LatinPSO: An Algorithm for Simultaneously Inferring Structure and Parameters of Ordinary Differential Equations Models

Xinliang Tian[a,b], Wei Pang[c], Yizhang Wang[a,b], Kaimin Guo[a,b], You Zhou[a,b,*]

[a] *College of Computer Science and Technology, Jilin University, Changchun, 130012, China*
[b] *Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, China*
[c] *School of Natural and Computing Sciences, University of Aberdeen, UK, AB24 3UE*

Abstract

Simultaneously inferring both the structure and parameters of Ordinary Differential Equations (ODEs) for a complex dynamic system is more practical in many systems identification problems, but it remains challenging due to the complexity of the underlying search space. In this research, we propose a novel algorithm based on Particle Swarm Optimization (PSO) and Latin Hypercube Sampling (LHS) to address the above problem. The proposed algorithm is termed LatinPSO, and it can be effectively used for inferring the structure and parameters of ODE models through time course data. To start with, the real Human Immunodeficiency Virus (HIV) model and several synthetic models are used for evaluating the performance of LatinPSO. Experimental results demonstrated that LatinPSO could find satisfactory candidate ODE models with appropriate structure and parameters.

Keywords: Ordinary Differential Equations, Particle Swarm Optimization, Latin Hypercube Sampling, Structure and parameters optimization

1. Introduction

Ordinary Differential Equations (ODE) are extremely useful in modeling many real-world dynamic systems, including the ecosystems, cell regulation system, and HIV dynamics [1, 2, 3]. For example, a typical ODE model of a biological network takes the form of a system of rate equations, where each equation models the change rate of a single network variable [4]. The (re)construction of biological networks, including metabolic, regulatory gene, and signaling networks, is of fundamental and ultimate importance to the emerging field of computational systems biology [5, 6, 7]. Indeed, a formalism commonly used for capturing and describing the dynamics of biological networks is the form of ODE models. The general form of ODEs is defined in Eq. (1), where X_i is the state variable, n is the number of observable components, and f_i is the relationship among variables [8].

$$\frac{dX_i}{dt} = f_i\left(X_1, X_2, ..., X_n\right)(i = 1, 2, ..., n) \tag{1}$$

In this research, the variables of interest and the interrelations between variables constitute the structure of an ODE model. Furthermore, the constants in an ODE model are called parameters. For instance, an ODE model for the dynamics of Human Immunodeficiency Virus (HIV) in human blood cells [9] is described by Eqs. (2)-(4). In this HIV model, the variables of interest are T, I, and V, where T is the number of uninfected cells, I is the number of infected CD4+T lymphocytes, and V is the number of free viruses. These equations express the rate of changes of the components ($\frac{d.}{dt}$) as the sum of the sources minus the sinks. Uninfected cells are created from sources within the body at rate λ, and they die off at a constant

*Corresponding author
Email address: zyou@jlu.edu.cn (You Zhou)

rate ρ, and become infected. The later process is proportional to the product of the number of uninfected cells and the number of viruses, by a parameter β. δ is the death rate of infected cells, and n is the number of viruses that are released during lysis of one infected cell; c is the rate at which viruses disappear.

$$\frac{dT}{dt} = \lambda - \rho * T - \beta * T * V \tag{2}$$

$$\frac{dI}{dt} = \beta * T * V - \delta * I \tag{3}$$

$$\frac{dV}{dt} = n * \delta * I - c * V - \beta * T * V \tag{4}$$

The structure of an ODE model describes the interrelations between variables. For instance, the structure of the above HIV model can be described as ((-, -, *, T, V), (-, *, T, V), (-, -, *, T, V))(The reason for this description is described in Section 2.1). The parameters of an ODE model describe the influence rate between variables. For instance, the parameters are described as $((\lambda, \rho, \beta), (\beta, \delta), (\eta, \delta, c, \beta))$. The time course data is a matrix containing the time and the variable values. They are the variable values at different time points. For example, the time course data for T is show in Figure 1.

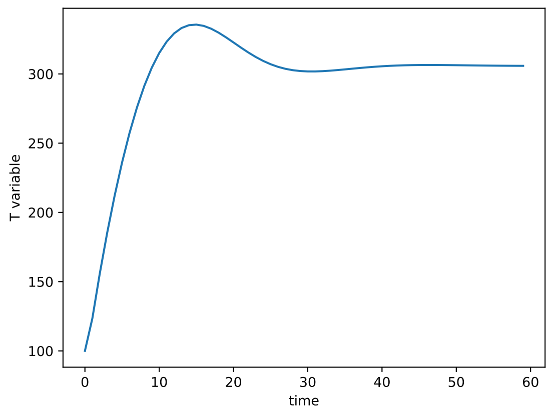

Figure 1: The time course data of variable T in the HIV model

The real HIV ODE model is described in Eqs. $(5 \sim 7)$, where λ is 80, ρ is 0.15, β is 0.00002, δ is 0.55, n is 900, c is 5.5. The initial conditions (variable values at time $t = 0$) is as follows: $T_0 = 100, I_0 = 150$, and $V_0 = 50,000$.

$$\frac{dT}{dt} = 80 - 0.15 * T - 0.00002 * T * V \tag{5}$$

$$\frac{dI}{dt} = 0.00002 * T * V - 0.55 * I \tag{6}$$

$$\frac{dV}{dt} = 900 * 0.55 * I - 0.55 * V - 0.00002 * T * V \tag{7}$$

Our specific task is to identifying the structure and parameters of ODEs from time course data. For this dynamic model of HIV in human blood cells, the known information is the corresponding variables and time course data of these variables. The corresponding variables are T, I, and V. The time course data are observational values of each variable, for instance, those shown in Figure 1. An inference algorithm should find correct structure and parameters simultaneously for the ODE model, which means the inferred solution can closely match the time course data.

Various methods are proposed to infer ODEs during the last few years [4, 7, 10]. Most of these methods can be classified into two groups: the first group is to identify the parameters of the ODEs and the second group is to identify the structure. The former is illustrated by genetic algorithms (GA), and the latter by the genetic programming(GP) approach [13]. In the work presented in [1], M-estimators were proposed for robust estimation of ODE parameters, but the structure of ODE models was given as prior knowledge, thus it becomes a parameter optimization task. The research in [4] focuses on constructing biological networks, but it needs to employ specific domain knowledge to narrow down the search space of candidate model structures. The research in [12] uses a multi-objective Genetic Algorithm (GA) to find robust adaptation gene regulatory networks described as ODEs, and the aim is to find robust adaptation gene regulatory networks, but the learnt ODE model is not closely matching the time course data. The research in [8] evolves differential equation models by genetic programming, but this method had limitations in selecting the basis function and scalability. The research in [13] presents a hybrid evolutionary method for identifying a system of ordinary differential equations (ODEs) to predict the small-time scale traffic measurements data, and it is not to search alternative structures with the same (or similar) system functionalities and behavior.

In this research, we propose a novel swarm-based algorithm to infer the ODE model from biological time course data. More importantly, our algorithm will infer the structure and parameters of the ODE model simultaneously by an improved Particle Swarm Optimization (PSO) algorithm.

Our work on exploring such a kind of hybrid search space containing both discrete space (model structure) and continuous space (model parameters) is motivated by the uncertain nature of biological systems and the incomplete knowledge available: it is often the case that for many biological systems only incomplete or even little knowledge is known, and the underlying interactions among system components may not be fully understood. This necessitates taking into consideration both the structural and parameter uncertainty of the system and broaden the search space when performing system identification [14] so that alternative structures with the same (or similar) system functionalities and behavior may be identified. These alternative structures may be of interest to biologists for further investigation, and new insights into the biological problems may be gained through such interactive processes.

However, because we will explore both the discrete model structural space and the continuous parameter space, the underlying search space for our problem is much more complicated compared with parameter estimation problems. Because of such complexity, PSO is easy to get trapped into local optima. Therefore, we improve the PSO algorithm with the introduction of the Latin Hypercube Sampling (LHS) technique [15, 11], and we name this new algorithm Latin Particle Swarm Optimization (LatinPSO) algorithm.

The rest of this paper is organized as follows. In Section 2, we describe the proposed LatinPSO algorithm. In Section 3, we present the experimental results and perform comparisons. And finally, we conclude the paper and explore future work in Section 4.

2. Algorithm Detail

2.1. Representation of solution space

Each position in the solution space corresponds to one solution of the problem to be solved. We will try to find the best reasonable solution during the iterative search process. The solution space may be high dimensional, and each position is a high-dimensional vector. All the particles will update their position and velocity by global best and personal best. The cost of each particle is calculated by the function introduced in Section 2.3.

An ODE model consists of structure and parameters. To infer the structure and parameters, we first define the structure as the operation relations between variables. In this research, we only deal with the ODE model whose structure contains the $Addition, Subtraction, Multiplication$ and $Division$ operations. We argue that more operators could be included if needed in the future considering problem complexity and available knowledge about the problem. The parameters are the coefficients of the polynomials. Here, we give a more detailed definition of ODEs in Eq. (8) instead of Eq. (1), where $x_1 \sim x_4$ are the state variables, and $k_1 \sim k_4$ are the parameters.

$$\frac{dx_1}{dt} = +k_1 * x_1 + k_2 * x_2 - k_3 * x_3 * x_4 + k_4 \tag{8}$$

Eq. (9) is a solution of inferring ODE model, if all the parameter values and structure are calculated. Now, the problem is converted to how to express the structure and parameters of ODEs by a particle's position in the search space for PSO. The parameters are easy to be represented in PSO, and we use a particle dimension to represent each parameter. For the structure of ODE, we enumerate all structures of this four-variable ODEs model to create a list and use the indices of the list to express all structures. Finally, we add one additional dimension after all the parameters to represent the structure of ODE, as in Eq. (10) and Table 1, and we use S_i to express structure $(+, +, -, +, *, x_3, x_4)$.

$$(k_1, k_2, k_3, k_4, (+, +, -, +, *, x_3, x_4)) \tag{9}$$

$$(k_1, k_2, k_3, k_4, S_i) \tag{10}$$

Table 1: ODE structures list about Eq. (8)

S_i	**ODE structures**
1	$(+, +, +, +, *, x_1, x_2)$
2	$(-, +, +, +, *, x_1, x_2)$
3	$(-, -, +, +, *, x_1, x_2)$
...	......
n	$(-, -, -, -, /, x_4, x_4)$

In our algorithm, one ODE is made up of four items: two single-variable items, like $k_1 * x_1, k_2 * x_2$ in Eq. (8). a two-variable item, like $k_3 * x_3 * x_4$, and a constant item, like k4. we use plus or minus to express the relationship between this four items, and *multiplication* or *division* to express the relationship between two variables in the two-variable items. So, we can use $(+, +, -, +, *, x_3, x_4)$ to express the structure of Eq. (8), and add parameter parts (k_1, k_2, k_3, k_4) to express one ODE like Eq. (8). It is noted that we apply the following assumption on the structure of the ODE model: there is only one item representing variable interaction (e.g., protein binding) and no more than three variables can be bound together (i.e., there is only one two-variable item). This assumption will narrow down the underlying search space. We argue that this assumption is reasonable in biology as more variable interactions means less robust of the system and more expensive to maintain for a biological system of up to 4 variables. We can certainly remove or change this assumption and assume more variable interactions for more complex systems, and this can be easily done by changing the structure encoding strategy. However, this is beyond the scope of this research and will be part of our future work.

2.2. The LatinPSO algorithm

The solution space is huge even if the number of variables of ODEs is small. For a four-variable ODE, when the range of a parameter is [0, 1.0], the number of solutions is more than 1.0E60. When constructing ODE models by PSO, it is easy to get trapped in local optima during the search process. We improve the original PSO by introducing the Latin Hypercube Sampling technique [15, 11], which resulted in the proposed LatinPSO.

2.2.1. Particle Swarm Opitimisation (PSO) algorithm

PSO is a swarm inspired computational technique proposed by Kennedy and Eberhart in 1995 [16, 17, 18, 19, 20], where the swarm consists of a number of particles. Each particle represents a potential solution.

Each particle keeps track of its coordinate in hyperspace, which is associated with the best solution (cost) it has achieved so far. This is called *pbest*. Another best value is called *gbest*, which is the entire swarm's best-known position.

Suppose that the search space is D-dimensional, the $i-th$ particle of the swarm can be represented by a D-dimensional vector $X_i = (X_{i1}, X_{i2}, ..., X_{iD})$. The velocity of the particle can be represented by another D-dimensional vector $V_i = (v_{i1}, v_{i2}, ..., v_{iD})$. The PSO updates its velocity and position according to its previous pbest and the gbest, and the position and velocity update methods are given in Eq. (11) and Eq. (12)., where c_1 and c_2 are positive constants called acceleration, $rand_1$ and $rand_2$ are random numbers uniformly distributed in [0, 1], and w is the inertia weight [21, 22, 23, 24].

$$V_i(t) = w * V_i(t-1) + c_1 * rand_1 * (pbest_i - x_i^{[t]}) + c_2 * rand_2 * (gbest - x_i^{[t]}) \tag{11}$$

$$x_i^{[t]} = x_i^{[t-1]} + V_i^{[t]} \tag{12}$$

2.2.2. Latin Hypercube Sampling (LHS) Technique

LHS [15, 11] is a random-based sampling method, which ensures that the samples are distributed across the entire range even if the number of particles is small. The basic idea of LHS is described as follows. Suppose Z is an n-dimensional hypercube, and the innovative thej^{th} variable z_j, where $j = 1, ..., n$. We need to get H samples (solutions) within this hypercube given as follows: 1) Divide the Z_j range, $[Z_j^L, Z_j^U]$, into H sub-ranges, and every sub-range has the same size; 2) generate a matrix D_{H*n}, and every column of this matrix is the random full permutation of sequence $\{1, 2, ..., H\}$; 3) every row of this D_{H*n} matrix has n elements, and these n elements are used as an index to select one sub-hypercube, and then we get a sample solution randomly within this sub-hypercube. For example, when $H = 10$ and $n = 3$, the corresponding matrix $D10 * 3$ is shown in Table 2.

Table 2: The corresponding matrix D10*3

	Sampling Particle Value		
Sampling Particles Numbers	1	7	2
	2	9	9
	3	2	10
	4	1	7
	5	5	3
	6	6	1
	7	8	2
	8	4	5
	9	10	6
	10	3	8

This configuration of D10*3 indicates that there is a problem in three-dimensional search space, and every dimension is divided into 10 sub-spaces indexed by [1, 2, ...,10] and, therefore, this search space is divided into 1,000 sub-squares. So, the following solutions (1,7,2), (2,9,9), (3,2,10), (4,1,7), (5,5,3), (6,6,1), (7,8,2), (8,4,5), (9,10,6), (10,3,8) are 10 samples produced by the LHS method. Therefore, the samples generated by the LHS algorithm are evenly distributed in the entire hypercube spaces. Therefore it is a space-filling sampling method [25].

2.2.3. The LatinPSO algorithm

We propose the LatinPSO algorithm, which is characterized by updating the position of poorly performed particles and initializing population with LHS algorithm. LatinPSO improves the original PSO algorithm from two aspects: the initialization phase and the iterative process phase. In the initialization phase, we use the LHS technique to assign the initial position to each particle rather than the random initialization method. In the iterative process, we select some particles which have poor cost values and clear the current position value and give them new position value generated by the LHS algorithm. Then we place these new particles into the population to continue the iterative process until the termination conditions are met. The procedure of LatinPSO algorithm is shown in Algorithm 1.

Algorithm 1 LatinPSO algorithm

```
1: 1) Initialize population.
2: 1.1) The population is initialized randomly.
3: 1.2) The population is initialized with LHS.
4: 2) Calculate the cost value of each particle.
5: 3) Population evolution.
6: 3.1) Update the velocity and position of each particle.
7: 3.2) Choose particles with poor cost values and update their positions according to LHS.
8: 4) Evaluate termination condition
9: 4.1) Yes, stop iteration
10: 4.2) No, go to Step (2).
```

There are three versions of LatinPSO which are different in choosing Step (1) and Step (3). By comparing the three versions of the algorithm with the PSO algorithm, we find that all of them have improved the performance of the algorithm. LatinPSO-1: choose Step (1.2) and Step (3.1). LatinPSO-2: choose Step (1.1) and Step (3.2). LatinPSO-3: choose Step (1.2) and Step (3.2). In Step (3.2), we choose 10% of all particles with the poorer cost values in every ten iterations, and change the positions of these chosen particles with LHS.

2.3. Genetic Algorithm

In Section 3, we will compare the performance of the proposed LatinPSO with Genetic Algorithm(GA) because GA is also generally used for optimization problems. Here is a brief description of the GA for inferring ODE. GA uses the genetic evolution process and does not wrok in the same way as PSO, but the comparison can be presented. For instance, in Equation.13, we assumed that an individual parameter a_1x represents a gene, the genes are joined to form on a chromosome and all chromosomes are joined together to form the whole genome. Algorithm 2 summarizes the procedure of GA.

$$Genome = \begin{cases} \dfrac{dx}{dt} = a_1x + b_1y + c_1z \\ \dfrac{dy}{dt} = a_2x + b_2y + c_2z \\ \dfrac{dz}{dt} = a_3x + b_3y + c_3z \\ Chromosome = \begin{cases} a_1x + b_1y + c_1z \\ Gene = \begin{cases} a_1x \end{cases} \end{cases} \end{cases} \tag{13}$$

Algorithm 2 Genetic Algorithm

```
1: 1) Assume a problem domain as a chromosome.
2: 1.1) Set the size of population N.
3: 1.2) Set the crossover probability Pc
4: 1.3) Set the mutation probability Pm
5: 2) Define fitness function.
6: 3) Generate the Initial population size N.
7: 4) Calculate the fitness of individual chromosome.
8: 5) Select high the fit chromosomes from current population for mating.
9: 6) Create offspring chromosomes by genetic operators.
10: 7) Set the new offspring in the new population.
11: 7.1) Repeat step 5 until the new population size reaches to initial population
12: 8) Evaluate termination condition
13: 8.1) Yes, stop iteration
14: 8.2) No, go to Step 4
```

2.4. The Definition of Cost Function

The cost value of each individual is defined as the sum of the squared error and shown in Eq. (14).

$$cost = \sum_{i=1}^{n} \sum_{k=0}^{T-1} \left(x_i^{'}(t_0 + k\Delta t) - x_i(t_0 + k\Delta t) \right)^2 \tag{14}$$

In Eq. (14), t_0 is the starting time, Δt is the step size, n is the number of state variables, and T is the number of data points. $x_i(t_0 + k\Delta t)$ is the given target time course data ($k = 0, 1, ..., T-1$). $x_i^{'}(t_0 + k\Delta t)$ is the time course data acquired by calculating the system of ODE represented by a particle from LatinPSO.

In this research, we obtain the time course data for each ODE through an ODE simulator (i.e. the odeint function in Python scipy package (https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.odeint.html), and it integrates a system of ODE and solves the initial value problem for both stiff or non-stiff systems of first order ODEs. In the future we will use real-word data to verify our algorithm if more data are available.

3. Experimental results

We prepared three different tasks to test the effectiveness of the proposed LatinPSO algorithm. These experiments are the HIV model, the 3-variable model, and the 4-variable model. In each experiment, we compare the experimental results of the three versions LatinPSO and the original PSO. The experimental parameters for these algorithms are the same in three experiments: the population size is 1,000, the iteration time is 1,000, w is 0.7, $c1$ is 1.5, and $c2$ is 1.5. In the LHS algorithm, the parameters are set as described in the previous LHS algorithm, and we choose 10% of all particles on the basis of cost value in every ten iterations, and we change the positions of those chosen particles with LHS. The parameters of time course data are set as follows: in the HIV model and 3-variable model experiments, $t0$ is 0, T is 60, Δt is 1.0. In the 4-variable model experiment, $t0$ is 0, T is 200, Δt is 0.1.

3.1. The HIV Model

This example models the dynamics of the HIV virus in human blood cells. The model describes three components, comprising the number of uninfected (T) and infected (I) $CD4 + T$ lymphocytes, and the number of free virions (V). It consists of three differential equations, as shown in Eq. (5) to Eq. (7). By applying LatinPSO, we acquired the best cost value of 988.1 and the corresponding ODE model is described as Eqs. (15)∼(17).

$$\frac{dT}{dt} = -0.28 * T - 0.43 * I + 113 \tag{15}$$

$$\frac{dI}{dt} = 0.13 * T - 1.63 * 10e - 8 * V * V + 38.7 \tag{16}$$

$$\frac{dV}{dt} = -0.48 * V + 417.8 * I + 199.6 \tag{17}$$

The time course data generated by this system are shown in Figure 2, Figure 3 and Figure 4 along with those of the target model. We can see that the time course data created by ODE model that we constructed closely match the target time course data.

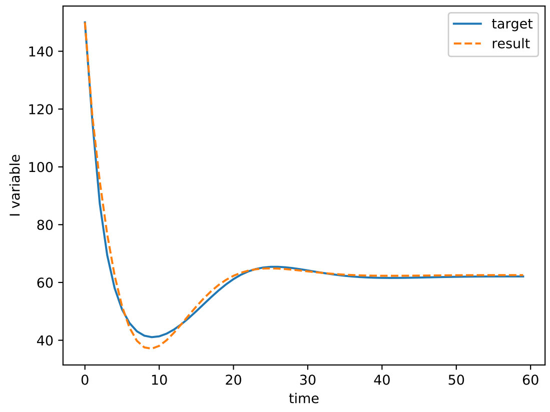

Figure 2: A comparison of time course data between the target and learnt models on variable I

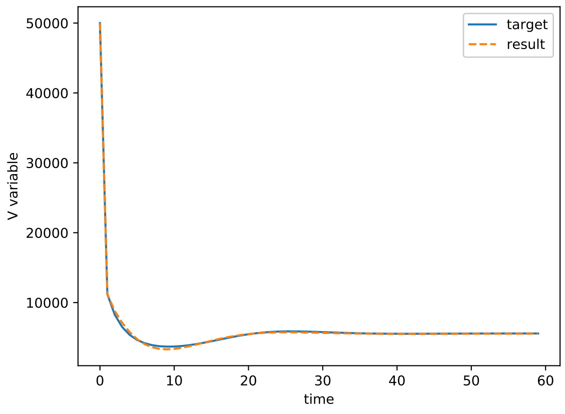

Figure 3: A comparison of time course data between the target and learnt models on variable V.

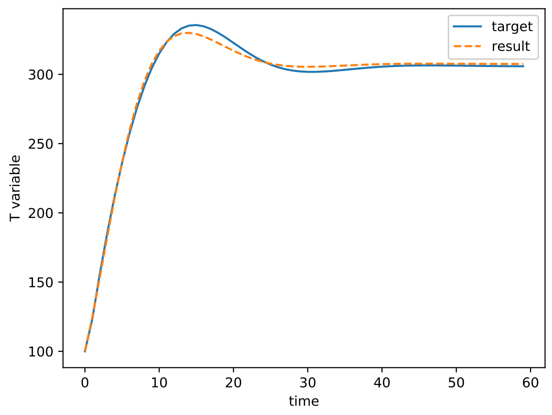

Figure 4: A comparison of time course data between the target and learnt model on variable T.

We performed 20 trial experiments to compare the performance of PSO and LatinPSO. The results are shown in Table 3 and Figure 5.(excessive outliers in the PSO data cause the y-axis to be too long, so these ouliters are ignored in Figure 5) The results show that the three modified algorithms have better ability to search for the system of ODEs given the known time course data, and LatinPSO-3 outperforms LatinPSO-1 and LatinPSO-2 in this problem. Table 4 shows that the t-test of 20 experimental results of LatinPSO-3 and PSO, $P(T < t) = 0.17 > 0.05$, so the difference between the two is not statistically significant.

A close investigation of both the learnt model shown in Eqs. (15)∼(17) and the real model shown in Eqs. (5)∼(7) reveals that our algorithm identified an alternative structure for the HIV model: the identified model suggests that there is interaction between variables T and V (T*V). This is because we relaxed the search constraints by allowing interactions between variables, but in the real HIV model there is no such interactions. Such a model may be of interest to domain experts as they can either reject such interactions or further investigate it. From the complex system point of view, the identified model has very similar behavior to the real one, and this may also reveal more insights if we are further to examine the system dynamics and how to achieve them.

Table 3: The cost values of Different algorithms in the HIV model

	Average Cost	**Min Cost**	**Max Cost**	**Standard Deviation**
PSO	45181.37	12927.7	250209.98	71040.54
LatinPSO-1	25583.19	8945.10	75408.13	16149.64
LatinPSO-2	20807.84	1308.76	52225.34	12975.22
LatinPSO-3	19913.61	988.10	58903.02	13239.53

Table 4: T-test results of two algorithms

	df	**t Stat**	**P(T <= t) two-tailed**	**two-tailed critical value**
PSO and LatinPSO-3	19	1.43	0.17	2.09

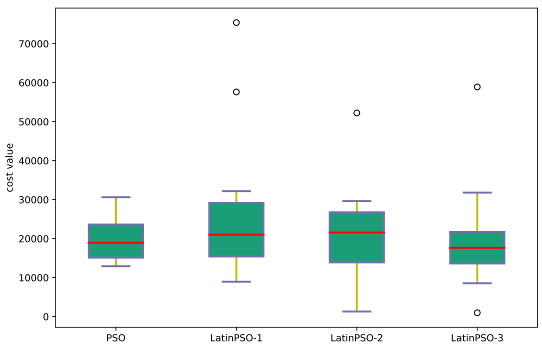

Figure 5: The box plot of cost values in HIV model.

175 ## *3.2. The Artificial Three-Variable ODE Model*

We artificially synthesize an ODE model containing three variables, as shown in Eqs. (18)∼(20). The initial value of $(X1, X2, X3)$ is $(0.1, 0.5, 1.0)$.

$$\frac{dX_1}{dt} = 0.1 * X_1 - 1.0 * X_1 * X_2 + 0.3 \tag{18}$$

$$\frac{dX_2}{dt} = 0.3 * X_3 - 0.1 * X_2 * X_3 + 1.0 \tag{19}$$

$$\frac{dX_3}{dt} = 0.2 * X_1 + 0.2 * X_1 * X_3 \tag{20}$$

By applying LatinPSO, we acquired the best cost value 0.152, and the corresponding ODE model is given in Eqs. (21)∼(23). The time course data generated by this system is shown in Figure 6. Along with that of targets, we can see that they are also closely matched.

$$\frac{dX_1}{dt} = -0.478 * X_1 + 0.188 * X_3 - 0.3 * X_2 * X_2 + 0.222 \tag{21}$$

$$\frac{dX_2}{dt} = -0.139 * X_2 + 0.229 * X_1 + 1.0 * X_1 * X_2 + 1.0 \tag{22}$$

$$\frac{dX_3}{dt} = -0.001 * X_2 + 0.114 * X_1 * X_3 + 0.026 \tag{23}$$

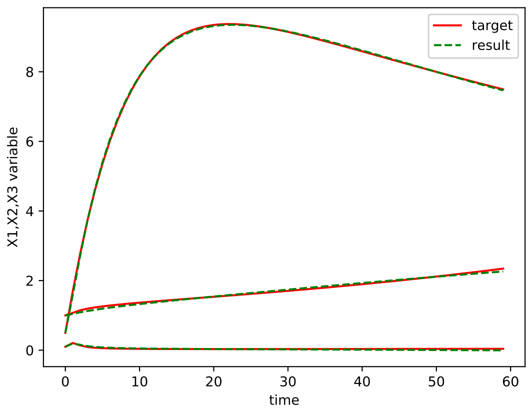

Figure 6: A comparison of time course data between initial and result about $x1, x2, x3$.

Similarly, for each experiment we perform 20 trials to compare the performance of PSO and LatinPSO. The results are shown in Table 5 and Figure 7. The results show that the three modified algorithms outperform the original PSO when searching the system of ODEs given the time course data, and LatinPSO-3 is still better than LatinPSO-1 and LatinPSO-2. Table 6 shows that the t-test of 20 experimental results of LatinPSO-3 and PSO, $P(T < t) = 0.01 < 0.05$, so the difference between the two is statistically significant.

Table 5: The cost value of the three-variable model

	Average Cost	**Min Cost**	**Max Cost**	**Standard Deviation**
PSO	17.59	0.58	26.00	7.50
LatinPSO-1	14.60	0.21	23.53	7.53
LatinPSO-2	14.07	0.33	24.33	8.78
LatinPSO-3	8.78	0.15	22.53	8.45

Table 6: T-test results of two algorithms

	df	**t Stat**	**P(T <= t) two-tailed**	**two-tailed critical value**
PSO and LatinPSO-3	26	2.75	0.01	2.06

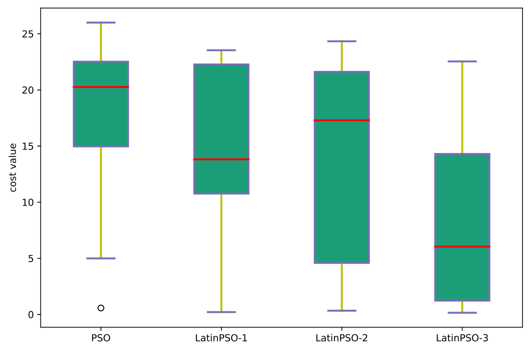

Figure 7: The box plot of cost values in the three-variables model

3.3. *The Artificial Four-Variable ODE Model*

We prepared an ordinary differential equation model which contains four variables as shown in Eqs. (24)∼(27). The initial value of $(X1, X2, X3, X4)$is (0.01, 0.01, 0.01, 0.01).

$$\frac{dX_1}{dt} = -0.1 * X_1 + 0.2 * X_3 - 0.1 * X_3 * X_4 + 0.2 \tag{24}$$

$$\frac{dX_2}{dt} = -0.2 * X_2 + 0.1 * X_1 \tag{25}$$

$$\frac{dX_3}{dt} = -0.2 * X_3 + 0.2 * X_1 * X_4 \tag{26}$$

$$\frac{dX_4}{dt} = -0.1 * X_2 * X_4 - 0.1 \tag{27}$$

By applying our algorithm, we have acquired the best cost value 0.09. And the corresponding ODE model is described as Eqs. (28)∼(31). The time course data generated by this inferred model are shown in Figure 8 along with those of the target model, and we can see that they are also closely matched.

$$\frac{dX_1}{dt} = -0.478 * X_1 + 0.188 * X_4 - 0.3 * X_2 * X_2 + 0.222 \tag{28}$$

$$\frac{dX_2}{dt} = -0.20 * X_2 + 0.113 * X_1 + 3.12le - 7 * X_3 * X_3 - 0.009 \tag{29}$$

$$\frac{dX_3}{dt} = -0.096 * X_3 - 0.081 * X_1 - 0.188 * X_1 * X_2 + 0.019 \tag{30}$$

$$\frac{dX_4}{dt} = 0.043 * X_4 - 0.109 \tag{31}$$

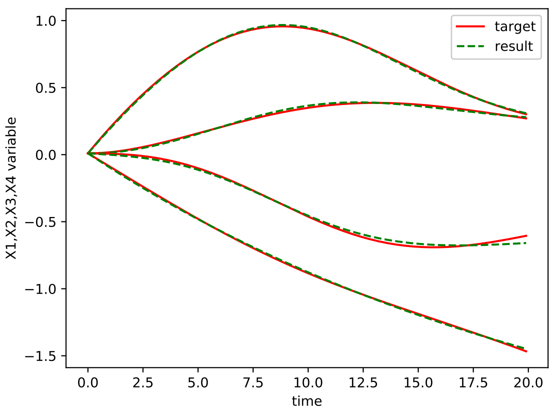

Figure 8: Time course data compare between initial and result about $x1, x2, x3, x4$.

We performed 20 trials for each experiment to compare the performance of PSO and LatinPSO. The results are shown in Table 7 and Figure 9. The results show that the three modified algorithms have better ability to search the system of ODEs given the time course data, and LatinPSO-3 outperforms LatinPSO-1 and LatinPSO-2. Table 8 shows that the t-test of 20 experimental results of LHSPSO-3 and PSO, $P(T < t) = 0.21 > 0.05$, so the difference between the two is not statistically significant.

Table 7: The cost value of the four-variable model

	Average Cost	**Min Cost**	**Max Cost**	**Standard Deviation**
PSO	5.22	0.35	19.58	5.86
LatinPSO-1	4.66	0.11	18.35	5.73
LatinPSO-2	4.02	0.09	17.95	5.25
LatinPSO-3	3.07	0.17	12.95	3.45

Table 8: T-test results of two algorithms

	df	**t Stat**	**P(T <= t) two-tailed**	**two-tailed critical value**
PSO and LatinPSO-3	26	1.27	0.21	2.05

It can be seen from both Sections 3.2 and 3.3 that the identified models do not have the same structure as the original model. This is similar to the HIV model identification in Section 3.1, and again we point out that due to the complexity of the problem, we found alternative model structures. We also point out that given more domain knowledge about the system, it will be more likely for us to identify the correct model. Furthermore, from the system design and synthetic biology point of view, alternative model structures will be of interest as they offer alternative design choices for achieving the same biological functionalities.

We also discover that the models we found for the 3-variable and 4-variable systems are more complicated than the actual models. This means we can further use the model parsimony principle to guide the search, that is, we assign higher cost values to simpler models and penalize complicated ones. This will become part of our future work.

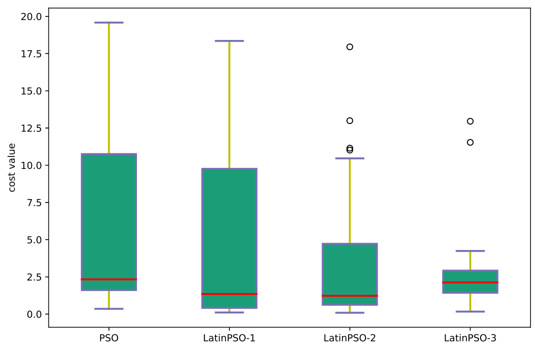

Figure 9: The box plot of cost values in the four-variables model

3.4. A Comparison of Genetic Algorithm and LatinPSO

Genetic Algorithm is shown in Section 2.3. In this section, we tried to analyse the matching degree between original data and the ODE model that the two algorithm find in the same time. The experimental target model is the artificial three-variable ODE model, as shown in Section 3.2. These simulations were conducted on intel Core i5 2.5GHz, 8GB memory in Python IDLE environment. Both algorithms were started at the same time, given a time frame of 600 seconds. Ten trials were performed for each experiment. In GA, the size of population is 1000, the size of crossover probability Pc is 0.6, the size of mutation probability Pm is 0.1, the experimental parameters of the LatinPSO algorithm are the same as the previous experiments.

By applying LatinPSO, we acquired the best cost value 9.84, and the corresponding ODE model is given in Eqs. (32)∼(34). And by applying GA, we acquired the best cost value 18.34, and the corresondding ODE model is given in Eqs. (35)∼(37). The comparison between the data generated by the best models obtained by the two algorithms and the target data is shown in Figure 10. As one can be seen from Figure 10, in terms of the degree of data matching, the model that LatinPSO algorithm finds is superior to the model that the GA finds.

(1).The Resultant ODE Model of LatinPSO

$$\frac{dX_1}{dt} = -0.506 * X_1 + 0.341 * X_3 - 0.874 * X_1 * X_2 \tag{32}$$

$$\frac{dX_2}{dt} = -0.241 * X_2 + 0.442 * X_3 * X_3 + 0.701 \tag{33}$$

$$\frac{dX_3}{dt} = -0.086 * X_3 + 0.254 * X_1 + X_1 * X_3 \tag{34}$$

(2).The Resultant ODE Model of GA

$$\frac{dX_1}{dt} = -0.985 * X_1 - 0.012 \tag{35}$$

$$\frac{dX_2}{dt} = -0.251 * X_2 + 0.806 * X_3 * X_3 - 0.002 \tag{36}$$

$$\frac{dX_3}{dt} = -0.999 * X_1 + 0.950 * X_1 * X_2 + 0.113 \tag{37}$$

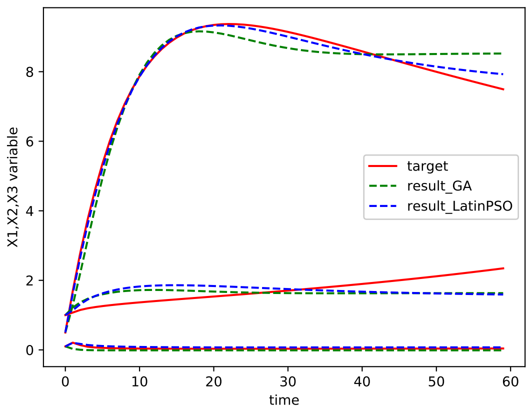

Figure 10: Comparison of result data generated by GA algorithm and LatinPSO algorithm with target data

215 For each algorithm we performed 10 trials to compare GA and LatinPSO algorithm. Under the premise of giving the same time, the cost values obtained by the two algorithms are shwon in Table 9 ,Table 10 and Figure 11. The results show that the LatinPSO algorithm outperform the GA for a given runtime. Table 10 shows that the t-test of 10 experimental results of LatinPSO and PSO, $P(T < t) = 0.01 < 0.05$, so the difference between the two is statistically significant.

Table 9: Cost value of two algorithms

	Average Cost	**Min Cost**	Max Cost	**Standard Deviation**
GA	28.15	18.34	51.22	9.34
LatinPSO	17.72	9.84	34.17	7.63

Table 10: T-test results of two algorithms

	df	**t Stat**	**P(T <= t)** two-tailed	**two-tailed critical value**
GA and LatinPSO	18	2.74	0.01	2.10

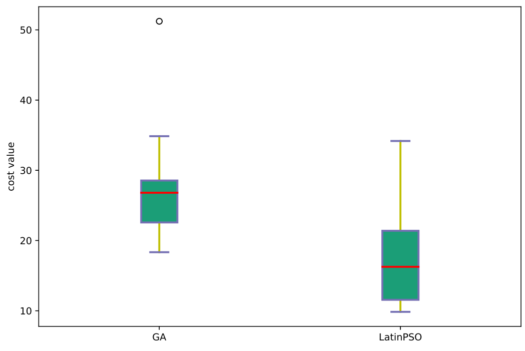

Figure 11: The box plot of cost value of two algorithm

3.5. The Influence of Parameter Values on Experimental Results

In order to select a more appropriate number of iterations, we performed four experiments using HIV model with different numbers of iterations to investigate the effect of the number of iterations on the final results. Other parameter settings remain the same and the number of iterations is set to four values: 250, 500, 750, and 1,000. The results are shown in Table 11. In the same way as above, we also performed four sets of experiments with different numbers of particles, and the number of particles in each group was 250, 500, 750, and 1,000, respectively. The results are presented in Table 12. The results show that as the number of iterations and population size increase, the cost value decreases gradually. When the number of iteration is 1000 and the population size is 1000, the quality of the cost value between the result data and the known data is better, so the value of population size and iteration times is set to 1000 in the experiments.

Table 11: Results with different numbers of iterations

Population Size	**Iteration Times**	**w**	**C1**	**C2**	**Average**
1000	250	0.7	1.5	1.5	55452.3
1000	500	0.7	1.5	1.5	30134.5
1000	750	0.7	1.5	1.5	20091.1
1000	1000	0.7	1.5	1.5	19913.6

Table 12: Results with different swarm size

Population Size	**Iteration Times**	**w**	**C1**	**C2**	**Average**
250	1000	0.7	1.5	1.5	67793.0
500	1000	0.7	1.5	1.5	40134.2
750	1000	0.7	1.5	1.5	34199.5
1000	1000	0.7	1.5	1.5	19913.6

In order to investigate the effect of two parameters (c1 and c2) on the experimental results, we performed a set of experiments with the artificial three-variable ODE model when two parameters took different values. The results are shown in Table 13. From the results, we can see that when both parameters (c1 and c2) are set to 1.5, better results can be obtained.

Table 13: Results with different c1 and c2

Population Size	**Iteration Times**	**w**	**C1**	**C2**	**Average Cost**
1000	1000	0.7	1.5	1.5	8.07
1000	1000	0.7	1.5	2.0	16.88
1000	1000	0.7	2.0	2.0	21.08

In addition, when the number of particles is set to be small, better experimental results can be obtained by increasing the number of iterations. We designed a set of experiments: the number of particles is fixed at 100, and the number of iterations increases from 1,000 to 10,000. The results show that the cost value gradually improves with the number of iterations. The optimal solution is obtained when the number of iterations is 100000, but there is still a gap compared with the results when the number of particles and the number of iterations is 1,000.

Table 14: Results with few population and larger iterations

Population Size	Iteration Times	w	C1	C2	Average Cost
1000	1000	0.7	1.5	1.5	8.07
100	1000	0.7	1.5	1.5	29.809
100	2000	0.7	1.5	1.5	22.157
100	5000	0.7	1.5	1.5	17.413
100	10000	0.7	1.5	1.5	14.868
200	1000	0.7	1.5	1.5	30.743
200	2000	0.7	1.5	1.5	18.949
200	5000	0.7	1.5	1.5	14.282
200	10000	0.7	1.5	1.5	12.389

4. Conclusions and Future work

The task of system identification in terms of inferring the structure and parameters of ODEs plays an important role in many scientific fields [26, 27, 10]. In this research, we proposed a system identification approach for inferring the system of ODEs from given time course data by using LatinPSO algorithm. The experimental results demonstrate that we can succeed in acquiring the system of ODEs which is close to the observed time course data, and the accuracy is better than the original PSO algorithm. More intriguingly, our system identification approach can explore alternative structures of a system with similar behaviour, and this can provide valuable information for not only further investigation into the system but also for system design and implementation.

As one direction of future research, we will apply more constraints and heuristics to further narrow down the search space, for instance, the use of model parsimony principle and more specific constraints on the number of interacting variables. We will also further investigate the fundamental reasons why and how LHS contributes to the overall search process of PSO when performing ODE model inference.

From the practical perspective, we aim to build a platform that offers services to some real-world problems. For example, to help biologists identify possible structure and the associated parameter values of dynamics networks from time course data. And we are aware of the fact that the more the variables of a ODE has, the more complexity of the corresponding solution space. Therefore, another direction of our future work will be extending and improving our algorithms for inferring more complex large-scale ODE models of biological systems.

5. Acknowledgment

This research is supported by the National Natural Science Foundation of China (Grants Nos.61772227, 61572227), the Science & Technology Development Foundation of Jilin Province (GrantsNo. 20180201045GX), the Science Foundation of Education Department of Guangdong Province(Grants Nos.2017KQNCX251, 2018XJCQSQ026) and the Social Science Foundation of Education Department of Jilin Province (Grants No.JJKH20181315SK). WP was supported by the 2015 Scottish Crucible award funded by Royal Society of Edinburgh.

References

[1] Yanping, LIANG, Baosheng, Hengjian, Robust estimation of parameters in nonlinear ordinary differential equation models, Journal of Systems Science and Complexity 29 (1) (2016) 41–60.

[2] R. Mattheij, J. Molenaar, Ordinary differential equations in theory and practice, International Journal of Rock Mechanics and Mining Sciences and Geomechanics Abstracts 27 (4) (1996) 343.

[3] H. Xue, H. Miao, H. Wu, Sieve estimation of constant and time-varying coefficients in nonlinear ordinary differential equation models by considering both numerical error and measurement error., Annals of Statistics 38 (4) (2010) 2351–2387.

[4] S. Dzeroski, L. Todorovski, Modeling the Dynamics of Biological Networks from Time Course Data, Springer New York, 2010.

[5] H. Cao, L. Kang, Y. Chen, J. Yu, Evolutionary modeling of systems of ordinary differential equations with genetic programming, Genetic Programming and Evolvable Machines 1 (4) (2000) 309–337.

[6] N. D. Price, I. Shmulevich, Biochemical and statistical network models for systems biology., Current Opinion in Biotechnology 18 (4) (2007) 365–370.

[7] S. E. F. Spencer, S. M. Hill, S. Mukherjee, Inferring network structure from interventional time-course experiments, Annals of Applied Statistics 9 (1).

[8] H. Iba, Inference of differential equation models by genetic programming, Information Sciences 178 (23) (2008) 4453–4468.

[9] K. Soetaert, T. Petzoldt, J. D. Leeuw, A. Zeileis, Inverse modelling, sensitivity and monte carlo analysis in r using package fme, Journal of Statistical Software 33 (03) (2010) 1–28.

[10] S. Russell, P. Norvig, Artificial intelligence. a modern approach 2 (1995) 27–34.

[11] M.D.Mckay, R.J.Beckman, W.J.Conover, A comparison of three methods for selecting values of input variables in the analysis of output from a computer code, Technometrics 42 (1) (2000) 55–61.

[12] H. P. Ren, X. Huang, J. Hao, Finding robust adaptation gene regulatory networks using multi-objective genetic algorithm, IEEE/ACM Transactions on Computational Biology and Bioinformatics 13 (3) (2016) 571–577.

[13] Y. Chen, B. Yang, Q. Meng, Y. Zhao, A. Abraham, Time-series forecasting using a system of ordinary differential equations, Information Sciences 181 (1) (2011) 106–114.

[14] R. Kashyap, System identification, Academic Press,, 1976.

[15] Z. Shu, P. Jirutitijaroen, Latin hypercube sampling techniques for power systems reliability analysis with renewable energy sources, IEEE Transactions on Power Systems 26 (4) (2011) 2066–2073.

[16] R. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: International Symposium on MICRO Machine and Human Science, 2002, pp. 39–43.

[17] Y. Shi, R. C. Eberhart, Parameter selection in particle swarm optimization, Springer Berlin Heidelberg, 1998.

[18] J. Kennedy, R. Eberhart, Particle swarm optimization, in: IEEE International Conference on Neural Networks, 1995. Proceedings, 1995, pp. 1942–1948 vol.4.

[19] R. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: International Symposium on MICRO Machine and Human Science, 2002, pp. 39–43.

[20] L. Zhang, H. Yu, S. Hu, A new approach to improve particle swarm optimization, Lecture Notes in Computer Science 2723 (2003) 134–139.

[21] X. Yang, J. Yuan, J. Yuan, H. Mao, A modified particle swarm optimizer with dynamic adaptation, Applied Mathematics and Computation 189 (2) (2007) 1205–1213.

[22] R. Poli, J. Kennedy, T. Blackwell, Particle swarm optimization, Swarm Intelligence 1 (1) (2007) 33–57.

[23] K. L. Du, M. N. S. Swamy, Search and Optimization by Metaheuristics, Springer International Publishing, 2016.

[24] I. C. Trelea, The particle swarm optimization algorithm: convergence analysis and parameter selection, Information Processing Letters 85 (6) (2003) 317–325.

[25] X. N. Huang, H. P. Ren, Identification of robust adaptation gene regulatory network parameters using an improved particle swarm optimization algorithm, Genetics and Molecular Research 15 (2).

[26] S. eroski, L. Todorovski, Discovering dynamics: from inductive logic programming to machine discovery, Journal of Intelligent Information Systems 4 (1) (1995) 89–108.

[27] P. Langley, Elements of machine learning, Morgan Kaufmann Publishers Inc., 1995.