Journal of Computational Biology and Bioinformatics Research Vol. 3(5), pp. 53-62, 2 September, 2011
Available online http://www.academicjournals.org/jcbbr
ISSN-2141-2227 ©2011 Academic Journals

Full Length Research Article

Using the ant colony optimization algorithm in the network inference and parameter estimation of biochemical systems

Philip Christian C. Zuniga

Department of Computer Science, University of the Philippines, Diliman, Quezon City, Philippines.
E-mail: philip_zuniga@yahoo.com.

Accepted 3 June, 2011

Developing models that can represent biochemical systems is one of the hallmarks of systems biology. Scientists have been gathering data from actual experiments, but there is a lack in computer models that can be used by scientists in analysing the various biochemical systems more effectively. In this research, we propose to use an ant colony optimization (ACO) algorithm for the network inference and parameter estimation of biochemical systems, particularly S-systems. The ACO has been used for various problems, and with several improvements, it can also be used to solve the problems that we are considering. Since the ACO has discrete and continuous forms, we plan to use each form for the network inference and parameter estimation problems respectively. The results of our work show that the ACO can be effectively used in the formation of model for biochemical systems.

Key words: Biochemical systems, S-systems, ant colony optimization.

INTRODUCTION

The use of biochemical systems theory (BST) has found many applications in metabolic engineering, drug development, etc. Biochemical systems theory is a framework to model and analyze biochemical systems. Since biochemical systems are highly complex and are highly correlated with each other, models used to represent these kinds of systems are usually non-linear. A biochemical system is a system of metabolites/chemical compounds that interact with one another. These interactions can lead to a change in the concentration of each metabolite/compound or even to the formation of new components. One of the main challenges in systems biology is the development of mathematical or computational models that can simulate the effects of the biochemical system on the various components in it.

The GMA and S-systems formulations of BST are non linear differential equations which involves parameters whose values have to be estimated. Unlike linear systems where effective parameter estimation can be used, there is currently no available algorithm which can efficiently handle S-system or GMA-systems. For this work, we concentrate mainly on the use of S-system [del Rosario et al., 2008]. In an S- system formulation, we let the concentration of metabolites/chemical component at time be represented by: $X_1(t), X_2(t) \dots X_n(t)$ Relationships of the metabolites can be represented as:

$$\dot{X}_i = \alpha_i \prod_{j=1}^{n+m} X_j^{g_{ij}} - \beta_i \prod_{j=1}^{n+m} X_j^{h_{ij}} \quad (1)$$

We replace $X_j(t)$ with X_j with the understanding that X_j is actually a function of time. The parameters α_i and β_i are called rate constants. These parameters represent the rate at which the concentrations of the metabolites/chemical components of the system increases or decreases with respect to time. The parameters g_{ij} and h_{ij} are the kinetic orders and they represent which metabolites are related to other metabolites. Take note that $g_{ij} = 0$ or $h_{ij} = 0$ implies that metabolite or chemical X_j is not related in the production or degradation of X_i. An S-system formulation, like Equation 1 is one of the structures proposed by Voit and

Almeida [2004] in modeling biochemical systems. Each differential equation represents a reaction. Each reaction involves the production or degradation of a metabolite. A system is usually consists of n reactions, each one associated to a metabolite/chemical component that is part of the biochemical system.

The system of differential equations then represents a biochemical system, So, given a biochemical system, our goal is to find an S-system formulation that can represent the system. This problem can be divided into two problems: (1) Network inference problem, and (2) Parameter estimation problem. The network inference problem involves determining which component interacts with another component in the system. This can be considered as a combinatorial optimization problem since the challenge is to find which combination of metabolites is needed in the production and/or degradation of a given chemical component. This is a hard problem since given n metabolites there are $O(2^n)$ possible combinations of metabolites and so a brute force search of all possible combinations is ineffective. Parameter estimation is the problem of determining the actual values of the kinetic and rate constants.

Unlike the network inference problem, parameter estimation is a continuous problem since it involves the estimation of real values. A parameter set consisting of rate constants and kinetic orders represent a unique biochemical system. The problem of determining the model is reduced to a problem of determining which parameters will be plugged in to the model so that it will produce the values of the $X_j's$. This is considered as an inverse problem, because we are given the time series data of the concentrations of the various chemicals involved in the system, and we need to find what kind of model will produce such results. This inverse problem is difficult because we need to estimate a large number of parameters, given only the time series of the concentration of metabolites. In general, given a set of n metabolites, we need to estimate a $2n$ rate constants and $2n(n-1)$ kinetic orders.

Basically, the two problems can be reduced to a minimization problem. We want to find which combination of metabolites/chemical components and parameters will produce the corresponding time series data that will have as small error as possible with the actual time series data. Since these are minimization problems, we will need to define a fitness or cost function. Given a parameter set q, this parameter set is consists of the rate constants and the kinetic constants, we define $\overline{X_i}(q)$ to be the time series data produced by the parameter set. The cost function will be defined as:

$$F(q) = \sum_{i=1}^{n} \frac{(X_i - X_i(q))^2}{X_i} \quad (2)$$

This cost function (Equation 2) is similar to the one used in [del Rosario et al., 2008].

We propose the use of the ant colony optimization algorithm in solving the network inference and parameter estimation problems.

Ant colony optimization algorithm

It is known that ants have the behavior of leaving a certain kind of chemical called pheromone. This chemical acts as a tool for ants to communicate, so that they will know which paths are being used by the other ants. The more pheromone concentrated in the area means that there were more ants which used the path, and hence it should be a better route for them to find their food. This idea was the one used by Marco Dorigo, in developing a novel optimization algorithm now known as the ant colony optimization algorithm [Dorigo and Caro, 1999; Tsutsui et al., 2005].

There are two basic forms of the ACO. The discrete type is used for combinatorial optimization problems, while the continuous type is used for continuous problem. The basic principle behind the ACO is given by the pseudo code in Figure 1. In the ACO, the solutions are called paths. The first step is to first generate random solutions or paths. Then the cost values of the path will be computed. Then the pheromones will be updated based on the cost of the solutions/path. Then the next time the new generation of solutions will be produced, the production of the solutions will be biased towards the solutions with higher pheromone values.

In some ACO algorithms, pheromone values of previous solutions will be reduced by incorporating an evaporation routine. This is important so that bad solutions will be eliminated thus reducing the search space. The algorithm will terminate once the termination condition is satisfied. Usually, the termination condition is based on the cost value of the best solution.

Discrete ACO

Two things can characterize the discrete ant colony optimization algorithm:

1. The probabilistic transition rule to determine the direction of each of the ants.
2. The pheromone update mechanism.

In solving the traveling salesman problem, the algorithm considers a set of n solutions and performs several iterations until a certain termination condition is reached. For the TSP, a solution to the problem is any random tour that does not pass any city more than once and that terminates at the starting point.

For the first iteration, the probability that a node will be selected is just the same as the other nodes. While

```
1 procedure ACO_MetaHeuristic
2 while (not_termination)
3     generateSolutions()
4     daemonActions()
5     pheromoneUpdate()
6 end while
7 end procedure
```

Figure 1. ACO pseudo code.

making a tour, each ant will leave pheromone trails on its path. The amount of pheromone that will be left on the path will depend on the distance of the path travelled. The shorter the distance, the more the pheromone that will be left, while the longer the distance, the lesser the amount of pheromone that will be dropped.

For the succeeding iterations, the selection of the nodes will depend on the amount of pheromones that were left in the trail. The more pheromone that were left the higher the probability that a certain path will be chosen. The amount of pheromone that will be left on ground will be updated depending on the cost of value of the solution. Consider a general optimization problem where, given a set of discrete choices, the goal is to find the best combination of choices depending on a given cost functions.

Continuous ACO

A similar approach is used for solving continuous optimization problems. We will be adapting the continuous ACO method that was proposed in [Dorigo and Socha, 2007; Tsutsui, 2006]. The objective is to find the parameter set q , that will minimize the cost function (Equation 2). We have to note, that the elements of the q are real numbers. The continuous ACO algorithm solves this problem by first generating a set of m random vectors. These solutions were generated from a Gaussian distribution. The initial means and the variances of the Gaussian distribution are arbitrary.

After the generation of the vectors, the cost of each vector will be computed and the resulting vectors will be sorted based on the cost values. The vector that has the best (lowest) cost value will then be used for the next iteration of algorithm. New solutions will be generated in the next iterations with the elements of the best vector, as the mean of the Gaussian distributions. The process will be repeated, until a certain threshold value, based on the cost function, is obtained.

METHODOLOGY

Given an initial data set (time courses), the objective is to find a model that produces the given data. The data set will be consists of time courses of the metabolites that are contained in the system that we are modeling. If that data is from a real system, then it is expected that the data set will contain some noise. Hence, data smoothing can first be done to denoise the data. Standard data smoothing techniques can be done on the data [del Rosario et al., 2008].

In the absence of a real system, a system of differential equations can be used to generate an artificial data set. Noise can be added in the artificial data set. After generating the time course, the differential values at the left side of Equation (1) needs to be approximated. These differential values can approximated by:

$$\frac{dx_i}{dt} \approx \frac{X(t_1) - X(t_2)}{t_1 - t_2} \tag{3}$$

To reduce the computational complexity of solving for the parameters, Voit and Almeida [2004] proposed the use of decoupling. Decoupling is the process where each differential equation is replaced by an algebraic equation. This is done by replacing the left hand side of Equation (1) with Equation (3). After converting the system into an algebraic system, each equation will be treated as an independent equation from the other equations. Network inference and parameter estimation can now be done per equation and not as a system.

We will then use the two-phase ACO for the network inference and parameter estimation of biochemical systems.

Network inference

Given the time courses, the discrete ACO can be used to predict the relationship among the metabolites. We will present in here our algorithm for the network inference problem. This algorithm is based on the ACO for the knapsack problem. We will call this algorithm ACOI. Take note that since the system is decoupled, we

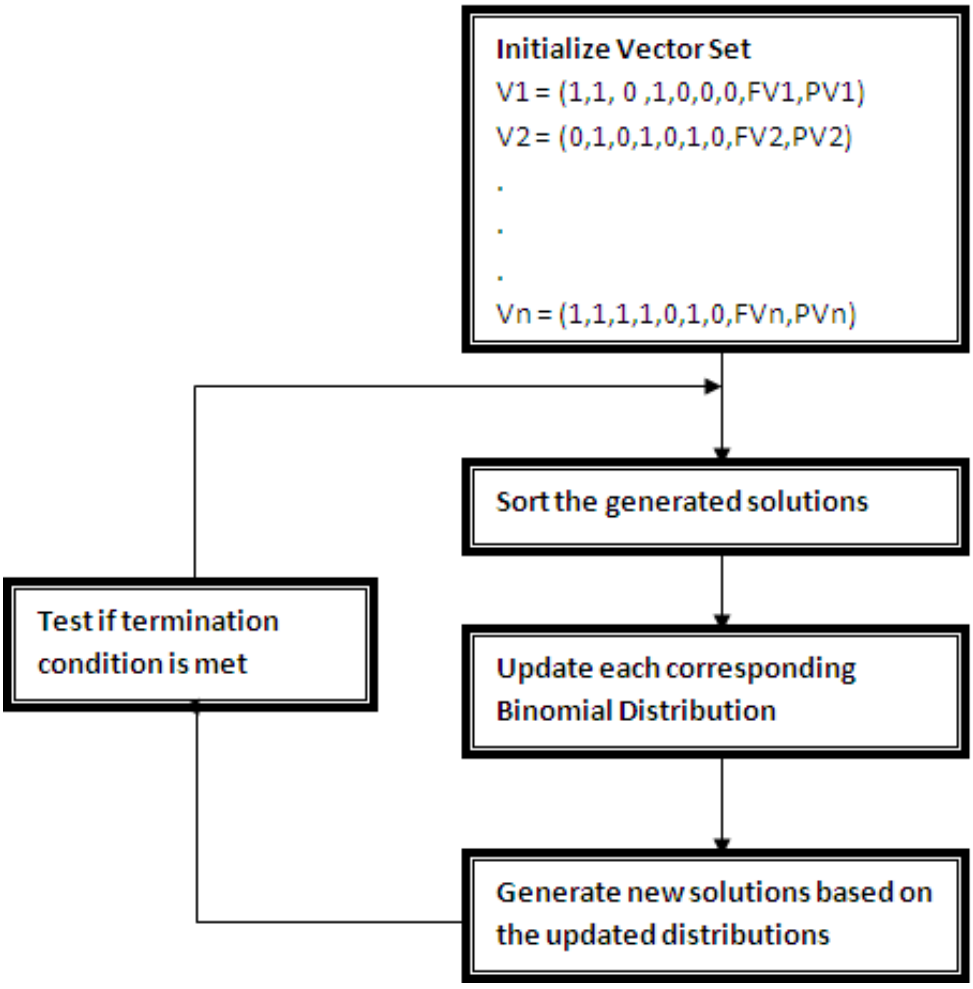

Figure 2. ACOI algorithm.

will be working with each equation separately. Since each equation represents a reaction that yields to the production or degradation of a metabolite, we can associate each equation to a metabolite. We let Y_i to be the metabolite associated to equation/reaction i. A solution vector v for the network inference problem is a $2n+2$ vector. The first n elements of *S* are associated to the production of Y_i. If the jth element of the vector is 1, then it means that the jth metabolite is involved in the production of Y_i. If the value is 0, then this means that jth metabolite is not involved in the production of Y_i. The $(n+1)th$ to $2nth$ elements of *S* are associated to the degradation of the Y_i. If the $(n+j)th$ element of the vector is 1 then it means that the jth metabolite is involved in the degradation.

Of Y_i. If the value is 0, then this means that jth metabolite is not involved in the degradation of Y_i. The $(2n+1)th$ element is the cost value (*CV*) of the solution. This is computed using Equation 2, while the $(2n+2)th$ element is the pheromone value. The pheromone value, *PV* is computed as:

$$PV = \frac{1}{CV+1}$$

Figure 2 shows a flowchart of this revised algorithm:

Initialization

In solving equation i, we will allot a $(2n+2)\times m$ matrix. We will call this matrix as *S*. Each column vector of *S* is a potential solution vector for equation i. Each row *K*, $k = 1, 2 \ldots 2n$ of *S* is assigned with a binomial distribution with probability constant p_k. Set $p_k = 0.5$. We used p_k to be initially be equal to 0.5 since we want that there will be an equal probability between getting a 1 (the metabolite is involved in the reaction) or 0 (the metabolite is not involved in the reaction). For each column vector, generate its first $2n$ elements by using the binomial distribution $B(p_k)$. Since $p_k = 0.5$ then it is expected that there is an equal chance of getting a 1 or a 0. The $(2n+1)th$ element of each column vector is the cost value of the vector, while the $(2n+2)th$ element is the pheromone value (Equation 3).

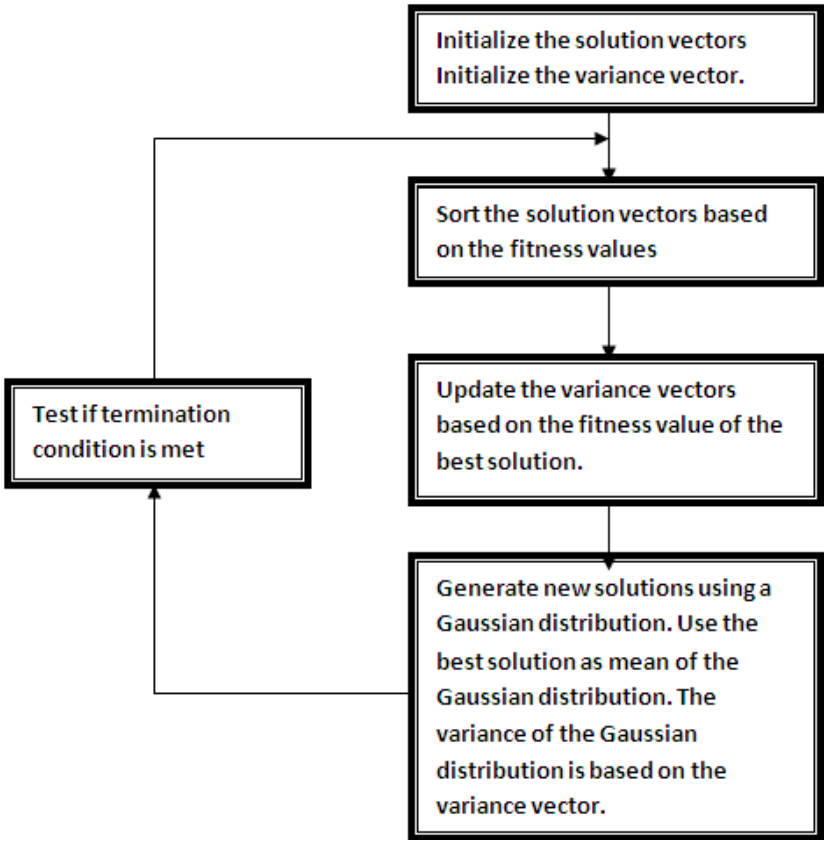

Figure 3. ACO II algorithm.

Sorting of solutions

After generating S , we sort the columns based on the pheromone value of the columns. We then get the upper 10% of the solution. The use of 10% is just an arbitrary choice. These upper 10% serves as the seed for the generation of the next solutions.

Probability update

Each p_k is updated by looking at the kth elements of the seed vectors. If the kth element is 1, then we add an arbitrary value ε to p_k. If the kth element is 0, then we subtract ε from p_k. The update routine is made so that p_k will tend to produce solutions with high pheromone values. We generate new solutions using the updated values of p_k.

Evaporation of pheromones and termination of the ACOI

For each iteration, the stored pheromone values are depreciated using the following equation:

$$PV = \gamma PV$$

where the evaporation constant is γ . This eventually allows the algorithm to delete the combinations with low pheromone value, so we can save space, and increase the rate of convergence, since we will be searching on a smaller space. We have to remember that γ should not be small enough that it will lead to the immediate termination of the algorithm. Once the cost values of the best solution is less than the threshold, then the algorithm will terminate.

Parameter estimation

We now present our ACO for the parameter estimation problem. We call this algorithm as ACOII. This algorithm is based on the ACO for continuous problems [Dorigo and Socha, 2007; Tsutsui, 2006]. The algorithm is almost similar with ACOI. The main difference is the use of a Gaussian distribution, instead of a binomial distribution. We also put special treatment on the variance that will be used in the Gaussian distribution. Just like in the network inference, the equations will be solved separately. A solution for equation i to the parameter estimation problem is called the parameter set q , where $q = \{\alpha, \beta, g, h\}$. The parameter set is consists of approximations for the kinetic orders and rate constants of equation or reaction i.

Figure 3 shows a flowchart of the ACO for continuous search spaces.

Initiailization

Just like in the network inference problem, we use a matrix, *M* with size $(2n + 3) \times m$. Each column of *M* represents a parameter

set. The first $2n$ rows represent the kinetic orders, rows $2n+1$ and $2n+2$ represent the rate constants. The $(2n+3)th$ row represents the cost value of the solution. To generate contents, we first use uniform distribution. We assume that the range of the kinetic orders are $(-2,2)$. while the range of the rate constants are $(-10,10)$.

Sorting

We then sort the columns of *M* based on the cost of the solution. The one with the smallest cost will be considered as the best solution and will be represented by vector $\bar{s}$.

Generation of new solutions

We replace the contents of *M*. For each column vector of *M*, we generate a corresponding solution vector *S*. To generate the elements of *S*, we use a Gaussian distribution $N(s_i, v_i)$. The ith element of *S* uses a Gaussian distribution with mean, $\overline{S_i}$ and variance v_i . Then we compute for the cost of *S*. If the cost value of *S* is lower than the cost value of the corresponding solution vector in *M* then we replace the vector's contents of *S*. otherwise, the row vector will stay in *M.* This enables us to keep the good solutions and remove the bad solutions from *M*.

Role of the variance

The variance vector plays a very crucial role in the algorithm. The variance vector determines the rate of convergence of the algorithm. A fast depreciating variance vector might make the solutions converge to a local minimum. On the other hand, a slow depreciating variance vector prevents the algorithm from converging to a solution. Traditional continuous ACO algorithms depreciate the variance by either subtracting a constant (linear depreciation) or by multiplying the variance by an arbitrary constant that is less than 1 (exponential depreciation).

A novelty that is introduced in this work is to make the variance dependent on the cost value of the best solution. Currently, thealgorithm imposes a simple direct proportionality relationship between the variance vector and the cost value. If a solution has a higher cost value, then we want the variance to be high too so that ACOII will be able to find other solutions. On the other hand, if a solution has a lower cost value, then we want ACOII to converge immediately, thus requiring a smaller variance. For this purpose, we use a fraction of the cost value of the best solution as the variance of the Gaussian distribution, v_i .

Termination

The algorithm will terminate if the cost value of the best solution is smaller than the threshold value for cost.

Convergence to a local minimum

Converging to a local minimum is a common problem of optimazation algorithms. This means that the algorithm has converged to a good solution, but it is not the best solution. This is especially evident if we make our variance vector proportional with the cost value. A low cost value will mean a low variance vector, but if the solution vector corresponding to the low cost value is not the best solution, then the algorithm will converge to a local minimum.

In order to solve this issue, the researchers introduce a novel innovation to standard ACO algorithms. We will call this as the "jumping ants subroutine". This is a simple innovation to the traditional ACO to help the algorithm find other solutions other than the global minimum.

The jumping ants subroutine works as follows:

1. Set two threshold values, a threshold for the cost value and a threshold for the variance value.
2. If the cost value of the best solution is lower than the threshold for the cost solution, then the algorithm terminates.
3. Since the variance of ACOII constantly depreciates, it will occur that the variance will be less than the variance threshold value. If it is less than the threshold value, but the cost is still higher than the cost threshold, then ACOII will assign the initial value to the variance. This means that the algorithm would look for other solutions.

The graph in Figure 4 illustrates this innovation. It can be seen that the value of the variance already decrease to very small value, but the corresponding cost value is still high. This is a case where the solutions produced by the algorithm will converge to a local minimum. Hence, what ACOII will do is, it will assign a set the variance to its initial value so that the algorithm will be able to find other solutions, and so that the algorithm will not be trapped in a local minimum.

RESULTS AND ANALYSIS

We implemented our algorithm using Matlab 2009, using a 2.2 GHZ computer. We tested two test networks that were proposed in Voit and Almeida [2004]. We used standard networks in our tests. We will be testing the VA04 network given by the following system of differential equations

$$\dot{X_1} = 20{X_3}^{-0.8}X_5 - 10{X_1}^{0.5}$$

$$\dot{X_2} = 8{X_1}^{0.5} - 3{X_2}^{0.75}$$

$$\dot{X_3} = 3{X_2}^{0.75} - 5{X_2}^{0.5}{X_3}^{0.5}$$

$$\dot{X_4} = 2{X_2}^{0.5} - 6{X_4}^{0.8}$$

$$\dot{X_5} = 0.9$$

with the following initial values: $X_1 = 5.6$, $X_2 = 3.1$, $X_3 = 2.9$, $X_4 = 3.1$. We also tested the HS96 network given by the following system.

$$\dot{X_1} = 5X_3X_5^{-1} - 10{X_1}^2$$

$$\dot{X_2} = 10{X_1}^2 - 10{X_2}^2$$

$$\dot{X_3} = 10{X_2}^{-1} - 10{X_2}^{-1}{X_3}^2$$

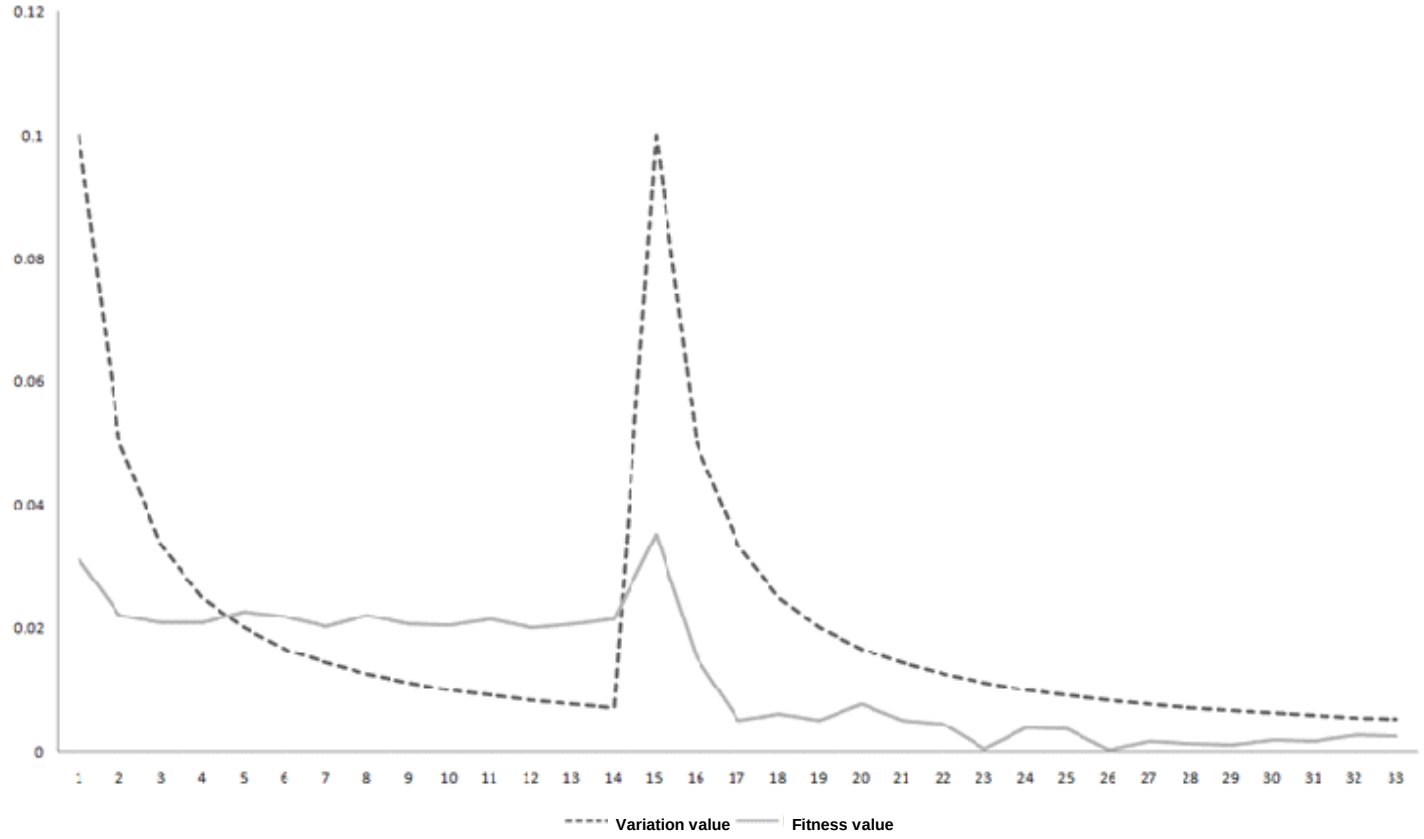

Figure 4. Local minimum.

$$\dot{X_4} = 8{X_3}^2{X_5}^{-1} - 10{X_4}^2$$

$$\dot{X_5} = 10{X_4}^2 - 10X_5$$

with the following initial values: $X_1 = 0.7$, $X_2 = 0.12$, $X_3 = 0.14$, $X_4 = 0.16$ and $X_5 = 0.18$

Network inference results

We first performed network inference on the test networks. The main objective in performing the network inference is to reduce the search space. We do not intend to find the final structure of the system, but rather, we want to reduce the number of metabolites that will be considered for the parameter estimation stage.

The Tables 1 and 2 are the result of the ACOI algorithm for VA04 and HS96. The tables show the number of times a metabolite/component is eliminated after 100 runs. It shows that the metabolites that are eliminated the least number of times are the actual metabolites that are involved in the reaction. The biggest problem in ACOI is when the algorithm eliminates a metabolite that is supposed to be part of the reaction. Once this is done, then the resulting parameter set will be significantly different to the true parameter set.

The results show that the discrete ACO can be used in the network inference problem, but there is still a relatively high rate of error (~10%) where it eliminates metabolites that are supposed to be part of the system.

Accuracy of results

We then tested the accuracy of the resulting time series data. The results in Figure 5 show our results in VA04, while the results of HS96 are shown in Figure 6. The results on both tests have shown that once ACOII terminates, the parameters produced by the algorithm will produce results that are very close to the actual concentration of the metabolites/chemical component. This is expected since termination of ACOII is based on the cost value of the parameters. The graph shows that the cost value after termination is already very small.

We also tested how close the resulting parameter set was with the actual parameter set. Due to the nonlinearity of the differential equations, it is possible that the parameter set producing the actual concentration of metabolites is not unique. Using 100 runs, we counted how many times we hit the correct values of the parameters in both VA04 and HS96. Our results have

Table 1. Network inference for VA04.

Network inference for VA04										
	Production					Degredation				
	X1	X2	X3	X4	X5	X1	X2	X3	X4	X5
X1	87	62	12	88	3	9	93	45	21	89
X2	12	90	51	41	73	88	11	82	64	39
X3	90	15	89	65	44	43	29	2	3	35
X4	92	19	84	82	82	32	43	49	9	90

Table 2. Network inference for HS96.

Network Inference for VA04										
	Production					Degredation				
	X1	X2	X3	X4	X5	X1	X2	X3	X4	X5
X1	89	34	10	32	13	6	56	67	98	99
X2	8	89	98	88	71	53	9	43	81	45
X3	19	8	66	91	97	93	9	2	78	88
X4	92	91	3	97	7	94	92	97	14	97
X5	87	65	71	12	65	64	32	21	29	11

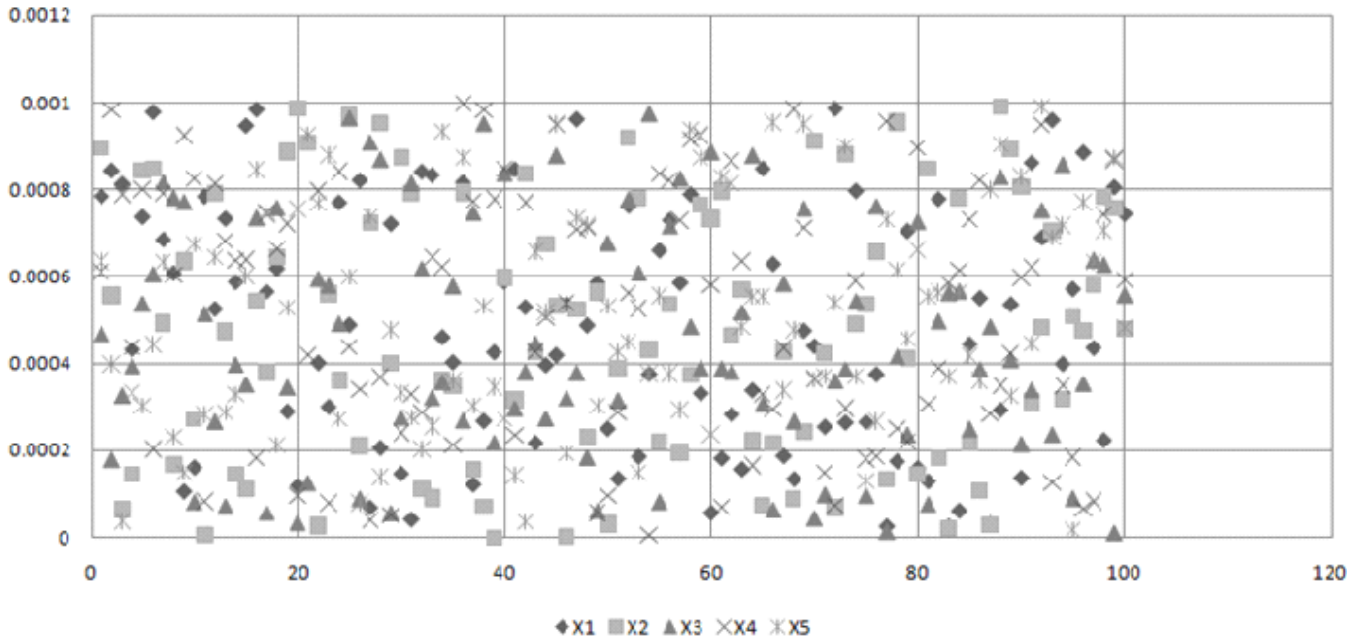

Figure 5. Fitness value results of VA04.

clearly shown that rate constants are more consistent, while the kinetic orders are more vulnerable and have a higher chance of having a different value.

Table 2 shows the percentage of getting the correct parameter values. Since the model depends on the rate constants rather than on kinetic orders, the table shows that the discovery of the rate constants is easier and is more accurate, since the value of the kinetic orders are small. Table 3 shows the % of accurate results that were obtained based on the model, and the original model.

Convergence of solutions

We recorded the number of iterations needed before ACOII terminates. Figures 7 and 8 shows that the

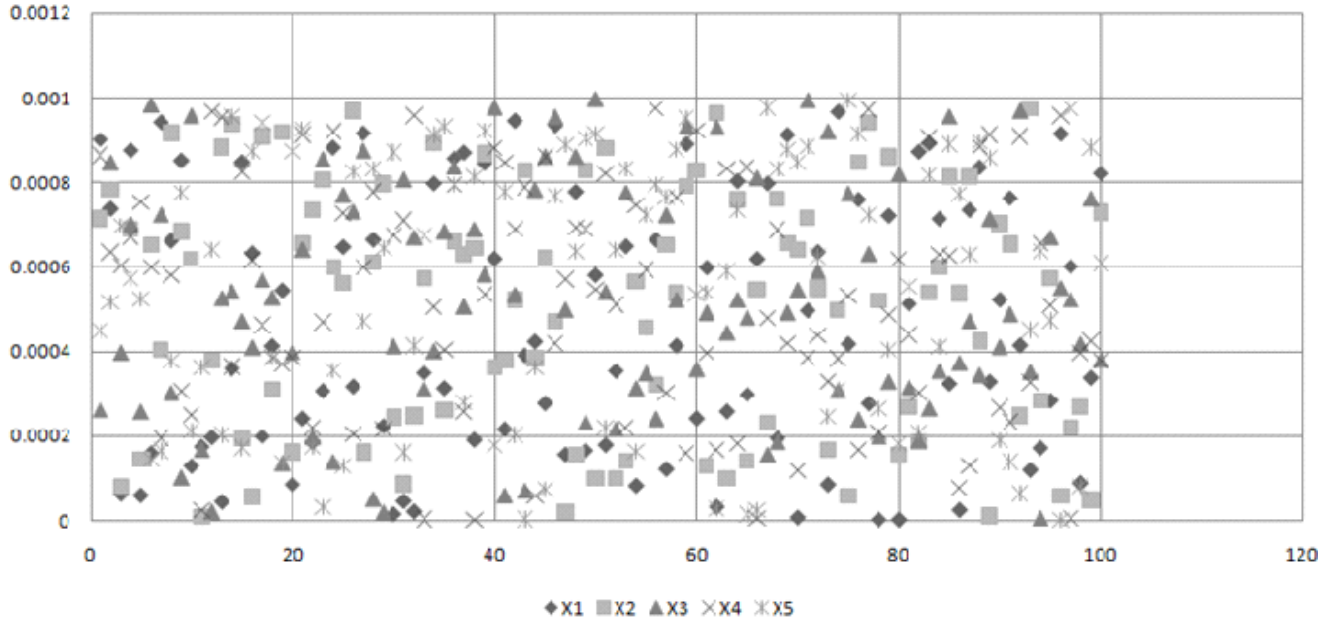

Figure 6. Fitness value results of HS96.

Table 3. Accuracy of parameters.

Parameters	Kinetic orders (%)	Rate constants (%)
VA04	59	83
HS96	67	72

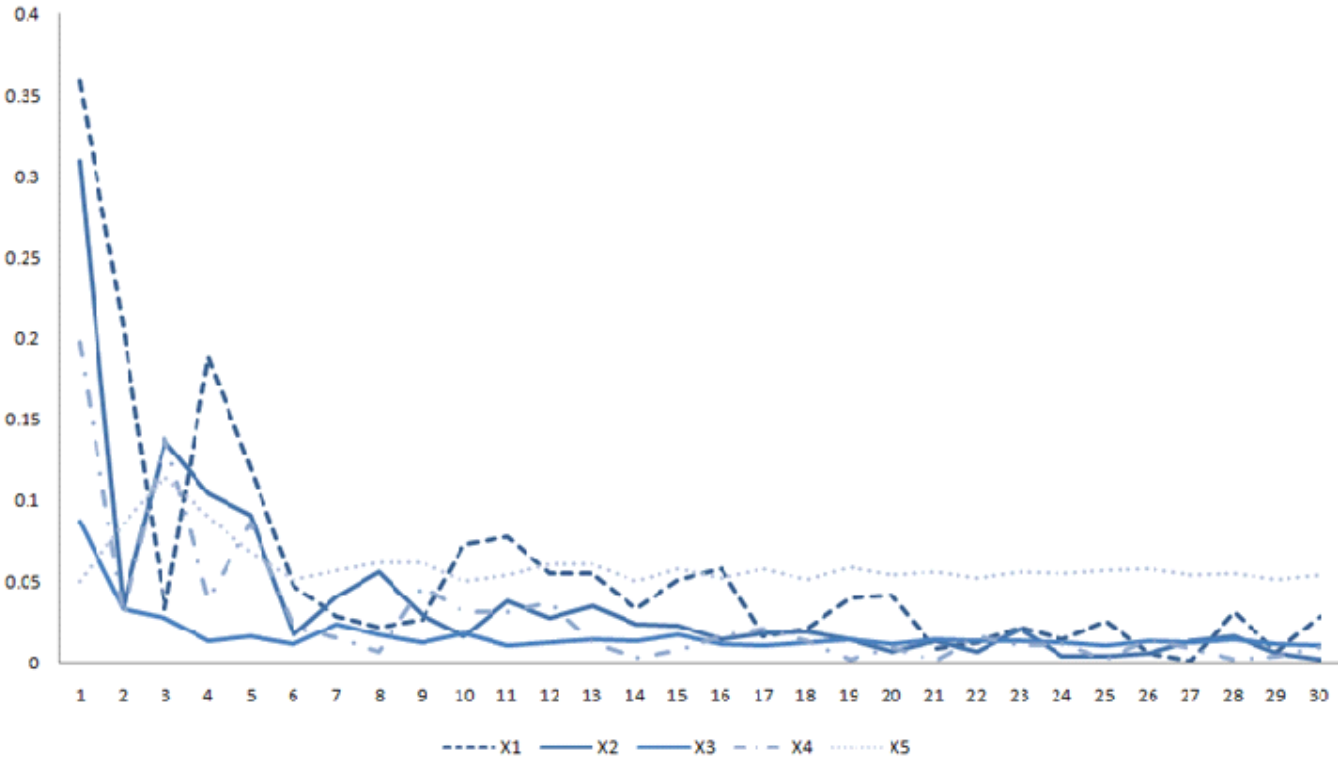

Figure 7. Convergence of solutions (10 ants).

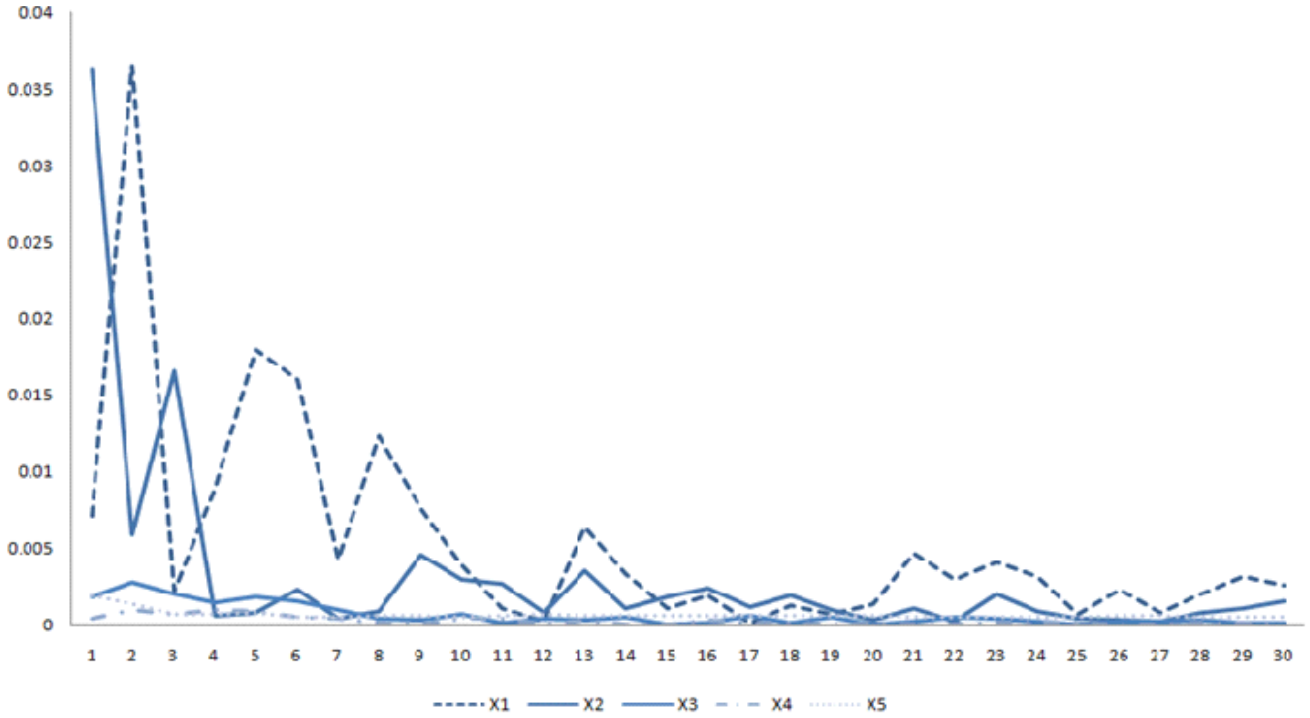

Figure 8. Convergence of solutions (100 ants).

convergence is partly based on the number of ants that were used. We are only presenting our results for VA04, since we obtained the same results as with HS96. We fixed the following parameters:

Cost threshold = 0.0001
Variance threshold = 0.00001

We fixed those two parameters since it is a trivial fact that the smaller the threshold values are, the longer the algorithm will terminate. We varied the number of solution vectors and recorded how many times the solution will fall to a local minimum. Our results have shows that, using more ants will make the solution converge slower (in terms of CPU time), but it approximates the model results closer.

Conclusion

We were able to show that the ant colony optimization algorithm can be used in the modelling biochemical system. The results of the experiments show that the two phase ACO can produce a model that will yield the correct time series data for the concentration of the different metabolites that are involved in the system. We were able to introduce novel innovations such as a relation between the cost value and the pheromone value, variance depreciation based on the cost value, and the use of the variance threshold and the cost threshold in avoiding converging to local minima.

REFERENCES

Dorigo M, Di Caro G (1999). The Ant Colony Optimization Metaheuristics, New Ideas in Optimization, McGraw-Hill, New York, pp. 11 - 32

Dorigo M, Socha K (2007). An Introduction to Ant Colony Optimization**,** Approximation Algorithms and Metaheuristics.

Tsutsui S (2006). An Enhanced Aggregation Pheromone System for Real-Parameter Optimization in the ACO Metaphor, ANTS Workshop.

Tsutsui S, Pelikan M, Ghosh A (2005). Performance of Aggregation Pheromone System on Unimodal and Multimodel Problems, Proceedings of IEEE.

Voit EO, Almeida J (2004). Decoupling Dynamical Systems for pathway identification of metabolic profiles, Bioinformatics, Pp. 1670 - 1681

del Rosario R, Echavez M, de Paz M, Zuniga PC, Bargo MC, Arellano C, Pasia JM, Naval P, Voit E, Mendoza E (2008). The MAD Man Benchmarking - User's Guide, Proceedings of the International Conference of Molecular Systems Biology.