OAA]\frac{[ATP]}{[ADP]}}{K_m^{OAA}\frac{[ATP]}{[ADP]}+[OAA]\frac{[ATP]}{[ADP]}+\frac{K_i^{ATP}K_m^{OAA}}{K_i^{ADP}}+\frac{K_i^{ATP}K_m^{OAA}}{K_m^{PEP}K_i^{ADP}}[PEP]+\frac{K_i^{ATP}K_m^{OAA}}{K_i^{PEP}K_l^{ATP}}\ \frac{[ATP][PEP]}{[ADP]}+\frac{K_i^{ATP}K_m^{OAA}}{K_i^{ADP}K_l^{OAA}}[OAA]}\right)$
Pyruvate dehydrogenase (PDH)	$\frac{\frac{v_{PDH}^{max}}{[NAD]}\left(\frac{1}{1+K_i\frac{[NADH]}{[NAD]}}\right)\left(\frac{[PYR]}{K_m^{PYR}}\right)\left(\frac{1}{K_m^{NAD}}\right)\left(\frac{[COA]}{K_m^{COA}}\right)}{\left(1+\frac{[PYR]}{K_m^{PYR}}\right)\left(\frac{1}{NAD}+\frac{1}{K_m^{NAD}}+\frac{[NADH]}{K_m^{NADH}[NAD]}\right)\left(1+\frac{[COA]}{K_m^{COA}}+\frac{[AcCOA]}{K_m^{AcCOA}}\right)}$
Phosphate acetyltransferase (Pta)	$\frac{v_{Pta}^{max}\left(\frac{1}{K_i^{AcCOA}K_m^P}\right)\left([AcCoA][P]-\frac{[AcP][CoA]}{K_{eq}}\right)}{\left(1+\frac{[AcCoA]}{K_i^{AcCoA}}+\frac{[P]}{K_i^P}+\frac{[ACP]}{K_i^{ACP}}+\frac{[CoA]}{K_i^{CoA}}+\left(\frac{[AcCoA][P]}{K_i^{AcCoA}K_m^P}\right)+\left(\frac{[AcP][CoA]}{K_m^{ACP}K_i^{CoA}}\right)\right)}$
(Acetate Kinase) (Ack)	$\frac{v_{Ack}^{max}\left(\frac{1}{K_m^{ADP}K_m^{ACP}}\right)\left([AcP][ADP]-\frac{[ACE][ATP]}{K_{eq}}\right)}{\left(1+\frac{[Acp]}{K_m^{AcP}}+\frac{[ACE]}{K_m^{ACE}}\right)\left(1+\frac{[ADP]}{K_m^{ADP}}+\frac{[ATP]}{K_m^{ATP}}\right)}$
Acetyl-CoA synthetase (Acs)	$\frac{v_{Acs}^{max}[ACE][NADP]}{(K_m+[ACE])\left(K_{eq}+[NADP]\right)}$
Citrate synthase (Cs)	$\frac{v_{CS}^{max}[AcCoA][OAA]}{\left(K_d^{AcCoA}K_m^{OAA}+K_m^{AcCoA}[OAA]\right)+\left([AcCoA]K_m^{OAA}\left(1+\frac{[NADH]}{K_{i1}^{NADH}}\right)\right)+\left([AcCoA][OAA]\left(1+\frac{[NADH]}{K_{i2}^{NDAH}}\right)\right)}$
Isocitrate dehydrogenase (ICDH)	$\frac{[ICDH]\frac{K_f}{K_m^{ICIT}}K_d^{NADP}\left([ICIT]-\frac{[NADH][2KG]}{K_{eq}^{ICDH}[NADP]}\right)}{\left(\frac{1}{[NADP]}+\frac{[ICIT]K_m^{NADP}}{K_m^{ICIT}K_d^{NADP}[NADP]}+\frac{1}{K_d^{NADP}}+\frac{[ICIT]}{K_m^{ICIT}K_d^{NADP}}+\frac{[ICIT]}{K_d^{ICIT}[NADP]}\frac{[NADPH]K_m^{NADP}}{K_m^{ICIT}K_m^{NADP}K_{einh}^{NADPH}}+\frac{[NADPH]K_{eknh}^{2KG}}{K_m^{2KG}K_{enhe}^{NADPH}[NADP]}+\frac{[2KG]K_m^{NADPH}}{K_m^{2KG}K_{enhe}^{NADPH}[NADP]}+\frac{[2KG]}{K_m^{2KG}}\frac{[NADPH]}{K_{enhe}^{NADPH}[NADP]}+\frac{[2KG]K_m^{NADPH}}{K_m^{2KG}K_m^{NADPH}}\frac{[NADPH]}{K_{ekn}^{NADP}[NADP]}\right)}$

Table 2. *Cont.*

Reactions	Kinetic Equation
Isocitrate lyase (IcL)	$\frac{v_{lcl-f}^{max}\frac{[ICIT]}{K_m^{ICIT}}}{\left(1+\frac{[ICIT]}{K_m^{ICIT}}+\frac{[SUC]}{K_m^{SUC}}+\frac{[PEP]}{K_m^{PEP}}+\frac{[2KG]}{K_m^{2KG}}+\frac{1}{K_l}\right)}$
Malate synthase (MS)	$\frac{v_{MS}^{max}\ \frac{[GOX]}{K_m^{GOX}}\ \frac{[AcCoA]}{K_m^{AcCoA}}-v_{MS}^{max}\ \frac{[MAL]}{K_m^{MAL}}}{\left(1+\frac{[GOX]}{K_m^{GOX}}+\frac{[MAL]}{K_m^{MAL}}+\left(1+\ \frac{[AcCoA]}{K_m^{AcCoA}}\right)\right)}$
2-Ketoglutarate (2KG)	$\frac{v_{2KGDH}^{max}\,[aKG][CoA]}{\left(\frac{K_m^{NAD}\,[aKG][CoA]}{[NAD]}+K_m^{CoA}\,[aKG]+K_m^{2KG}[CoA]+[aKG][CoA]+\frac{K_m^{2KG}K_z[aKG][SUC][NADH]}{K_1^{2KG}K_1^{SUC}[NAD]}\ \frac{K_m^{2KG}\,K_z[SUC][NADH]}{K_1^{SUC}\,[NAD]}+\frac{K_m^{NAD}\,[aKG][CoA][NADH]}{K_1^{NADH}[NAD]}+\frac{K_m^{CoA}\,[aK][SUC]}{K_1^{SUC}}\right)}$
Succinate dehydrogenase (SDH)	$\frac{v_{SDH1}v_{SDH2}\left([SUC]-\frac{[FUM]}{K_{eq}}\right)}{K_m^{SUC}v_{SDH2}+v_{SDH2}[SUC]+\frac{v_{SDH1}[FUM]}{K_{eq}}}$
Fumarate hydratase (Fum)	$\frac{v_{Fum1}v_{Fum2}\left([FUM]-\frac{[MAL]}{k_{Fum\ eq}}\right)}{K_m^{Fum}v_{Fum1}+v_{Fum2}[FUM]+\frac{V_{Fum1}[MAL]}{K_{eq}}}$
Malic enzyme (Mez)	$\frac{v_{Mez}^{max}\ [MAL]\ [NADP]}{(K_{MAL}+[MAL])\ \left(K_{eq}+[NADP]\right)}$
Malate dehydrogenase (MDH)	$\frac{v_{MDH1}v_{MDH2}\left([MAL]-\frac{[OAA]}{K_{eq}}\right)}{\left(\frac{K_1^{NAD}K_m^{MAL}v_{MDH2}}{[NAD]}+K_m^{MAL}\ v_{MDH2}+\frac{K_m^{NAD}v_{MDH2}[MAL]}{[NAD]}+v_{MDH2}[MAL]+\frac{K_m^{OAA}v_{MDH1}[NADH]}{K_{eq}[NAD]}+\frac{K_m^{NADH}v_{MDH1}[OAA]}{K_{eq}[NAD]}+\frac{v_{MDH1}[NADH][OAA]}{K_{eq}[NAD]}+\frac{v_{MDH1}K_m^{OAA}[NADH]}{K_{eq}K_1^{NAD}}+\frac{v_{MDH2}K_m^{NAD}[MAL][OAA]}{K_1^{OAA}[NAD]}+\frac{v_{MDH2}[MAL][NADH]}{K_1^{NAD}}+\frac{v_{MDH1}[MAL][NADH][OAA]}{K_{eq}K_1^{MAL}[NAD]}+\frac{v_{MDH2}[MAL][OAA]}{K_{II}^{OAA}}+\frac{v_{MDH1}[NADH][OAA]}{K_{II}^{NAD}K_{eq}}+\frac{K_1^{NAD}v_{MDH2}[MAL][NADH][OAA]}{K_{II}^{NAD}K_m^{OAA}K_1^{NADH}}\right)}$

After the model structure is described, the algorithms used to estimate the kinetics are stated below.

3.2. GA Algorithm

One of the most well-known heuristic search algorithms is the Genetic Algorithm, which is based on the process of natural evolution. In recent decades, GA has received a lot of attention in engineering design optimization. GA was first introduced in computer science in the 1960s, when a group of biologists attempted to implement the process of evolution in nature in computer code [35]. GA refers to any population-based algorithm that finds the best solution by using selection, crossover, and mutation across chromosomes. A member of the population is referred to as a chromosome/genotype, which is a binary or real-valued string. Several types of GA have been introduced in optimization studies following Barricelli [35]. However, a given optimization problem can be simply defined in GA by following three main steps, which include [36]:

Initialization is the first step.

Step 1: generation;

Step 2: selection;

Step 3: stopping criteria.

The first step in the GA algorithm is the generation of initial chromosomes (genotypes) in step 0. In most practical problems, genotypes are generated at random, and the goodness of each chromosome is assessed using an objective function and the constraints that go with it.

Step 1: Introduce the generation operator and apply it to the current population to generate an intermediate one. In the first step, the initial and intermediate populations are the same (step 0). However, the imposition of the generation operator forms the intermediate population in subsequent iterations;

Step 2: To create the next population, crossover-mutation operators are applied to the results of Step 1. When the generator (operator) creates a chromosome by combining the properties of two parental chromosomes, the term crossover is used. However, the term mutation is used when a new chromosome is formed by making minor changes to the properties of a single parent chromosome [37].

The algorithm design is influenced by experience, model specifications, and the association of experimental results with various heuristic search algorithms. Thus, the algorithm used in this work is described in Algorithm 1 below.

Algorithm 1. The GA Adoption

```
1.  Generate an initial population with $m$ chromosomes at random;
2.  Define the kinetic boundaries;
3.  Determine the fitness, $f_{(m)}$, of each chromosome in $m$;
4.  Build the evolved population using the following criteria;
5.  Using proportional fitness selection for $m1$ and $m2$ chromosomes;
6.  Use the crossover function on $m1$ and $m2$ to create a new chromosome ($m3$);
7.  Use the mutation function on $m3$ to generate $\mathrm{m}'$;
8.  Include $m^{\backslash}$ in the next population;
9.  Replace the old population with the new population;
10. If the stopping criteria are not met, repeat step 2 of the procedure.
```

3.3. DE Algorithm

More than two decades ago, Storn and Price [38] presented Differential Evolution (DE), a novel optimization method designed to handle nondifferentiable, nonlinear, and multimodal objective functions. To meet this requirement, DE was designed as a stochastic parallel direct search method that borrows concepts from the broad class of evolutionary algorithms while requiring only a few easily chosen control parameters. Early experimental results show that DE outperforms other well-known evolutionary algorithms in terms of convergence [38,39].

The combination of randomly chosen vectors produces new individuals (vectors) in each population. In our context, this operation is known as mutation. The resulting vectors are then mixed with another predetermined vector—the target vector—in a process known as recombination. This operation produces the trial vector. If and only if the trial vector reduces the value of the objective function f, it is accepted for the next generation. This operation is known as selection. The following is a high-level description of the aforementioned operators (for one generation). Thus, the algorithm used in this work is described in Algorithm 2 below.

Algorithm 2. The DE Adoption

```
1. Initialization operation: Generate the initial individuals $x_i^0, i = 1, 2, \ldots NP$ in $S_0$.
   Determine the mutation probability $F$, the crossover probability CR, and the maximal
   number of generations $G_m$. Set the current generation $G = 0$;
2. Initialize the kinetic boundaries;
3. For each individual $x_i^G, i = 1, 2, \ldots NP$, perform steps 3–5 to produce the population for
   the next generation $G + 1$;
4. Mutation operation: a perturbed individual $x_i^{G+1}$ is generated as follows:
   $x_i^{\sim G+1} = x_{r_1}^G + F.(x_{r_2}^G - x_{r_3}^G);$
5. Crossover operation: the perturbed individual $x_i^{\sim G+1} = \left[x_{i1}^{\sim G+1}, x_{i2}^{\sim G+1}, \ldots, x_{in}^{\sim G+1}\right]$ and
   the current individual $x_i^G = [x_{i1}^G, x_{i2}^G, \ldots, x_{in}^G]$ are selected by a binomial distribution to
   perform the crossover operation to generate the offspring $x_i^{-G+1}$;
6. Evaluation operation: the offspring $x_i^{-G+1}$ competes one-to-one with its parent $x_i^G$;
7. $G = G + 1$;
8. Repeat steps 2–7 as long as the number of generations is smaller than the allowable
   maximum number $G_m$ and the best individual is not obtained.
```

3.4. PSO Algorithm

Kennedy, a social psychologist, and Eberhart, an electrical engineer, developed the idea of particle swarms to create computational intelligence by exploiting already existing natural interacting systems [40,41].

The PSO method was developed in the middle of the 1990s while attempting to mimic the elegant, well-choreographed motion of a flock of birds as part of a social cognition study looking into the idea of collective intelligence in biological populations. Soon after its creation, it was recognized as an evolutionary technique [42]. This technique has received extensive study and in-depth reflection due to its incredibly effective problem-solving capabilities in a variety of technical and scientific applications.

The earliest simulations [42] modelled flocks of birds foraging for grain and were driven by social behavior. This quickly became a potent optimization technique, called Particle Swarm Optimization [43,44] (PSO).

The collection of particles in the search space in the PSO algorithm aims to optimize a fitness function, similar to the movement of flocks of birds in the natural environment in search of food. The particles are randomly placed in the search space and their quality or fitness at that position is evaluated. Then, after a predetermined number of iterations, each particle moves to a new location that is a better fit than the previous one. With some random perturbations, this movement is based on the history of the particle's best and current locations with those of the best positions attained by other particles in the swarm. Thus, with a fixed number of particles working together, the swarm achieves the most optimal solution to the fitness function in the problem space in subsequent iterations [44,45]. The fitness or objective function in the PSO algorithm is a performance evaluation criterion that is dependent on the algorithm's application area. A performance criterion is typically defined by a mathematical formulation that quantifies system performance via a performance index. Thus, the algorithm used in this work is described in Algorithm 3 below.

Algorithm 3. The PSO Adoption

```
1.  Initialize particles of PSO;
2.  Initialize the parameters;
3.  Initialize the velocities and positions of the swarm;
4.  For each particle;
5.  Iteration $i : bird_{step} : i++$;
6.  Initialize the kinetics and their boundaries;
7.  Calculate the fitness using Equation (1);
8.  Find the particles with best fitness in the neighborhood;
9.  Calculate the velocity of each particles using this equation:
    $v_i\ (t+1) = wv_i(t) + c_1 r_1 (p_i(t) - x_i(t)) + c_2 r_2 (G_i(t) - x_i(t))$;
10. Update the position of each particle using this equation: $x_i(t+1) = x_i(t) + v_i\ (t+1)$;
11. If the fitness value > the best fitness the best fitness $p_i(t)$, then set the current values as the
    new $G_i(t)$;
12. Otherwise, modify steps 1 and 2 and repeat steps 3–11;
13. Print the best fitness $G_i(t)$ of each particle;
14. End.
15. End.
```

3.5. Se-PSO Algorithm

The Se-PSO algorithm is derived from a mix of segmentation and Particle Swarm Optimization (PSO) algorithms, wherein segmentation is utilized to identify the local and global optimal point problems of PSO. On the basis of the dimension, the segmentation can be divided into more than two groups. The concept of parameter segmentation is theoretically illustrated in Figure 2. Assuming that we have three kinetic parameters that are initialized with search space boundaries, parameters 1 and 2 are divided into two segments, whereas parameter 3 is divided into one segment. Then, on the basis of the objective function, a group of particles moving in the search space is initiated. Each search iteration displays the local optimum position in each segment parameter. This scenario is shown in Figure 2 [28].

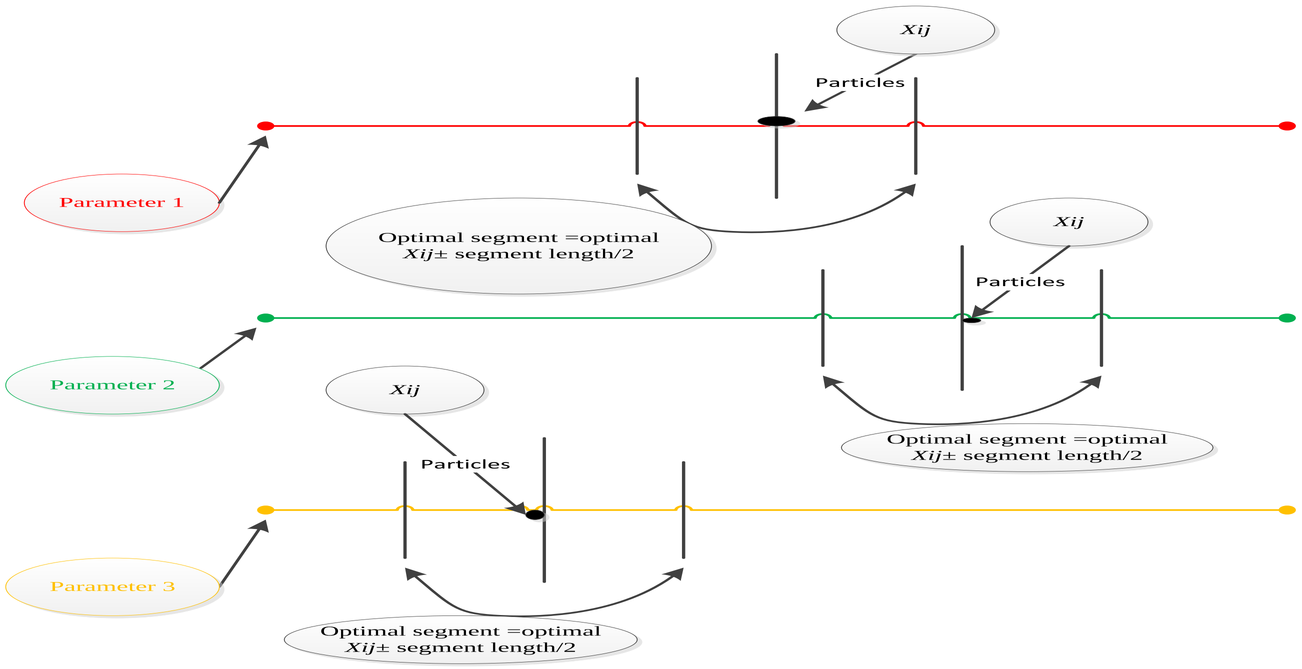
Xij
Particles
Parameter 1
Optimal segment =optimal Xij± segment length/2
Xij
Particles
Parameter 2
Optimal segment =optimal Xij± segment length/2
Xij
Particles
Parameter 3
Optimal segment =optimal Xij± segment length/2

Figure 2. The Se-PSO scenario.

The following parameters are set in this optimization identification case: the bird steps for PSO = 30, c_2 = 1.2, c_1 = 1.2, and ω = 0.9. According to the experimental results, it is better to initially set the inertia weight to a higher value and gradually reduce it to obtain refined solutions. To improve the algorithm, a new inertia weight is proposed that is set to the damping process as a linearly decreasing time-varying function.

3.6. ESe-PSO Algorithm

The primary parameters of the PSO algorithm (c_1, c_2, *and* ω) help the algorithm obtain the best possible result. When dealing with high-dimensional difficult situations, PSO suffers from early convergence to local optima [26]. As a result, the three parameters indicated above must be carefully calculated for a robust PSO [46]. The first is the learning factor that pulls particles toward their personal optimum position, while the second is the social learning component that pushes particles toward their global optimum position. It also remembers the prior velocity of the particles, preventing them from converging to local minima. Thus, in the basic PSO utilized in Se-PSO, the inertia weight (ω) was set (0.9) [28], indicating the necessity to adjust it in order to produce a better estimation in this work.

This approach was created using the fundamental PSO algorithm used in PSO segmentation. This advancement is related to the inertia weight (ω), which determines the influence that the last iteration speed has on the current speed and allows the particles to explore larger regions in the beginning and exploit neighboring areas at lower speeds in later stages. The development is carried out by initializing the inertia weight and damping parameter, which is determined throughout the iteration process to improve control convergence and enable the particle to search for a global solution.

According to the PSO algorithm, a particle with a higher fitness value is thought to be nearer to the global optimum than one with a lower fitness value. Stronger local exploration capabilities may be necessary for the particle with a higher fitness to seek through its immediate surroundings for the best solution. On the other hand, a particle with a lower fitness requires stronger global exploration capabilities to move fast to the particles with higher fitness. This improves the likelihood that the particle discovers the global optimum and speeds up the PSO's convergence. Lower inertia weights can be employed for particles with superior fitness, which helps to accelerate the PSO's convergence. Higher inertia weights are used for particles that are less fit and far from the optimal particle position, which can improve their capacity for global exploration and help these particles escape from local maxima [24].

The process of damping is subsequently executed before the end of the loop iteration process. This process supports the searching process by calculating the inertia weight after each iteration until the iteration is finished. This method supports the particles through control of convergence, thereby balancing the local and global search by increasing the exploration and exploitation of the particles to locate the optimal values [46,47]. When the inertia weight was initialized to the original interval value ω [1, 0.01], the damping value was set to $\omega damp$ = 0.99. Subsequently, the damping process $damp$ is calculated in a decreasing manner in the iteration loop, using Equations (3) and (4), until the iteration is finished:

$$damp = \omega * \omega damp \tag{3}$$

$$\omega = \left(\frac{\omega_{max} - \omega_{min}}{iter_{max} - iter_{min}} \right) + \omega_{min} \tag{4}$$

where $damp$ is the damping process, $\omega damp$ is the damping value, ω_{max} is the maximum value of $\omega = 1$, ω_{min} is the minimum value of $\omega = 0.01$, and $iter_{max}$ and $iter_{min}$ are the maximum and minimum iterations, respectively.

In this regard, this process explores wider areas in the beginning and exploits nearby areas in the later stages at a reduced speed. The changes are added to Se-PSO since this algorithm combines segmentation and particle swarm optimization based on the inertia weight (ω) modification in addition to segmentation. This modification helps the basic Se-PSO to more effectively reach the best optimum solution and increase exploration and exploitation. However, after the damping process is added, the velocity of Se-PSO [28] should be modified according to the damping changes as described in Equation (5):

$$v_i(t+1,\mathrm{j}) = \mathrm{damp}.v_i(t,j) + c_1 r_1 (p_i(t,j) - x_i(t,j)) + c_2 r_2 (G_i(t,j) - x(t,j)) \tag{5}$$

where $x_{t,j}$ is the position of particle i, $v_{t,j}$ is the velocity of particle i, t is the iteration of particles, j is the number of segments, ω is the inertia weight, $damp$ is a damping process, c_1 *and* c_2 are the acceleration coefficients, and r_1 *and* r_2 are random numbers between (0 and 1). The p_i term is the local optimum position of particle i, and G_i is the global optimum position of particle i.

After the ω is modified, the ESe-PSO algorithm proceeds according to the process illustrated by Algorithm 4 below:

Algorithm 4. The ESe-PSO Adoption

```
1.  Set Se-PSO parameters, the problem dimension, and the kinetic boundaries;
2.  Initialize the $v_{t,j}$ and $x_{t,j}$;
3.  Set the segment_number and the segment_length [2,28] for each kinetic parameter;
4.  For $k = 1$ to the number of the segment;
5.  Evaluate $f$;
6.  Select the best $G_i\ (t, j)$;
7.  Set iteration $i = 1$;
8.  Update the $\omega$, $damp$, $v_{t,j}$, $x_{t,j}$;
9.  Evaluate $f$;
10. If $f_{new} < f$;
11. Update the $G_i(t, j)$;
12. $x_{i,j}(t+1) = G_i(t, j)$;
13. If $f_{new} > f$, return to step 1 until the iteration $i = iteration$ is finished or a good solution is
    discovered. If $f_{new} < f$, then print $G_i(t, j)$;
14. Set $x_{i,j}(t+1) = G_i(t, j)$;
15. Set the optimal segment of each particle;
16. New optimal_segment $(k_p)$ = $current\ position\ (k_p)$;
17. Apply the PSO [11] Algorithm for the new optimal values.
```

The ESe-PSO adoption algorithm can be described based on the parameters and steps described in the above pseudocode. Specifically, all of the parameters for the ESe-PSO method and the problem's dimensions and kinetic boundaries are calculated and set. Then, according to the kinetic sensitivity, the number of segments to be created and the length of each segment for each kinetic are determined. Following that, starting with segment 1 and continuing until the number of segments is reached, the objective function is assessed, and then the global optimum position is selected for each segment in the evaluation. Thereafter, the iteration counter is set and the damping process *damp* is included, after which you should update the inertia weight ω, the velocity $v_{t,j}$, and the location $x_{t,j}$. Then, once the updating process has been completed, the fitness function f results are evaluated. A particle segment's global optimum position $G(t, j)$ is determined based on whether the new fitness function f_{new} is better than the current fitness f update. If, on the other hand, the new fitness function f_{new} is less than the present fitness function f, the algorithm returns to the beginning of its parameters and makes the necessary modifications. A second option is that the algorithm finds issues that are highly solvable, or that the iteration is completed. In this situation, the best particle segment is published from each particle segment on a global scale. Following that, the global best of each particle segment is used to determine the current position of each particle segment $G(t, j) = current\ position\ (k_p)$. The PSO algorithm then iterates through the possibilities until it finds the best solution.

4. Results

4.1. Algorithms Estimation Result

A comparative inertia weight is shown in Table 3. Five (5) different inertia weights were tested 30 times to calculate the best solution with the best fitness function (the lowest) updating the scheme (minimization) as described in [48–50]. The purpose here was to investigate and enhance the Se-PSO algorithm's ability with the inertia weight to increase accuracy. These inertia weights were selected because the Se-PSO algorithm uses a constant value and the particles face difficulties in exploration and exploitation since the problem is highly nonlinear and the inertia weight is constant. However, applying different inertia weight ω strategies can enhance the algorithm. Table 3 records the best five inertia weights ω in terms of accuracy and efficiency of the ESe-PSO algorithm when dealing with small-scale kinetic parameters estimation.

Figure 3 describes the 5 scenarios of the inertia weight (ω) to determine the best and worst solution before starting the estimation.

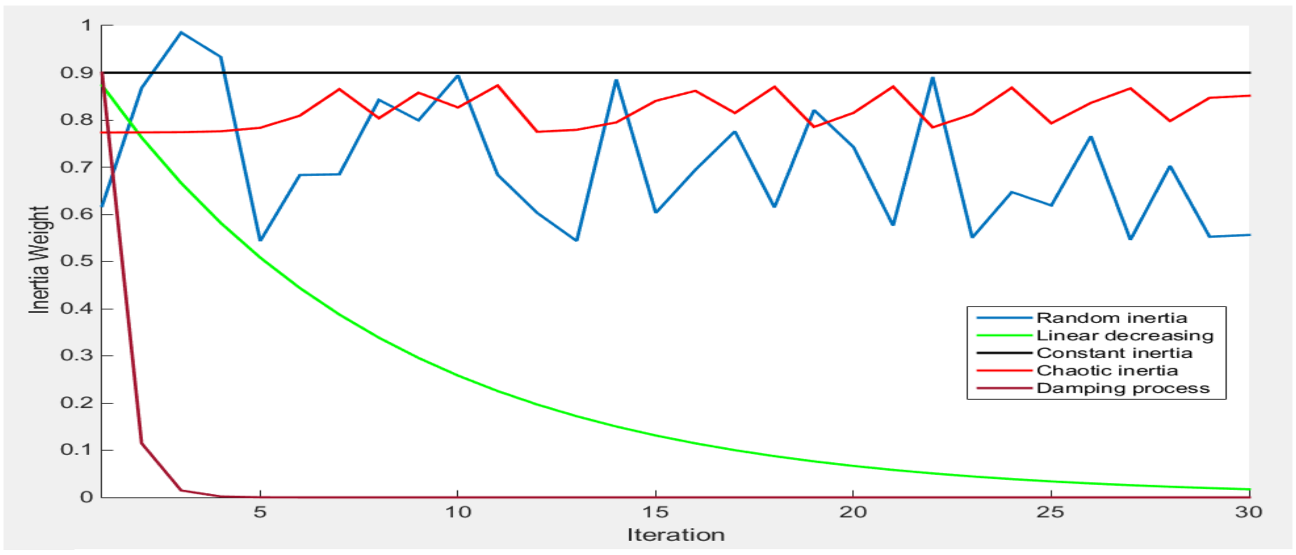

Figure 3. The inertia weight (ω) scenario.

Table 3. The inertia weight test.

Number	References	Name of the Inertia Weight	The Formula of Inertia Weight	Best Solution Function	Worst Solution Function
1	[26]	Random inertia weight	$\omega = 0.5 + \frac{rand()}{2}$ $rand$ is a random number between 0 and 1.	0.051	0.9
2	[38]	Linear decreasing inertia weight	$\omega = \left(\frac{(\omega_{max} - \omega_{min}) * (iter_{max} - iter_{min})}{iter_{max}}\right) + \omega_{min}$ where ω_{min} is the minimum inertia weight = 0.4, ω_{max} is the maximum inertia weight = 0.9, and $iter_{max,\ min}$ is the maximum and minimum iterations.	0.023	0.07
3	[27]	Constant inertia weight	$\omega = 0.9$	0.04	0.8
4	[39]	The chaotic inertia weight	$\omega = (\omega_1 - \omega_2) * \frac{iter_{max} - iter_{min}}{iter_{max}} + \omega_2 * z$ $z = 4 * z * (1 - z)$ where z is interval number (0, 1), $\omega_{1,2}$ are the maximum and minimum inertia weight, and $iter_{max,\ min}$ are the maximum and minimum iterations.	0.031	0.09
5	ESe-PSO	Damping process	$damp = \left(\left(\frac{\omega_{max} - \omega_{min}}{iter_{max} - iter_{min}}\right) + \omega_{min}\right) * \omega damp$ $\omega_{max} = 1$ is the maximum inertia weight, $\omega_{min} = 0.01$ is the minimum inertia weight, $iter_{max,\ min}$ is the maximum and minimum iterations, and $\omega damp = 0.99$ is the damping process value.	0.021	0.06

Note: The shaded cells represent the best and lower objective function.

Estimation of small-scale kinetics needs a sensitivity analysis to explore the kinetics' efficacy. However, the sensitivity analysis shows how many outputs are affected by the changes made to each kinetic parameter in order to identify the most sensitive kinetics. The sensitive kinetic parameters of [4,10] were used for estimation purposes, where the sensitivity percentage efficacy is shown in Table 4. It is difficult to optimize 172 kinetics at the same time due to the small-scale kinetic parameters. As a result, the sensitivity analysis was used to identify the most sensitive parameters and reduce estimation costs. All of the kinetics were tested up to a 200% perturbation. Furthermore, during the simulation, 7 kinetic parameters were identified to be the most effective kinetics out of a total of 172 and were designated as the parameters that needed to be estimated, while the remaining kinetics were left at their original values because they had an efficacy of below 20% if the perturbation was increased to 200%. The kinetic parameters' segments were increased throughout the ESe-PSO execution by adding one segment to each kinetic to boost the likelihood of finding an accurate solution, which allowed the algorithm to search a broad space. Additionally, the kinetic limits, with their upper and lower values, were started with tiny increments. On the basis of the objective function, the optimal segment was determined, as shown in Table 5. Thereafter, after updating the inertia weight by adding the damping process, the method of updating the location, velocity, and the objective function follows that of the PSO algorithm. The best segment is then used as the new boundary, and PSO searches around it. The objective behind the damping process is to boost the particles' exploration and exploitation capabilities in order to find an optimal solution. The damping process has a value of 0.99, and the inertia weight is [1, 0.01].

Table 4. The kinetic parameter segments.

Kinetics	Number of Segments	Sensitivity Affection
v_{max}^{pyk}	2	39 (metabolites and enzymes) 73.58%
n_{pk}	3	44 (metabolites and enzymes) 83.09%
$icdh$	2	41 (metabolites and enzymes) 77.35%
k_{icdh}^{f}	2	42 (metabolites and enzymes) 79.13%
$k_{icdhnap}^{d}$	3	43 (metabolites and enzymes) 80.24%
$Kicdhnapm$	2	42 (metabolites and enzymes) 79.24%
v_{max}^{icl}	3	46 (metabolites and enzymes) 86.79%

Table 5. The kinetic parameters boundaries.

Kinetics	Original	Lower	Upper	Kinetic Estimation
v_{max}^{pyk}	1.08500	0.82000	1.5000	0.820000
n_{pk}	3.00000	2.30000	3.5000	3.402000
$icdh$	24.4210	23.5000	24.900	23.82500
k_{icdh}^{f}	289,800	289,799	289,800	289,799.9
$k_{icdhnadp}^{d}$	0.0060	0.0030	0.0500	0.027000
$k_{icdhnadp}^{m}$	0.0170	0.0060	0.0600	0.007350
v_{max}^{icl}	3.8315	3.3000	4.3000	3.956000

As a result, the damping changes its values in a decreasing manner in each iteration until the iteration is complete.

Notably, the ESe-PSO approach utilized in this work precisely minimizes the model response distance. Moreover, rather than the molecule itself, this reduction reduces just 15 metabolites. This is due to the addition of the Se-PSO algorithm's kinetic parameter segmentation and damping procedure. Tables 4 and 5 illustrate the segmentation of kinetic parameters and the predicted kinetic parameters.

The segmentation was recommended based on the sensitivity analysis results' effect on the kinetic parameters. Only two kinetic factors influence more than or equal to 43 metabolites and enzymes, accounting for more than 80% of the total. These were considered two segments as a result of the extremely important alterations to the model output. The others, on the other hand, were regarded as a single section. As a result, each segment was expanded by one in order to maximize the likelihood of discovering an optimal solution.

The estimation of the kinetic parameters functions to minimize the distance of the model responses from [9] and move it towards the experimental data from [20]. From Table 6, it is evident that the simulation concentration result is remarkably close to the real experimental data for the model under study measured by mM = 0.001 mole liter^(−1.0).

Table 6. The simulation results.

Metabolites	**Experimental Data**	**Model Data**	**Optimized Values PSO**	**Optimized Values Se-PSO**	**Optimized Values ESe-PSO**	**Optimized Values DE**	**Optimized Values GA**
Glc	0.0617 mM	0.12276 mM	0.16717 mM	0.20195 mM	0.0395 mM	0.1523 mM	0.192 mM
$G6P$	1.76 mM	0.20364 mM	0.18432 mM	0.17731 mM	0.98621 mM	0.1602 mM	0.2013 mM
$F6P$	0.42 mM	0.02132 mM	0.01967 mM	0.01906 mM	0.0942 mM	0.02013 mM	0.019 mM
$DHAP/GAP$	0.231 mM	0.31106 mM	0.02496 mM	0.23518 mM	0.23518 mM	0.26023 mM	0.2812 mM
FDP	0.67 mM	1.4645 mM	0.84333 mM	0.71747 mM	0.65053 mM	0.9523 mM	1.214 mM
PEP	1.04 mM	1.4917 mM	1.2159 mM	1.1587 mM	1.087 mM	1.21 mM	1.325 mM
PYR	1.71 mM	2.8101 mM	3.3025 mM	3.832 mM	2.3502 mM	3.025 mM	2.894 mM
$6PG$	0.96 mM	0.01785 mM	0.01828 mM	0.01881 mM	0.0932 mM	0.01925 mM	0.0181 mM
$Ru5P$	0.088 mM	0.02135 mM	0.02093 mM	0.02101 mM	0.0352 mM	0.01933 mM	0.0183 mM
$R5P$	0.243 mM	0.0762 mM	0.07443 mM	0.07457 mM	0.0835 mM	0.07023 mM	0.0752 mM
$E4P$	0.112 mM	0.02744 mM	0.02162 mM	0.02003 mM	0.0625 mM	0.02625 mM	0.0252 mM
$AcCoA$	0.145 mM	1.0015 mM	1.117 mM	1.2679 mM	0.9325 mM	1.1025 mM	1.081 mM
OAA	0.241 mM	0.02962 mM	0.02268 mM	0.01592 mM	0.0503 mM	0.02525 mM	0.0282 mM
$ICIT$	0.21 mM	0.2103 mM	0.01241 mM	0.00206 mM	0.1902 mM	0.01925 mM	0.038 mM
$2KG$	0.134 mM	5.3656 mM	4.2219 mM	2.8222 mM	2.6253 mM	4.335 mM	4.4241 mM
Ace	0.36 mM	0.00347 mM	0.00499 mM	0.00626 mM	0.0825 mM	0.00225 mM	0.00342 mM
Distance	0	57.16%	37.09%	26.29%	16.18%	33.9%	35.34%

Note: The shaded cells represent the best simulation result of each algorithm.

As presented in Table 6, using the experimental data from [20], the estimation result of the ESe-PSO algorithm improved as compared to the results from the Se-PSO and PSO algorithms. The estimated result is highlighted. In the ESe-PSO estimation results, 15 metabolites (GLc, $G6P$, $F6P$, $GHAP/GAP$, FDP, PEP, PYR, $6PG$, $Ru5P$, $R5P$, $E4P$, $AcCoA$, OAA, $2KG$, and Ace) were well-optimized with a minimized error of about 16.81%.

Only the $ICIT$ was not properly minimized. In the Se-PSO estimation results, seven (7) metabolites ($GHAP/GAP$, FDP, PEP, $6PG$, $Ru5P$, $2KG$, and Ace) were well-optimized and minimized, with an error of about 26.29%. In the PSO estimation results, seven (7) metabolites ($GHAP/GAP$, FDP, PEP, $6PG$, $Ru5P$ $2KG$, and Ace) were well-optimized and minimized, with errors of approximately 37.09%. In the DE estimation results, six (6) model outputs ($DHAP/GAP$, FDP, $6PG$, $Ru5P$, and $2KG$) were well-optimized, with a minimized distance of about 33.9%. In the GA estimation result, six (6) metabolites (GLc, $DHAP/GAP$, FDP, PEP, $6PG$, $Ru5P$, and $2KG$) were well-optimized, with a distance minimized to 35.34%. Therefore, it can be confirmed that the results were perfect when compared to the model under study, which has a distance of 57.16% [9].

This indicates that, for kinetic parameter estimation, all these algorithms could provide a decent result, but the adopted and enhanced algorithm produces a better estimation than the others in terms of accuracy. Some of the metabolites were not minimized to a slight degree due to the other pathways' involvement, the model complexity, and the lumping together of various metabolites [9].

A comparative experiment is shown in Table 7. The ESe-PSO achieved the best ($7.04 * 10^{-5}$ and $7.41 * 10^{-5}$) objective function mean as compared to the Se-PSO algorithm and the best mean (0.000603 and 0.00379), PSO (0.003893 and 0.00549), GA (0.11476 and 0.269007), DE (0.049185), and 0 and DE (0.049185 and 0.280478), respectively, for the Hoque dataset. Overall, the ESe-PSO can be adopted to effectively estimate small-scale kinetic parameters to obtain accurate and acceptable results.

Table 7. Comparative objective function results over 20 runs [19].

Methods	Mean	STD	Best	Lower
PSO	0.00549	0.018391	0.00057	0.030252
Se-PSO	0.000379	0.003526	0.000021	0.00758
ESe-PSO	7.41×10^{-5}	0.000338	3.7×10^{-6}	0.00052
DE	0.023385	0.280478	0.00024	0.50257
GA	0.114985	0.269007	0.0081	0.1472

Note: The shaded cells represent the MEAN, STD, and BEST and LOWER objective functions.

Notably, the ESe-PSO was superior to the original Se-PSO, PSO, and other state-of-the-art approaches in terms of distance minimization, and the smallest objective function's value produced appropriate fits to two sets of experimental data. This is because the ESe-PSO algorithm added a damping process to increase the exploration and exploitation of the search space to support the particle in finding a global optimum solution. This modification facilitates the accurate determination of the optimal solution. The inertia weight ω was adjusted to the maximum and minimum during the damping process.

However, as stated in [32,51], the mean, STD (standard deviation), distance minimization, and F-test can be calculated for the result accuracy. The STD is a well-known measurement of how broad the meanings of the values being distributed are. The distance minimization is used to see how much the algorithm moves the estimation closer to real experimental data. An F-test is any statistical test in which the test statistic has an F-distribution under the null hypothesis. It is most often used when comparing statistical models that have been fitted to a dataset, in order to identify the model that best fits the population from which the data was sampled. As a result, using the experimental dataset, the algorithms were implemented and adopted to minimize the distance between the simulation results and the results in [20].

The hypothesis of this study is based on the results from the six estimations as follows:

$$H_0 : STD_E^2 \geq STD_D^2 \tag{6}$$

$$H_1 : STD_E^2 < STD_D^2 \tag{7}$$

where STD_E^2 is the standard deviation of the optimized result E, STD_D^2 is the standard deviation of the model under study D, and n_E, n_D are the number of variables for the optimized and model result, respectively.

To ensure that the final simulated results were statistically consistent with the experimental results in Table 5, a statistical test, the F-test [51], was applied to the ESe-PSO algorithm results with the model under study and the experimental data. The results, using the method from Hoque et al., 2005, show that all the metabolites achieved an STD close to the mean and 0. Thus, this demonstrates that the results produced by ESe-PSO are consistent with Equation (9). The hypothesis of the result in Table 5 was calculated and confirmed using Equations (8) and (9).

$$F_{test} = \frac{STD_E^2}{STD_D^2} = \frac{0.7381}{2.0941} = 0.3525 \tag{8}$$

$$F_{1-0.05} = \frac{1}{F_{0.05,n_E,n_D}} = \frac{1}{F_{0.05,15,15}} = \frac{1}{2.4034} = 0.4161 \tag{9}$$

The hypothesis in Table 5 aims to minimize the distance of the model under study. Therefore, it was concluded that H_0 is rejected, while H_1 is accepted as a reasonable result. The model simulation, after estimation, is presented in Table 5.

After the kinetic parameters were estimated and the model outputs under study were minimized, an observable increase or decrease in the model pathway output simulation results was noted as compared to the model under investigation, as shown in Figure 4. In the glycolysis pathway, the model response simulation of $GLcex$, FDP, $GAP/DHAP$,

PEP, and PYR decreased, while $G6P$ and $F6P$ increased due to the pts system and a little consumption of $GLcex$. In the pentose phosphate pathway, the model outputs simulation of $6PG$, $Ru5P$, $R5P$, $Xu5P$, and $E4P$ increased, while $S7P$ decreased due to the increase in $G6P$ and the involvement of $F6P$ and $GAP/DHAP$ in the calculation.

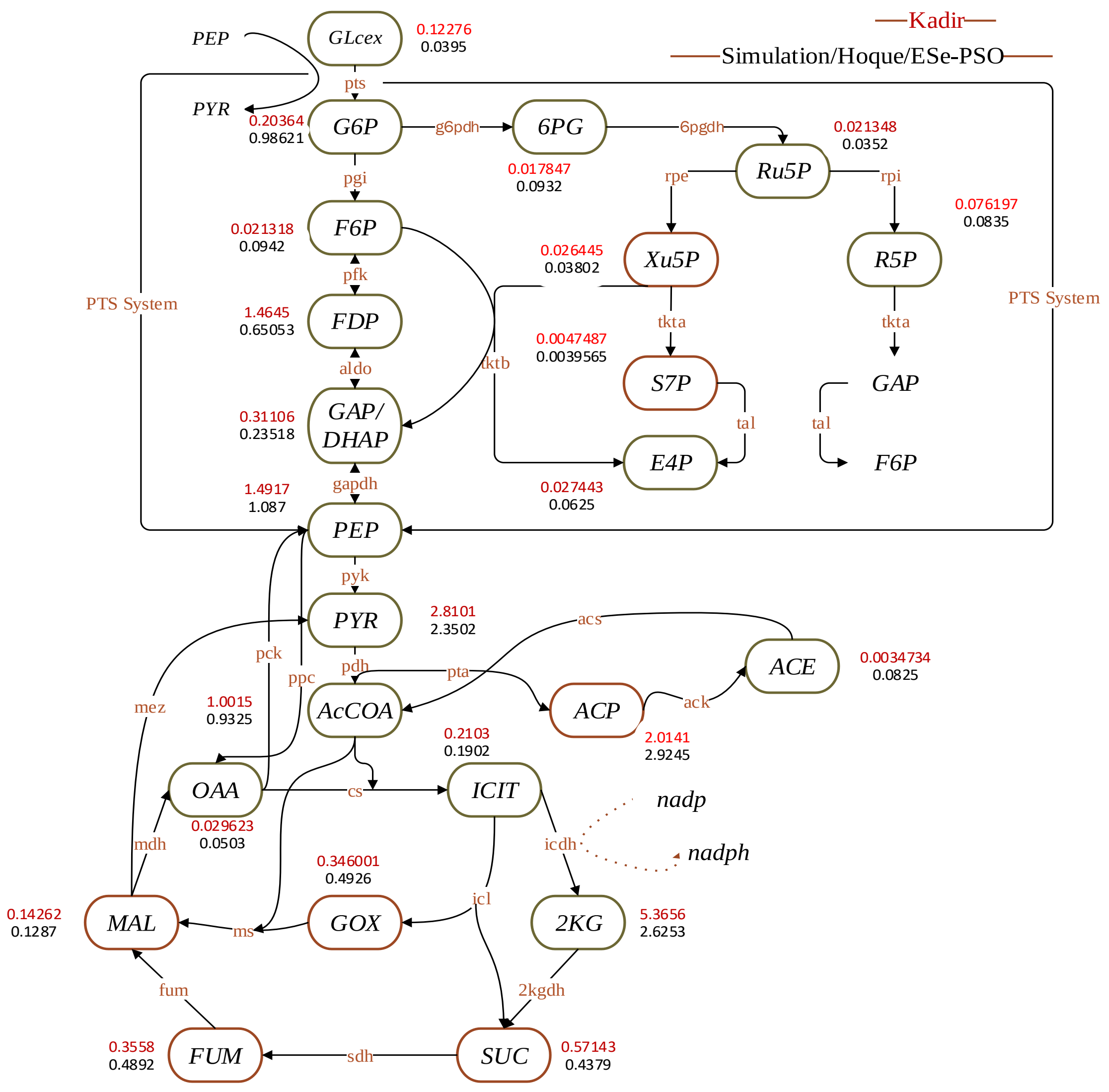

Figure 4. The model simulation using the Ese-PSO algorithm.

In the TCA cycle, the simulated model outputs of OAA, FUM, and GOX increased due to gluconeogenesis/analaprotic pathway involvement and the effect of the mez, pck, and ppc enzymes. This was also due to the increases in PEP and PYR, whereas the metabolites $ICIT$, 2KG, SUC, and MAL decreased. Moreover, acetate formation had a certain impact on the model response, which resulted in increases in ACP and decreases in ACE and $AcCOA$. This was due to $AcCOA's$ involvement in the TCA cycle and the glyoxylate pathways. In the glyoxylate pathway, the metabolites $ICIT$, SUC, and MAL decreased, while GOX increased. This was attributed to the involvement of the TCA cycle, the analaprotic pathway, and $AcCOA$ involvement. In the analaprotic pathway, the metabolites PEP, PYR, and MAL decreased, while the metabolite OAA increased due to the TCA cycle and analaprotic pathway involvement.

On the contrary, the other metabolites moved slightly towards the experimental data with small errors. These changes occurred due to other metabolites' participation, model complexity, glucose depletion, and the lumping together of various metabolites to simplify the model.

4.2. ESe-PSO Algorithm with Different Optimization Problems

The performance of the enhanced segment particle swarm optimization (ESe-PSO) algorithms was compared to that of the original segment particle swarm optimization (Se-PSO) algorithms. The test functions were chosen from six different benchmark functions using the Sphere, Rosenbrock, Rastrigin, Griewank, Shubert, and Booth functions with asymmetric initial range settings (higher and lower boundary values). The experimental results indicated that the ESe-PSO method outperformed the original Se-PSO algorithm in terms of convergence speed under all test conditions. However, the experimental results established ESe-PSO as a potentially useful optimization algorithm in a variety of other fields.

Nonlinear functions are used as a comparison here. The first function is the Sphere function, which is represented by equation $(f(x))$, as follows:

$$f(x) \ = \ \sum_{i\,=\,1}^{n} X^2 \tag{10}$$

where $X = (X_1,\ X_2, \ldots,\ X_n)$ is an n-dimensional real valued vector. The second function is the Rosenbrock function as described by equation $(f_1(x))$:

$$f_1 \ = \ \sum_{i\,=\,1}^{n}(100\left(X_i - X_i^2\right)^2 + (X_i - 1)^2) \tag{11}$$

The third function is the generalized Rastrigrin function as described by equation $(f_2(x))$:

$$f_2 \ = \ 10d + \sum_{i\,=\,1}^{d}\left[x_i^2 - 10\cos(2\pi\ x_i)\right] \tag{12}$$

The fourth function is the generalized Griewank function as described by equation $(f_3(x))$:

$$f_3(x) \ = \ \frac{1}{4000}\sum_{i\,=\,1}^{n} X_{i\,=\,1}^2 - \prod_{i\,=\,1}^{n}\cos\left(\frac{X_i}{\sqrt{i}}\right) + 1 \tag{13}$$

The fifth function is the generalized Shubert function as described by equation $(f_4(x))$:

$$f_4 \ = \ \left(\sum_{i\,=\,1}^{5} i\ \cos((i+1)x_1 + i)\right)\left(\sum_{i\,=\,1}^{5} i\ \cos((i+1)x_2 + i)\right) \tag{14}$$

The sixth function is the generalized Booth function as described by equation $(f_5(x))$:

$$f_5 \ = \ (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2 \tag{15}$$

As shown in Table 8, the maximum number of iterations for each function in both algorithms was set to 50, 100, and 150. The bird number was set to 20, 40, 60, and 80. Each algorithm was evaluated 10 times in order to determine the mean global optimum position.

Table 8. The functions' boundaries.

Functions	**Lower and Upper Values**
f	$[-5, 5]$
f_1	$[-5, 10]$
f_2	$[-5.12, 5.12]$
f_3	$[-600, 600]$
f_4	$[-10, 10]$ for $i \ = \ 1, 2$, $[-5.12, 5.12]$ $i \ = \ 1, 2$
f_5	$[-10, 10]$ for all $i \ = \ 1, 2$

Furthermore, as demonstrated in Tables 9 and 10, the ESe-PSO convergence speed towards the optimal values was faster than that of Se-PSO. It is worth noting, however, that the ESe-PSO method's convergence was swift in all functions but slowed when scanning a huge space for the global optimum location before being chosen by the ESe-PSO algorithm as it approached the optimum.

Table 9. ESe-PSO consumption for Sphere function.

Function	Calls	Total Time	Self-Time
ESe-PSO	1	0.194 s	0.003 s
PSO	3	0.191 s	0.040 s
Sphere	4920	0.161 s	0.151 s

Table 10. Se-PSO consumption for Sphere function.

Function	Calls	Total Time	Self-Time
Se-PSO	1	0.213 s	0.004 s
PSO	3	0.209 s	0.030 s
Sphere	4920	0.179 s	0.179 s

The ESe-PSO took 0.194 s to attain the optimum global position, whereas the Se-PSO took only 0.213 s. The self-time column in Tables 9 and 10 show that ESe-PSO took 0.004 s to determine the global optimum position, while Se-PSO took 0.013 s. Furthermore, as indicated in the calls column (4920 and 650), ESe-PSO searched the vast space nearly twice as fast as Se-PSO.

When compared to Se-PSO, the global optimum position of ESe-PSO in the Sphere function produced a far superior outcome Table 11 in a short period of time.

Table 11. ESe-PSO and Se-PSO global best positions for Sphere function.

Bird Steps	Dimension	Iteration	ESe-PSO	Se-PSO
5	10	15	1.21607×10^{7}	2.12225×10^{6}
15	10	15	3.68313×10^{9}	4.25861×10^{8}
25	10	15	2.83561×10^{10}	2.03589×10^{9}

Furthermore, as shown in Tables 12 and 13, ESe-PSO's convergence speed towards the optimal values was faster than the Se-PSO's. It is worth noting, however, that the convergence of ESe-PSO was swift across the board, but it slowed down when scanning a broad space for the global optimum location before being chosen by the PSO algorithm as it approached the optimum. The ESe-PSO took 0.240 s to obtain the optimum global position, but the Se-PSO only took 0.363 s. The self-time column in Tables 12 and 13 shows that ESe-PSO took 0.0009 s to determine the global optimum position, while Se-PSO took 0.001 s. Furthermore, as indicated in the calls column (4920), ESe-PSO searched the vast space nearly twice as fast as Se-PSO.

Table 12. ESe-PSO consumption for Rastrigin function.

Function	Calls	Total Time	Self-Time
ESe-PSO	1	0.240 s	0.0009 s
PSO	3	0.239 s	0.0291 s
Rastrigin	4920	0.210 s	0.210 s

Table 13. Se-PSO consumption for Rastrigin function.

Function	Calls	Total Time	Self-Time
Se-PSO	1	0.363 s	0.001 s
PSO	3	0.362 s	0.028 s
Rastrigin	4920	0.334 s	0.334 s

In Table 14, the global optimum position of ESe-PSO exhibited a considerably improved outcome in a short time as compared to Se-PSO in the Rastrigin function.

Table 14. ESe-PSO and Se-PSO global best position for Rastrigin function.

Bird Steps	Dimension	Iteration	ESe-PSO	Se-PSO
5	10	15	3.4572×10^{-7}	8.3919×10^{-6}
15	10	15	5.7394×10^{-7}	2.2586×10^{-7}
25	10	15	6.8241×10^{-9}	2.03589×10^{-8}

Furthermore, as shown in Tables 15 and 16, the ESe-PSO's convergence speed towards the optimal values was faster than that of the Se- PSO. It is worth noting, however, that the convergence of ESe-PSO was swift across the board, but it slowed down when scanning a broad space for the global optimum location before being chosen by the PSO algorithm as it approached the optimum. The ESe-PSO took 0.135 s to obtain the optimum global position, but the Se-PSO took only 0.158 s. The self-time column in Tables 15 and 16 shows that ESe-PSO took 0.001 s to determine the global optimum position, while Se-PSO took 0.004 s. Furthermore, as indicated in the calls column (4920), ESe-PSO searched the vast space nearly twice as fast as Se-PSO.

Table 15. ESe-PSO consumption for Rosenbrock function.

Function	Calls	Total Time	Self-Time
ESe-PSO	1	0.135 s	0.001 s
PSO	3	0.134 s	0.021 s
Rosenbrock	4920	0.113 s	0.113 s

Table 16. Se-PSO consumption for Rosenbrock function.

Function	Calls	Total Time	Self-Time
Se-PSO	1	0.158 s	0.004 s
PSO	3	0.154 s	0.026 s
Rosenbrock	4920	0.128 s	0.128 s

In Table 17, the global optimum position of ESe-PSO exhibited a considerably improved outcome in a short time as compared to Se-PSO in the Rosenbrock function.

Table 17. ESe-PSO and Se-PSO global best position for Rosenbrock function.

Bird Steps	Dimension	Iteration	ESe-PSO	Se-PSO
5	10	15	0	0
15	10	15	0	0
25	10	15	0	0

Furthermore, as shown in Tables 18 and 19, the ESe-PSO's convergence speed towards the optimal values was faster than that of the Se-PSO. It is worth noting, however, that the convergence of ESe-PSO was swift across the board, but it slowed down when scanning a broad space for the global optimum location before being chosen by the PSO algorithm as it approached the optimum. The ESe-PSO took 0.363 s to obtain the optimum global position, but the Se-PSO took only 0.048 s. The self-time column in Tables 18 and 19 shows that ESe-PSO took 0.001 s to determine the global optimum position, while Se-PSO took 0.013 s. Furthermore, as indicated in the calls column (4920), ESe-PSO searched the vast space nearly twice as fast as Se-PSO.

Table 18. ESe-PSO consumption for Griewank function.

Function	Calls	Total Time	Self-Time
ESe-PSO	1	0.023 s	0.001 s
PSO	3	0.022 s	0.014 s
Griewank	4920	0.008 s	0.008 s

Table 19. Se-PSO consumption for Griewank function.

Function	Calls	Total Time	Self-Time
Se-PSO	1	0.037 s	0.002 s
PSO	3	0.035 s	0.022 s
Griewank	4920	0.013 s	0.013 s

In Table 20, the global optimum position of ESe-PSO exhibited a considerably improved outcome in a short time as compared to Se-PSO in the Griewank function.

Table 20. ESe-PSO and Se-PSO global best position for Griewank function.

Bird Steps	Dimension	Iteration	ESe-PSO	Se-PSO
5	10	15	9.8935×10^{-8}	8.1222×10^{-7}
15	10	15	5.4572×10^{-12}	8.4574×10^{-10}
25	10	15	6.1742×10^{-15}	3.5258×10^{-13}

Furthermore, as shown in Tables 21 and 22, the ESe-PSO's convergence speed towards the optimal values was faster than that of the Se-PSO. It is worth noting, however, that the convergence of ESe-PSO was swift across the board, but it slowed down when scanning a broad space for the global optimum location before being chosen by the PSO algorithm as it approached the optimum. The ESe-PSO took 0.25 s to obtain the optimum global position, but the Se-PSO took only 0.032 s. The self-time column in Tables 21 and 22 shows that ESe-PSO took 0.002 s to determine the global optimum position, while Se-PSO took 0.003 s. Furthermore, as indicated in the calls column (4920), ESe-PSO searched the vast space nearly twice as fast as Se-PSO.

Table 21. ESe-PSO consumption for Shubert function.

Function	Calls	Total Time	Self-Time
ESe-PSO	1	0.025 s	0.002 s
PSO	3	0.024 s	0.015 s
Shubert	4920	0.006 s	0.006 s

Table 22. Se-PSO consumption for Shubert function.

Function	Calls	Total Time	Self-Time
Se-PSO	1	0.032 s	0.003 s
PSO	3	0.029 s	0.020 s
Shubert	4920	0.009 s	0.009 s

In Table 23, the global optimum position of ESe-PSO exhibited a considerably improved outcome in a short time as compared to Se-PSO in the Shubert function.

Table 23. ESe-PSO and Se-PSO global best position for Shubert function.

Bird Steps	Dimension	Iteration	ESe-PSO	Se-PSO
5	10	15	−186.7278	−186.7234
15	10	15	−186.7300	−186.7245
25	10	15	−186.7309	−186.7289

Furthermore, as shown in Tables 24 and 25, the ESe-PSO's convergence speed towards the optimal values was faster than that of the Se-PSO. It is worth noting, however, that the convergence of ESe-PSO was swift across the board, but it slowed down when scanning a broad space for the global optimum location before being chosen by the PSO algorithm as it approached the optimum. The ESe-PSO took 0.363 s to obtain the optimum global position, but the Se-PSO took only 0.048 s. The self-time column in Tables 24 and 25 shows that ESe-PSO took 0.001 s to determine the global optimum position, while Se-PSO took 0.004 s. Furthermore, as indicated in the calls column (4920), ESe-PSO searched the vast space nearly twice as fast as Se-PSO.

Table 24. ESe-PSO consumption for Booth function.

Function	Calls	Total Time	Self-Time
ESe-PSO	1	0.015 s	0.001 s
PSO	3	0.014 s	0.010 s
Booth	4920	0.004 s	0.004 s

Table 25. Se-PSO consumption for Booth function.

Function	Calls	Total Time	Self-Time
Se-PSO	1	0.027 s	0.004 s
PSO	3	0.023 s	0.014 s
Shubert	4920	0.009 s	0.009 s

In Table 26, the global optimum position of ESe-PSO exhibited a considerably improved outcome in a short time as compared to Se-PSO in the Booth function.

Table 26. ESe-PSO and Se-PSO global best position for Booth function.

Bird Steps	Dimension	Iteration	ESe-PSO	Se-PSO
5	10	15	0.0000	0.0023
15	10	15	0.0000	0.0012
25	10	15	0.0000	0.0001

5. Conclusions

For the purposes of this study, ESe-PSO and a number of other state-of-the-art algorithms were developed and assessed. Particle segmentation was used in order to direct the motion of the particles toward the global optimal location. The inertia weight and dampening procedure of this algorithm were also changed to boost exploration and exploitation in order to discover the global optimum location more quickly. We demonstrate the universal applicability of the adopted technique by successfully applying it to small-scale kinetic parameters. The *E. coli* model's small-scale kinetic parameters were evaluated using the ESe-PSO, Se-PSO, PSO, DE, and GA algorithms. Small-scale models can benefit greatly from the ESe-PSO method because of its high estimate efficiency. The seven kinetic parameters (v_{max}^{pyk}, n_{pk}, $icdh$, k_{icdh}^{f}, $k_{icdhnap}^{d}$, $k_{icdhnadp}^{m}$, and v_{max}^{icl}) were effectively estimated. The F-test, the mean, and the STD proved that the results are moved closely to the real experimental data.

Author Contributions: Conceptualization, M.A.K.A.; methodology, M.A.K.A.; software, M.A.K.A. and A.S.J.; validation, T.A.A.K. and A.S.J.; formal analysis, M.A.K.A.; investigation, H.S.M.A., Y.H.Z.A. and M.S.B.H.; resources, M.Y.; data curation, M.A.K.A.; writing—original draft preparation, M.A.K.A.; writing—review and editing, J.M.Z.; visualization, A.S.J.; supervision, T.A.A.K.; project administration, J.M.Z.; funding acquisition, T.A.A.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors gratefully acknowledge the support from the Institute for Big Data Analytics and Artificial Intelligence (IBDAAI), Universiti Teknologi MARA (UiTM), and the Universiti Malaysia Pahang (UMP).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Mason, J.C.; Covert, M.W. An energetic reformulation of kinetic rate laws enables scalable parameter estimation for biochemical networks. *J. Theor. Biol.* **2019**, *461*, 145–156. [CrossRef] [PubMed]
2. Kunna, M.A.; Kadir, T.A.A.; Remli, M.A.; Ali, N.M.; Moorthy, K.; Muhammad, N. An enhanced segment particle swarm optimization algorithm for kinetic parameters estimation of the main metabolic model of *Escherichia coli*. *Processes* **2020**, *8*, 963. [CrossRef]
3. Villaverde, A.F.; Henriques, D.; Smallbone, K.; Bongard, S.; Schmid, J.; Cicin-Sain, D.; Crombach, A.; Saez-Rodriguez, J.; Mauch, K.; Balsa-Canto, E.; et al. BioPreDyn-bench. BioPreDyn-bench: A suite of benchmark problems for dynamic modelling in systems biology. *BMC Syst. Biol.* **2015**, *9*, 1–5. [CrossRef]
4. Kunna, M.A.; Kadir, T.A.; Jaber, A.S. Sensitivity Analysis in Large-Scale of Metabolic Network of *E. Coli*. In Proceedings of the 2013 International Conference on Advanced Computer Science Applications and Technologies, Kuching, Malaysia, 23–24 December 2013; pp. 346–351.
5. Chassagnole, C.; Noisommit-Rizzi, N.; Schmid, J.W.; Mauch, K.; Reuss, M. Dynamic modeling of the central carbon metabolism of *Escherichia coli*. *Biotechnol. Bioeng.* **2002**, *79*, 53–73. [CrossRef] [PubMed]
6. Usuda, Y.; Nishio, Y.; Iwatani, S.; Van Dien, S.J.; Imaizumi, A.; Shimbo, K.; Kageyama, N.; Iwahata, D.; Miyano, H.; Matsui, K. Dynamic modeling of *Escherichia coli* metabolic and regulatory systems for amino-acid production. *J. Biotechnol.* **2010**, *147*, 17–30. [CrossRef] [PubMed]
7. Matsuoka, Y.; Shimizu, K. Catabolite regulation analysis of *Escherichia coli* for acetate overflow mechanism and co-consumption of multiple sugars based on systems biology approach using computer simulation. *J. Biotechnol.* **2013**, *168*, 155–173. [CrossRef] [PubMed]
8. Oliver, K.; Zaugg, J.B.; Heinemann, M. Bacterial adaptation through distributed sensing of metabolic fluxes. *Mol. Syst. Biol.* **2010**, *6*, 355.
9. Kadir, T.A.; Mannan, A.A.; Kierzek, A.M.; McFadden, J.; Shimizu, K. Modeling and simulation of the main metabolism in *Escherichia coli* and its several single-gene knockout mutants with experimental verification. *Microb. Cell Factories* **2010**, *9*, 1–21. [CrossRef] [PubMed]
10. Azrag, M.A.; Kadir, T.A.; Jaber, A.S. Segment particle swarm optimization adoption for large-scale kinetic parameter identification of *Escherichia coli* metabolic network model. *IEEE Access* **2018**, *6*, 78622–78639. [CrossRef]
11. Kunna, M.A.; Kadir, T.A.; Jaber, A.S.; Odili, J.B. Large-scale kinetic parameter identification of metabolic network model of *E. coli* using PSO. *Adv. Biosci. Biotechnol.* **2015**, *6*, 120. [CrossRef]
12. Ceric, S.; Kurtanjek, Z.; Ceric, S.; Kurtanjek, Z. Model identification, parameter estimation, and dynamic flux analysis of *E. coli* central metabolism. *Chem. Biochem. Eng. Q.* **2006**, *20*, 243–253.
13. Nelder, J.A.; Mead, R. A simplex method for function minimization. *Comput. J.* **1965**, *7*, 308–313. [CrossRef]
14. Won, W.; Park, C.; Lee, S.Y.; Lee, K.S.; Lee, J. Parameter estimation and dynamic control analysis of central carbon metabolism in *Escherichia coli*. *Biotechnol. Bioprocess Eng.* **2011**, *16*, 216–228. [CrossRef]
15. Qin, H.; Ma, X.; Herawan, T.; Zain, J.M. MGR: An information theory based hierarchical divisive clustering algorithm for categorical data. *Knowl.-Based Syst.* **2014**, *67*, 401–411. [CrossRef]
16. Tohsato, Y.; Ikuta, K.; Shionoya, A.; Mazaki, Y.; Ito, M. Parameter optimization and sensitivity analysis for large kinetic models using a real-coded genetic algorithm. *Gene* **2013**, *518*, 84–90. [CrossRef]
17. di Maggio, J.; Paulo, C.; Estrada, V.; Perotti, N.; Ricci, J.C.D.; Diaz, M.S. Parameter estimation in kinetic models for large scale biotechnological systems with advanced mathematical programming techniques. *Biochem. Eng. J.* **2014**, *83*, 104–115. [CrossRef]

18. Villaverde, A.F.; Fröhlich, F.; Weindl, D.; Hasenauer, J.; Banga, J.R. Benchmarking optimization methods for parameter estimation in large kinetic models. *Bioinformatics* **2019**, *35*, 830–838. [CrossRef]
19. Sagar, A.; LeCover, R.; Shoemaker, C.; Varner, J. Dynamic Optimization with Particle Swarms (DOPS): A meta-heuristic for parameter estimation in biochemical models. *BMC Syst. Biol.* **2018**, *12*, 1–5. [CrossRef]
20. Hoque, M.A.; Ushiyama, H.; Tomita, M.; Shimizu, K. Dynamic responses of the intracellular metabolite concentrations of the wild type and pykA mutant *Escherichia coli* against pulse addition of glucose or NH_3 under those limiting continuous cultures. *Biochem. Eng. J.* **2005**, *26*, 38–49. [CrossRef]
21. Azrag, M.A.K.; Kadir, T.A.A.; Ismail, M.A. A Review of Large-Scale Kinetic Parameters in Metabolic Network Model of *Escherichia coli*. *Adv. Sci. Lett.* **2018**, *24*, 7512–7518. [CrossRef]
22. Egea, J.A.; Rodríguez-Fernández, M.; Banga, J.R.; Marti, R. Scatter search for chemical and bio-process optimization. *J. Glob. Optim.* **2007**, *37*, 481–503. [CrossRef]
23. Baker, S.M.; Schallau, K.; Junker, B.H. Comparison of different algorithms for simultaneous estimation of multiple parameters in kinetic metabolic models. *J. Integr. Bioinform.* **2010**, *7*, 254–262. [CrossRef]
24. Chen, Z.; Liu, H.; Tian, Y.; Wang, R.; Xiong, P.; Wu, G. A particle swarm optimization algorithm based on time-space weight for helicopter maritime search and rescue decision-making. *IEEE Access* **2020**, *8*, 81526–81541. [CrossRef]
25. Ghorpade, S.N.; Zennaro, M.; Chaudhari, B.S.; Saeed, R.A.; Alhumyani, H.; Abdel-Khalek, S. Enhanced differential crossover and quantum particle swarm optimization for IoT applications. *IEEE Access* **2021**, *9*, 93831–93846. [CrossRef]
26. Jamian, J.J.; Abdullah, M.N.; Mokhlis, H.; Mustafa, M.W.; Bakar, A.H.A. Global particle swarm optimization for high dimension numerical functions analysis. *J. Appl. Math.* **2014**, *2014*, 329193. [CrossRef]
27. Eberhart, R.; Kennedy, J. A new optimizer using particle swarm theory. In *MHS'95, Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, 4–6 October 1995*; IEEE: Piscataway, NJ, USA, 1995; pp. 39–43.
28. Jaber, A.S.; Ahmad, A.Z.; Abdalla, A.N. A new parameters identification of single area power system based LFC using Segmentation Particle Swarm Optimization (SePSO) algorithm. In Proceedings of the 2013 IEEE PES Asia-Pacific Power and Energy Engineering Conference (APPEEC), Hong Kong, China, 8–11 December 2013; pp. 1–6.
29. Azrag, M.A.; Kadir, T.A. Empirical Study of Segment Particle Swarm Optimization and Particle Swarm Optimization Algorithms. *Int. J. Adv. Comput. Sci. Appl.* **2019**, *10*, 480–485. [CrossRef]
30. Jahan, N.; Maeda, K.; Matsuoka, Y.; Sugimoto, Y.; Kurata, H. Development of an accurate kinetic model for the central carbon metabolism of *Escherichia coli*. *Microb. Cell Factories* **2016**, *15*, 1–19. [CrossRef] [PubMed]
31. Remli, M.A.; Deris, S.; Mohamad, M.S.; Omatu, S.; Corchado, J.M. An enhanced scatter search with combined opposition-based learning for parameter estimation in large-scale kinetic models of biochemical systems. *Eng. Appl. Artif. Intell.* **2017**, *62*, 164–180. [CrossRef]
32. Rodriguez-Fernandez, M.; A Egea, J.; Banga, J.R. Novel metaheuristic for parameter estimation in nonlinear dynamic biological systems. *BMC Bioinform.* **2006**, *7*, 1–8. [CrossRef]
33. Villaverde, A.F.; Egea, J.A.; Banga, J.R. A cooperative strategy for parameter estimation in large scale systems biology models. *BMC Syst. Biol.* **2012**, *6*, 1–7. [CrossRef]
34. Azrag, M.A.; Kadir, T.A.; Kabir, M.N.; Jaber, A.S. Large-Scale Kinetic Parameters Estimation of Metabolic Model of *Escherichia coli*. *Int. J. Mach. Learn. Comput.* **2019**, *9*, 160–167. [CrossRef]
35. Barricelli, N.A. Numerical testing of evolution theories. *Acta Biotheor.* **1962**, *16*, 69–98. [CrossRef]
36. Keshanchi, B.; Souri, A.; Navimipour, N.J. An improved genetic algorithm for task scheduling in the cloud environments using the priority queues: Formal verification, simulation, and statistical testing. *J. Syst. Softw.* **2017**, *124*, 1–21. [CrossRef]
37. Baluja, S.; Caruana, R. Removing the genetics from the standard genetic algorithm. In *Machine Learning Proceedings*; Morgan Kaufmann: Tahoe City, CA, USA, 1995; pp. 38–46.
38. Storn, R. System design by constraint adaptation and differential evolution. *IEEE Trans. Evol. Comput.* **1999**, *3*, 22–34. [CrossRef]
39. Storn, R.; Kenneth, P. Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
40. Kennedy, J.; Russell, E. Particle swarm optimization. In Proceedings of the ICNN'95-International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948.
41. Saini, D.K.; Prasad, R. Order reduction of linear interval systems using genetic algorithm. *Int. J. Eng. Technol.* **2010**, *2*, 316–319.
42. Wang, D.; Tan, D.; Liu, L. Particle swarm optimization algorithm: An overview. *Soft Comput.* **2018**, *22*, 387–408. [CrossRef]
43. Shi, Y.; Eberhart, R.C. Empirical study of particle swarm optimization. In Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), Washington, DC, USA, 6–9 July 1999; Volume 3, pp. 1945–1950.
44. Jiao, B.; Lian, Z.; Gu, X. A dynamic inertia weight particle swarm optimization algorithm. *Chaos Solitons Fractals* **2008**, *37*, 698–705. [CrossRef]
45. Cekus, D.; Skrobek, D. The influence of inertia weight on the Particle Swarm Optimization algorithm. *J. Appl. Math. Comput. Mech.* **2018**, *17*, 5–11. [CrossRef]
46. Mashayekhi, M.; Harati, M.; Estekanchi, H.E. Development of an alternative PSO-based algorithm for simulation of endurance time excitation functions. *Eng. Rep.* **2019**, *1*, e12048. [CrossRef]

47. Bansal, J.C.; Singh, P.K.; Saraswat, M.; Verma, A.; Jadon, S.S.; Abraham, A. Inertia weight strategies in particle swarm optimization. In Proceedings of the 2011 Third World Congress on Nature and Biologically Inspired Computing, Salamanca, Spain, 19–21 October 2011; pp. 633–640.
48. Chatterjee, A.; Siarry, P. Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization. *Comput. Oper. Res.* **2006**, *33*, 859–871. [CrossRef]
49. Xin, J.; Chen, G.; Hai, Y. A particle swarm optimizer with multi-stage linearly-decreasing inertia weight. In Proceedings of the 2009 International Joint Conference on Computational Sciences and Optimization, Sanya, China, 24–26 April 2009; Volume 1, pp. 505–508.
50. Feng, Y.; Teng, G.F.; Wang, A.X.; Yao, Y.M. Chaotic inertia weight in particle swarm optimization. In Proceedings of the Second International Conference on Innovative Computing, Information and Control (ICICIC 2007), Kumamoto, Japan, 5–7 September 2007; p. 475.
51. Yazici, B.; Cavus, M. A comparative study of computation approaches of the generalized F-test. *J. Appl. Stat.* **2021**, *48*, 2906–2919. [CrossRef] [PubMed]