Reverse Engineering of Gene Regulatory Networks Using Dissipative Particle Swarm Optimization

Leon Palafox, *Member, IEEE,* Nasimul Noman, and Hitoshi Iba, *Member, IEEE*

***Abstract*—Proteins are composed by amino acids, which are created by genes. To understand how different genes interact to create different proteins, we need to model the gene regulatory networks (GRNs) of different organisms. There are different models that attempt to model GRNs. In this paper, we use the popular S-System to model small networks. This model has been solved with different evolutionary computation techniques, which have obtained good results; yet, there are no models that achieve a perfect reconstruction of the network. We implement a variation of particle swarm optimization (PSO), called dissipative PSO (DPSO), to optimize the model; we also research the use of an L1 regularizer and compare it with other evolutionary computing approaches. To the best of our knowledge, neither the DPSO nor L1 optimizer has been jointly used to solve the S-System. We find that the combination of S-System and DPSO offers advantages over previously used methods, and presents promising results for inferencing larger and more complex networks.**

***Index Terms*—Gene regulatory networks (GRNs), particle swarm optimization (PSO), reverse engineering.**

I. Introduction

DIFFERENT groups have linked diseases such as cancer and diabetes to genetic regulations in organisms; thus, there is a great interest in modeling the interactions in a gene regulatory network (GRN). Using microarrays, we can measure the dynamics among genes in a cell, also called gene expression. A microarray experiment surveys the patterns of gene expression by monitoring their expression levels under various environmental conditions [1]. Using the experimental results, we obtain time series data, which represents the regulation process among genes (Fig. 1).

The researchers have modeled GRNs using different tools, such as Boolean networks [2]–[4], which model the gene's states as members of a binomial domain, and Bayesian networks, used to assign conditional probabilities to the genes' regulation parameters [5], [6]. This paper will use a system dynamics approach, which models the network as a system of differential equations. Since we are looking to model small-to-medium networks, we will focus on the S-System model [7], an effective model for short networks. To decrease the calculation time, we used a decoupled version of the S-System [8].

The S-System, however, has many different parameters that have different optimal values for a unique set of genes. To infer the correct parameters in an S-System, researchers have used optimization techniques such as genetic algorithms (GA) [9], global optimization methods [10], and linear time-variant models [11], among others.

Most of the current approaches, however, have yet to achieve a perfect reconstruction of a small network. While some of them find all the correct regulations, they also present false regulations. Thus, doing inference in small networks remains an open problem. To the authors' best knowledge, particle swarm optimization (PSO), effectively used in other areas [12]–[14], has yet to be applied for the S-System. PSO was originally designed to optimize continuous parameters in a problem, which are exactly the kinds of parameters the S-System presents, thus making it a good fit for it.

Other works have used PSO to estimate the parameters of a network. Xu *et al.* [15] used PSO to find the weights of recurrent neural networks that described the gene inference. In this paper, instead, we use the S-System and extend the PSO by adding two variations. First, we use an L1 regularization parameter in the fitness function, which is useful if we assume that the network's parameters are sparse; we will also use a dissipative PSO (DPSO) algorithm [16], which will prevent us from ending in a local minimum.

Classic PSO has inferior performance compared with differential evolution (DE) [17], a popular approach for GRN inference. It is, however, faster than most evolutionary computation techniques to find a local minimum. DPSO seizes in the PSO convergence speed to make multiple random jumps and find the global minimum in a reasonable time. Our L1 regularization approach along with DPSO both constrain the search spaces and broads the search jumps, reducing the inference time using an unconstrained DPSO.

This paper is organized in the following way. First, we give a brief introduction to the S-System as well as its parameters, then we look at the fitness function and the decoupled S-System. We then present a brief introduction to DPSO and the use of L1 regularization over the parameters. Then we will present the results we obtained using the artificial network NET1 and two sets of real data. Finally, we will do a preliminary inference on a medium-sized network to test the algorithm in larger networks.

Manuscript received December 15, 2011; revised May 10, 2012 and August 1, 2012; accepted August 21, 2012. Date of publication September 12, 2012; date of current version July 25, 2013.

L. Palafox is with the Department of Electrical Engineering, University of Tokyo, Tokyo 113-0033, Japan (e-mail: leon@iba.t.u-tokyo.ac.jp).

N. Noman and H. Iba are with the Graduate School of Information Science and Technology, University of Tokyo, Tokyo 113-8656, Japan (e-mail: noman@iba.k.u-tokyo.ac.jp; iba@iba.t.u-tokyo.ac.jp).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TEVC.2012.2218610

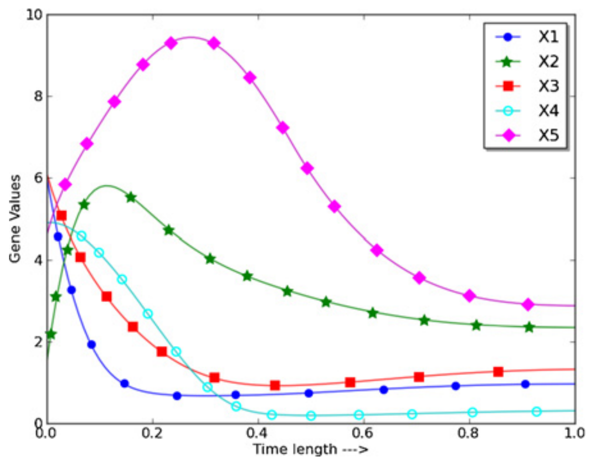


Fig. 1. Time series data of a GRN.

II. Methods

As mentioned earlier, time series data is an important tool to model gene expression. The biological networks' complexity, however, makes them difficult to describe and fit using a single function. Biochemical system theory (BST) [18]–[20] provides a mathematical framework to represent and analyze biological systems. The S-System, a popular model of BST, represents a network as a set of differential equations

$$\frac{dX_i}{dt} = \alpha_i \prod_{j=1}^{N} X_j^{g_{ij}} - \beta_i \prod_{j=1}^{N} X_j^{h_{ij}} \tag{1}$$

where X_i is the expression level of the ith gene of the network; N is the number of genes in the network; $\alpha_i, \beta_i \in \mathbb{R}_+^N$ are rate constants; and $g_{ij}, h_{ij} \in \mathbb{R}^{N\times N}$ are kinetic orders. It is worth mentioning that the kinetic orders g_{ij} and h_{ij} regulate the synthesis and degradation of X_i due to X_j.

A. Decoupled S-System

The evaluation of an S-System model for GRN has to be done using a numerical solver such as Runge–Kutta. In the Runge–Kutta method, we have the differential equation defined by

$$\frac{dx}{dt} = f(t,x) \tag{2}$$
$$y(t_0) = x_0 \tag{3}$$

where $f(t,x)$ is a nonlinear differential equation. For the case of the S-System

$$f(t,x) = \frac{dX_i}{dt} = \alpha_i \prod_{j=1}^{N} X_j^{g_{ij}} - \beta_i \prod_{j=1}^{N} X_j^{h_{ij}}. \tag{4}$$

Using a fourth-order Runge–Kutta method, we can integrate the solution as

$$x_{n+1} = x_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k4) \tag{5}$$
$$t_{n+1} = t_n + h \tag{6}$$

where the factors are defined as

$$k_1 = hf(t_n, x_n) \tag{7}$$
$$k_2 = hf(t_n + \frac{1}{2}h, x_n + \frac{1}{2}k_1) \tag{8}$$
$$k_3 = hf(t_n + \frac{1}{2}h, x_n + \frac{1}{2}k_2) \tag{9}$$
$$k_4 = hf(t_n + h, x_n + k_3). \tag{10}$$

If the networks are large, however, the numerical solutions often take a long time to calculate given that (4) depends on each of the N variables to do an update. Moreover, to optimize the parameters, evolutionary techniques, such as PSO and GAs, need to evaluate the S-System for multiple candidates and multiple iterations. As the networks grow larger, so does the computation time. Since the complexity of the Runge–Kutta method is inherent to the fact that it has to solve a simultaneous system of equations, by decoupling it and being capable of calculating independent genes, we can decrease the calculations. To this purpose, we use the data collocation approach, proposed by Tsai and Wang [8], to reduce the calculations.

Tsai and Wang used a linear Lagrange polynomial to reduce the calculation's time by introducing the experimental data when they used the Runge–Kutta method for solving the S-System of a network. Thus, we define each new step in the numerical solution of the S-System as

$$x_{n+1} = x_n + 0.5\eta(g[x_{n+1,\text{exp}}, \theta] + g[x_n, \theta]) \tag{11}$$

where the $x_{\text{exp}}(t)$ on the right side of the equation corresponds to the value of the experimental values at time t and $g[x,\theta]$ is the equation (1) evaluated at x with parameters θ. The parameter η is a smoothness rate that is to be set small so that the approximation does not overshoot. For notation convenience, we will name θ to the set of all parameters $\{g_{ij}, h_{ij}, \alpha_i, \beta_i | i, j \in 1 \ldots N\}$ in the S-System.

We use (11) instead of the update in (5) to decrease the calculations and the inference time. By using the experimental values for the solution of $x(t)$, the system can solve each gene's solution independently, instead of performing the joint solution of all genes in the classical Runge–Kutta method in (4).

B. Estimation Criteria and Regularization

Since our aim is to find the best parameters θ for the network, it is necessary to formulate it as an optimization problem. Tominaga *et al.* [21] standardized the use of the mean squared error (MSE) evaluation to measure a candidate's fitness in the S-System. Thus, we define the fitness function as

$$f = \sum_{t=1}^{T}\sum_{i=1}^{N}\left(\frac{X_{i,\text{cal}}(t) - X_{i,\text{exp}}(t)}{X_{i,\text{exp}}(t)}\right)^2 \tag{12}$$

where T represents the number of time samples in the experimental data, N is the number of genes, and $_{\text{cal}}$ and $_{\text{exp}}$ refer to the calculated and experimental values of the gene expression's data, respectively.

Since different networks can have the same time series data, we divide our experimental time series in M sets. This will force the parameters θ to recreate the same dynamics at different initial points, which helps us to infer the true parameters of the network. Each of the N genes' time expression is divided on M sets, which will create $N \times M$ training sets for the system.

There are many approaches to solve this optimization problem. Noman and Iba [22] used DE and other evolutionary computation techniques, while other people used simulated annealing [23] and artificial neural networks.

In this paper, we will infer the values of the parameters θ using PSO with regularization parameters. We will implement a DPSO, an implementation that helps the system to avoid local minima. For this, it increases the entropy of the particles at random points during the optimization.

1) *Regularization:* A regularization term $\lambda E(\theta)$ is often introduced in an error function to avoid overfitting as well as to restrict the search space. It achieves this by restraining the growth of the parameters. There are different regularization terms, which we choose according to the data. Ng [24] showed that an L1 regularizer is a good choice when trying to infer sparse parameters in a logistic regression. Thus, our regularization term for each of the N genes i will take the form

$$\lambda E_i(\theta^*) = \lambda \sum_{i,j=1}^{N} |g_{ij} + h_{ij}| \tag{13}$$

where the parameters θ^* for large gene networks are sparse. Since the L1 regularization is only used on sparse parameters, we only will apply it to the h_{ij} and g_{ij} parameters, while the parameters α and β are left without regularization.

Thus, with the regularization term and the decoupling included, we get the following optimization function for each gene i:

$$f_i = \sum_{t=1}^{T} \sum_{j=1}^{M} \left(\frac{X_{i,j,\text{cal}}(t) - X_{i,j,\text{exp}}(t)}{X_{i,j,\text{exp}}(t)} \right)^2 + \lambda E_i(\theta^*) \tag{14}$$

where M are the different time series in which we divided the data and $X_{i,j,\text{cal}}(t)$ will be calculated using (11). Since we used decoupling, we can calculate this value for each individual gene.

The problem then has two objective functions, where we are constraining the parameters $\{h_{ij}, g_{ij}\}$ to be small and at the same time to have the best MSE fitness for the original time series data.

C. DPSO

PSO is an algorithm widely used in different applications, from solving simple optimization problems to analyzing human activity recognition systems [25].

Like GAs, PSO is based on nature's self-organization. It tries to mimic the way groups of animals behave. Ducks and fish, e.g., are capable of going toward a goal, while keeping a group structure that improves their orientation. The PSO community refer to these groups as swarms.

In PSO, a swarm is composed of particles, each of which can record its best position, and the swarm's best position. Each particle also has a velocity, which we update to grow closer to the best positioned particle in the swarm.

We define each particle p_i at time $t \in (0, T-1)$ as $p_i(t) \in \mathbb{R}^N$, where T is the maximum number of iterations and N is the dimension of the feature space. The variables that control each particle at time t are its position $x_i(t)$ and its velocity $v_i(t)$.

Each particle's velocity and position will change according to

$$\begin{aligned} v_i(t+1) &= w \cdot v_i(t) + C_1 \cdot \varphi_1 \cdot (P_{li} - X_i(t)) \\ &\quad + C_2 \cdot \varphi_2 \cdot (P_G - X_i(t)) \end{aligned} \tag{15}$$

$$X_i(t+1) = X_i(t) + v_i(t+1) \tag{16}$$

where P_{li} is the best position for particle i, P_G is the best position the swarm has obtained, and w is the inertia factor, which controls the speed at which the particles adapt. C_1 and C_2 are random variables that control the dependence on the global closeness and the φ_1, φ_2 factors are manually set variables to control the swarm's convergence, which is usually set at 2 [26].

Classic PSO, however, will often reach a local minimum as its final solution. Xie *et al.* [16] proposed a variation to the classic PSO, where they added a dissipative property to the particles. They defined dissipation parameters that will restart the system at random iterations. This restart will allow the PSO to search for other optimum parameters that may have a better solution for the system. The dissipative equations, evaluated at each iteration, are

$$\text{IF}(\text{rand}() < c_v) \Rightarrow v_i = \text{rand}() * v_{\max} \tag{17}$$

$$\text{IF}(\text{rand}() < c_l) \Rightarrow x_{id} = \text{Random}(l_d, u_d) \tag{18}$$

where c_v and c_l are numbers between 0 and 1. We have to set small numbers to these variables, so we have few restarts, allowing each new iteration to reach a new optimum. Variables $v_{\max}$ and l_d, u_d are the particle velocity limit, and the lower and upper bound for the search space, respectively.

For the S-System, we will define each particle of the swarm with a variable vector p_i, composed of the parameters $\theta_i \in \{g_{ij}, h_{ij}, \alpha_i, \beta_i | i, j \in 1 \ldots N\}$.

III. Results and Discussion

In this paper, we will evaluate the inference algorithm's performance by doing inference over artificial and real networks. Since the actual parameters are unknown for real networks, we use artificial networks to validate our algorithm's proficiency to infer a set of given parameters. All the experiments were done using a computer with an Intel© i7 processor and 8 GB of RAM. The operating system used was Ubuntu's 10.10 release.

A. Inference in an Artificial System With Noise-Free Data

For the first set of experiments, we used the artificial network NET1 (Table I) [27], [28] to test the algorithm's effectiveness to find the parameter values in a known regulatory network. This network is a small five-gene network with simple interaction dynamics.

Earlier approaches [10], [11], [29], [31] used this network to validate their algorithms' inference performance. Thus, we

TABLE I
S-System Parameters for Artificially Generated Noise-Free Data, i is the Gene

i	α	g_{i1}	g_{i2}	g_{i3}	g_{i4}	g_{i5}	β	h_{i1}	h_{i2}	h_{i3}	h_{i4}	h_{i5}
1	5.0	0.0	0.0	1.0	0.0	−1.0	10.0	2.0	0.0	0.0	0.0	0.0
2	10.0	2.0	0.0	0.0	0.0	0.0	10.0	0.0	2.0	0.0	0.0	0.0
3	10.0	0.0	−1.0	0.0	0.0	0.0	10.0	0.0	−1.0	2.0	0.0	0.0
4	8.0	0.0	0.0	2.0	0.0	−1.0	10.0	0.0	0.0	0.0	2.0	0.0
5	10.0	0.0	0.0	0.0	2.0	0.0	10.0	0.0	0.0	0.0	0.0	2.0

TABLE II
Inferred Values for the NET1 Without Noise

i	α	g_{i1}	g_{i2}	g_{i3}	g_{i4}	g_{i5}	β	h_{i1}	h_{i2}	h_{i3}	h_{i4}	h_{i5}
1	4.387	0	0	1.425	0	−0.896	9.567	1.465	0	0	0	0
2	9.324	1.789	0	0	0	0	10.562	0	2.058	0	0	0
3	10.879	0	−1.659	0	0	0	9.847	0	−1.245	1.875	0	0
4	8.326	0	0	1.896	0	−0.754	9.885	0	0	0	2.235	0
5	9.632	0	0	0	2.056	0	10.657	0	0	0	0	2.136

Small values have been truncated to zero for convenience.

will use it as well to compare this algorithm with the previous works.

1) *Experimental Setup:* To do inference in the noiseless data, we used a population of 20 particles. We used the paper defined by Shi and Eberhart [32] to decide the population size, since in PSO the population size does not have an effect on whether it reaches a minimum or not. We set a search region of $[-3, 3]$ for the kinetic orders g_{ij}, h_{ij} and $[0, 10]$ for the rate constants α and β; the search region is set in such a manner that the search will be fast to converge. We will conduct further sensitivity analysis of this setting. We set the restart parameters c_v and c_l to 0.01, to restart the system about once every 100 iterations. We ran 10 000 iterations for the PSO. The value of η in (11) is 0.05 to keep the extrapolations smooth. After testing some possible values, we set λs in (13) at 5, so the effect of the regularization impacts the optimization.

To improve over previous works [29], [31], where they used 10 time series' data, we used $M = 4$ time series with 15 samples each to do the inference in the network.

2) *Result:* In Table II, we show the best set of resulting parameters, where we truncated values less than 0.0001 to zero. The values are similar to those in Table I, thus having a successful inference of the parameters. We found the correct values of the network in most of the algorithm's runs (70%). In every experiment, however, the PSO was able to infer the correct signs of the parameters and the location of the nonregulations, albeit having some variations in the numerical values. It took around 5000 iterations to reach stability in a single net.

Table III shows a simple comparison between our method and the works by Noman and Iba [31], [29] and Kabir *et al.* [11], which have used time series to train DE and linear time variant (LTV). In the three cases, the inference was flawless for noise-free data, and as such, the only comparative advantage is the reduction of the time series data.

B. Parameters Sensitivity

1) *Population:* In this paper, we did not test the sensitivity of the population size, since it has been shown by Shi and Eberhart [32] that PSO is relatively robust for different population sizes. We have added, however, a figure to show the behavior of different population sizes [Fig. 2(b)]. It shows how, for large populations, we have a faster convergence rate, but a large population also increases the calculations.

TABLE III
Comparison of Base Results for the Noise-Free NET1

	Time Series	FPs
L1–DPSO	4	0
LTV [11]	10	0
DE [29], [31]	10/25	0

2) *Lambda:* We also tested different values of λ and evaluated its effect on the number of false positives (FPs) and false negatives (FNs). Intuitively, a large λ gives more importance to the regularization parameter than to the actual fitness function. Thus, large λs make the inference gravitate toward small parameters and thus the FNs should increase.

On the other hand, small λs tend to be similar to the unregularized function, thus not having a penalty for FPs and large values in the parameters. A small λ should present similar results to those we obtained for an unregularized PSO.

These intuitions are confirmed by Fig. 2(c). It shows that as λ increases, the FPs decrease and the FNs increase. A small λ has zero FNs, since the values are not affected by the regularizer.

The best set of λs is in the $[2.5 - 8]$ interval, and we chose the value of 5 for our inferences, since it is the closest integer number to the middle.

3) *Search Ranges:* We also analyzed the search ranges in the scheme. We used different starting search ranges for the kinetic orders and the rate constants. In the S-System model, the rate constants always have positive values, so the search ranges will only increase in the positive scale.

Fig. 2(a) shows the algorithm's robustness against large search ranges. For an increased search range for the rates and the kinetic constants, the algorithm reached convergence,

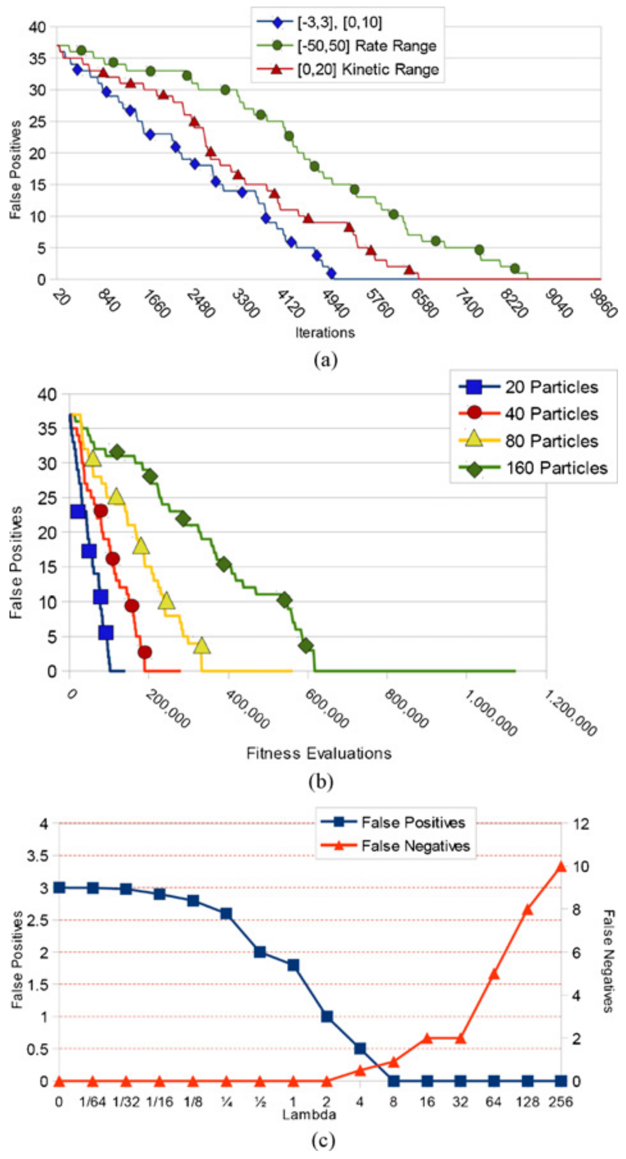


Fig. 2. Sensitivity tests for different variables of the algorithm. (a) Values of FPs for different search ranges. (b) Values of FPs' rate for different populations for the DPSO–L1 and a noiseless network. (c) Values of FPs and FNs for different values of λ.

although it took almost twice the time with the extended rate range.

The search range is specific to each problem and a similar analysis has to be done for different networks; however, these results show that we can use large search ranges and can still find good results.

C. Artificial System With Noisy Data

In the next experiments, we tested the algorithm using artificially generated data from the NET1 with noise. The noise was Gaussian White noise $N(0, \sigma)$ and we changed the percentage of noise to increase its effect in the data. We added the noise in the following manner:

$$x(t) : \text{Data} \tag{19}$$

$$\text{Noisy}(x) = x(t) + \epsilon N(0, \text{sigma}) \tag{20}$$

$$\epsilon = \text{Max}[x(t)] \times \text{Noise rate}. \tag{21}$$

As ϵ increases, the Noisy(x) loses information of the original data, and these tests are aimed to find the robustness limit of the inference method. For this paper, we show the results for $\sigma = 1$ and 5%, 15%, and 25% noise rates. We used the same search parameters as in the noise-free case. We also used the same optimization parameters for the regularization and the decoupling.

1) *Result:* In Table IV, we present the values of the inferred network's parameters. Adding 5% noise to the data affected the accuracy of the inference. Yet, most of the values were kept within an acceptable range. The noise had little effect on the zero values, since the regularizer is constraining these values to remain small. For a 15% noise ratio, the inference was still successful in finding the correct regulations. This means the PSO is robust for this amount of noise. Some of the FPs increased, which is expected with increased noise. The PSO + regularizer strength, however, lies in restricting the search to small numbers, thus decreasing the number of FPs in the inference.

In the final case, with 25% noise, the integrity of the data is hampered in a significant way. The number of FPs and FNs increased. Yet, the method can recover the correct interactions of the network. It is worth noting, though, that the previous works did not attempt a 25% error rate, which prevented us from doing a fair performance comparison. Since evaluating the algorithm's performance based on the data in Table IV is complicated, for further validation and comparison of these results, we will calculate the sensitivity S_n and specificity S_p of the network's parameters as follows:

$$S_n = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{22}$$

$$S_p = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{23}$$

where TP, FN, TN, and FP represent the true positive (TP), false negative, true negative (TN), and false positive predictions of the parameters. For this model, a TP is the existence of the right interaction between two genes—inhibition and activation—and a TN is the absence of interactions. In Table V, we show the average values of S_n and S_p of the noisy networks for different inference techniques. We see how, compared with the previous results [11], [30], [33], the PSO is very competitive with well-established optimization techniques, which proves its robustness against noise. We achieved these results using less time series data (four) than previous works (ten). One of the reasons this approximation worked better than similar approaches is because it is regularized, which, as we have mentioned, adds an extra penalty on large parameters, thus keeping small regulations small and preventing FPs from emerging even with increased noise in the data.

D. Comparison of Unregularized PSO and DPSO

To compare the effect of the regularizer over the optimization, we ran a classic PSO, a regularized PSO, a simple DPSO, and the regularized DPSO. We ran each variation 50 times and presented the average optimization runs. We did these runs in the NET1 artificial network inference problem described previously.

TABLE IV

INFERRED VALUES FOR THE NET1 5%, 15%, AND 25% NOISE

	i	α_i	g_{i1}	g_{i2}	g_{i3}	g_{i4}	g_{i5}	β_i	h_{i1}	h_{i2}	h_{i3}	h_{i4}	h_{i5}
5%	1	4.332	−0.070	−0.098	1.783	0.070	−0.568	10.235	1.235	−0.030	0.138	0.042	0.027
	2	9.299	1.549	−0.089	−0.025	−0.139	−0.139	11.263	−0.133	2.432	0.048	−0.038	−0.150
	3	10.771	0.086	−2.568	0.003	−0.145	0.010	9.076	−0.101	-1.569	2.564	−0.092	−0.062
	4	8.214	−0.144	−0.092	2.785	−0.048	−0.626	9.671	0.032	0.148	−0.087	2.568	0.039
	5	9.490	0.135	0.090	0.081	1.865	0.101	11.167	0.142	0.118	0.023	0.040	2.461
15%	1	3.987	−0.137	−0.432	2.654	0.426	-1.986	8.987	1.869	−0.086	−0.046	−0.137	0.078
	2	11.786	3.123	0.405	0.218	0.237	−0.258	11.786	−0.208	3.215	0.415	0.310	−0.461
	4	11.126	−0.020	−3.126	0.083	0.011	0.207	11.126	−0.244	-1.986	3.412	0.475	−0.369
	4	7.056	0.374	0.275	3.123	−0.318	-1.896	9.056	0.023	0.053	−0.473	1.456	−0.163
	5	8.334	0.208	0.357	−0.327	3.214	0.295	8.334	−0.038	−0.428	−0.318	0.282	3.604
25%	1	7.894	0.481	−0.759	3.879	0.946	−2.963	19.235	4.651	−0.326	0.356	0.738	0.526
	2	15.031	4.129	0.110	−0.491	−0.557	−0.423	18.605	−0.215	4.152	0.627	0.494	−0.723
	3	18.678	0.859	-4.214	−0.388	0.305	−0.042	15.065	−0.581	−2.546	4.421	−0.170	0.556
	4	6.579	−0.215	0.199	4.568	−0.635	−2.655	17.036	0.250	0.859	0.258	5.123	0.190
	5	6.598	−0.941	0.487	−0.833	4.125	−0.564	14.905	0.060	−0.106	0.785	−0.349	4.843

Small values have been truncated to zero for convenience.

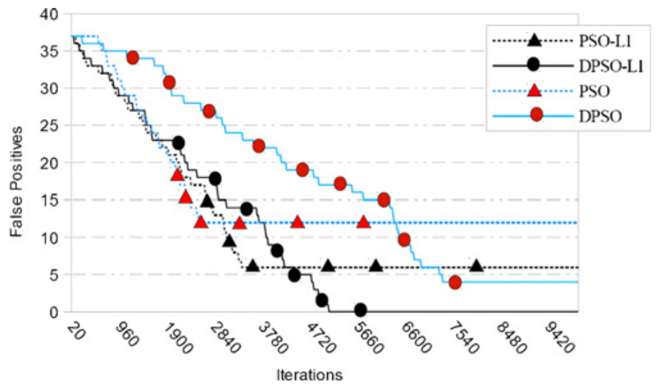


Fig. 3. FPs over generations for the noise-free network.

Fig. 3 shows the result of running these four variations over 10 000 generations. Since our objective is to find a clean network with zero FPs, we evaluate the FPs at each generation.

Fig. 3 shows how a classic PSO has a steep optimization curve; however, it quickly reaches a plateau and, because of its inherent inability to mutate, it finishes its iterations without doing any further optimization. It also shows that adding the L1 regularization term to PSO helps the final optimization; however, the L1 only restricts the sparseness of the result, not the correctness, and it also reaches a plateau at the end of the optimization. DPSO shows a good performance, with different plateaus along the optimization because of the random jumps it generates. The random jumps help the PSO to behave like a standard GA since we added the random jumping quality that mutating the population gives us. It, however, never reaches zero FPs, although it gives better results than a regularized PSO. Restricting DPSO with the L1, however, performs better than its counterparts, since it does both things at the same time, searches for multiple local optimum and restricts sparseness in the results. On average, it reached an optimal set of FPs and, since it is restricted for sparseness, it worked faster than standard DPSO.

To validate the performance over trials of the inference, we analyzed the stability for different noise rates to compare the performance between the standard DPSO and our regularized DPSO; we did the same runs and calculated the error rate and the mean for 50 runs.

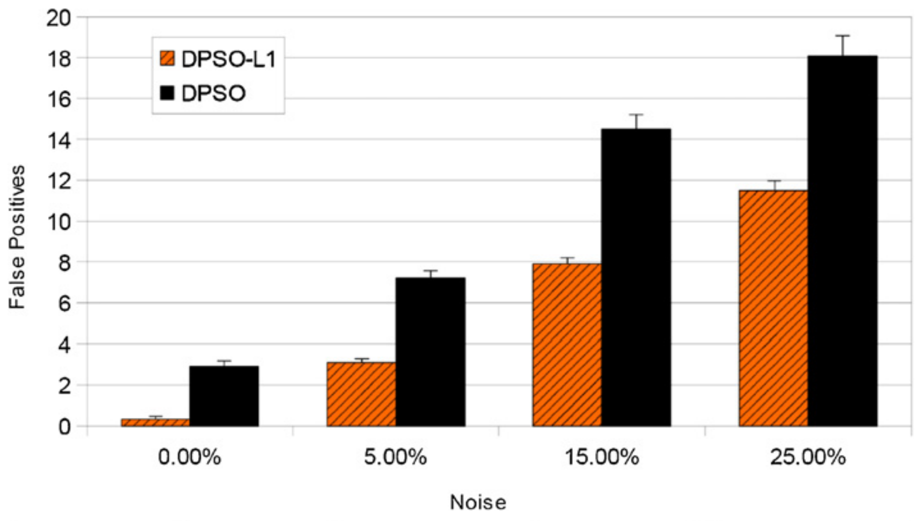
Fig. 4 shows the results for a noiseless inference as well as for 5%, 15%, and 25% data. As expected, the L1 regularization decreases the FPs; furthermore, the error, although increasing, is kept in check as we increase the noise. On the other hand, an unregularized DPSO adds a hefty number of FPs as we increase the noise and, furthermore, its error as well as the noise increases, which dampens the algorithm's performance in high noise settings.

E. *Inference in a Real System*

To test the algorithm with real data, we will present experiments using two sets of real data. First, we will analyze an S-System model of the ethanol production by yeast (*Saccharomyces cerevisiae*), used by many groups as a test system for dynamic models. Then, we will use a dataset by the Uri Alon Laboratory [34], where they report experiments using the SOS network in the *Escherichia coli* bacteria. In the former case, we have the parameters and thus we can validate our algorithm's accuracy; in the latter, we will compare it with similar works that have used the same dataset and inferred its regulations using similar techniques.

F. *Ethanol Production by Yeast*

Since this is an important and proven model for gene reconstruction [35], [36], we decided to use it as one of the benchmark tests for our data. To work with this network, we use an S-System model that has been already proposed and confirmed by previous groups [37]. This model has anaerobic and nongrowing conditions. Its carbon source is glucose and it lacks nitrogen. It has five variables: glucose (X_1), glucose-6-phosphate (X_2), fructose1,6-diphosphate (X_3), phosphoenolpyruvate (X_4), and adenosine triphosphate (ATP) (X_5). As well, the model has the following constant values in other variables: glucose uptake (X_6, 47.5 mM min^{-1}), hexokinase (X_7, 24.1 mM min^{-1}), phosphofructokinase (X_8,



Noise Rate	Error				Mean			
	0%	5%	15%	25%	0%	5%	15%	25%
DPSO-L1	0.15	0.18	0.28	0.45	0.30	3.10	7.90	11.50
DPSO	0.28	0.36	0.73	0.97	2.90	7.00	14.00	18.00

Fig. 4. Error rates and means for simulations with the NET1 network for 0%, 5%, 15%, and 25% noise.

53.9 mM min^{-1}), glyceraldehyde 3-phosphate dehydrogenase (X_9, 91.4 mM min^{-1}), pyruvate kinase ($X_1$0, 18.1 mM min^{-1}), polysaccharide storage ($X_1$1, 82.9 mM min^{-1}), glycerol production ($X_1$2, 92.4 mM min^{-1}), and ATPase ($X_1$3, 1.0 mM min^{-1}). We will follow the same approach of [38] by doing only the inference over the variable parameters, while considering the remaining constant. The S-System for the model is

$$\frac{X_1}{dt} = 1.0006X_2^{-0.0492}X_6 - 1.6497X_1^{0.5582}X_5^{0.0465}X_7 \tag{24}$$

$$\frac{X_2}{dt} = 1.6497X_1^{0.5582}X_5^{0.0465}X_7 - 0.5793X_2^{0.5097}X_5^{-0.2218}X_8^{0.8322}X_7 \tag{25}$$

$$\frac{X_3}{dt} = 0.4536X_2^{0.4407}X_5^{-0.2665}X_8 - 0.2456X_3^{0.4506}X_4^{0.0441}X_5^{0.092}X_9^{0.8547}X_{12}^{0.1453} \tag{26}$$

$$\frac{X_4}{dt} = 0.2365X_3^{0.5285}X_5^{0.0994}X_9 - 2.0892X_3^{-0.0075}X_4 0.304X_5^{0.0484}X_{10} \tag{27}$$

$$\frac{X_5}{dt} = 1.406X_3^{0.2605}X_4^{0.152}X_5^{0.0739}X_9^{0.5}X_{10}^{0.5} - 2.9437X_1^{0.1962}X_2^{0.1791}X_5^{0.2354}X_7^{0.3514}X_8^{0.2925}X_{11}^{0.0589}X_{13}^{0.297}. \tag{28}$$

1) *Experimental Setup:* For this model's inference, we used a population of 20 particles, and a search region of $[-1, 1]$ for the kinetic orders g_{ij}, h_{ij} and $[0, 3]$ for the rate constants α and β. The restart parameters and the values of the η and λ parameters were the same as for the NET1 inference, 0.05 and 5, respectively.

2) *Result:* Since the ethanol production by the yeast model is not conformed by time series data, but a set of parameters, the results we obtained are comparable with those found for the NET1 model without noise in the previous section. In the experiments, we used the real parameters to generate time series that were used to infer the network interactions as the experimental samples. In Table VI, we show the inferred parameters for this model along with the real parameters. We did 6000 iterations. The parameters we obtained using the PSO optimization are very close to those in the original system. The PSO presents, as well, a good performance compared to previous works using this model [38]. Since we used a regularization, the negative parameters were kept low and unable to grow in most iterations, thus preventing FPs from arising. We ran 10 tests and, in most experiments, the true interactions were inferred. We had an overall 89.6% sensitivity and 80.45% of specificity, which happens because some of the interactions in the model are too small, and the effect of the regularizer helped them to be treated as FNs.

TABLE V

COMPARISON OF PSO'S PERFORMANCE AGAINST PREVIOUS METHODS

	L1–DPSO		LTV [11]		Memetic [33]		DE [30]	
Noise	S_n	S_p	S_n	S_p	S_n	S_p	S_n	S_p
5%	1	0.816	1	0.769	1	0.675	1	0.730
15%	0.986	0.795	0.916	0.667	0.915	0.675	0.923	0.757
25%	0.895	0.745	–	–	–	–	0.877	0.795

G. SOS Network for E. coli

The SOS network for *E. coli* published by the Uri Alon group is a benchmark in genetic regulations; it is the time series data of different multiarray experiments. In the experiments, eight genes are expressed (uvrD, lexA, umuD, recA, uvrA, uvrY, ruvA, and polB). They irradiate the DNA with ultraviolet light, which will affect some genes, and the network will repair itself, thus expressing autoregulation. They did four experiments for different light intensities. Each experiment had 50 time steps spaced by 6 min. During each time step, they take measures of the eight genes.

TABLE VI

INFERRED REGULATIONS FOR THE ETHANOL PRODUCTION BY YEAST MODEL WITH VALUES UNDER 0.0001 TRUNCATED TO ZERO (INFERRED/REAL)

i	α_i	g_{i1}	g_{i2}	g_{i3}	g_{i4}	g_{i5}
1	0.8965/1.0006	0/0	−0.0122/−0.0492	0/0	0/0	0.0687/0
2	1.8654/1.6497	0.6589/0.5582	0/0	0/0	0/0	0.015/0.0465
3	0.568/0.4536	0/0	0.5264/0.4407	0/0	0/0	−0.489/−0.2665
4	0.369/0.2365	0/0	0/0	0.645/0.5285	0/0	0.0684/0.0994
5	1.568/1.4060	0/0	0/0	0.248/0.2605	0.168/0.152	0.058/0.0739
i	β_i	h_{i1}	h_{i2}	h_{i3}	h_{i4}	h_{i5}
1	1.2548/1.6479	0.654/0.5582	0/0	0/0	0/0	0.0541/0.0465
2	0.598/0.5793	0/0	0.705/0.5097	0/0	0/0	−0.189/−0.2218
3	0.198/0.2456	0/0	0/0	0.584/0.4506	0.00698/0.0441	0.061/0.092
4	1.854/2.0892	0/0	0/0	−0.0002/−0.0075	0.128/0.304	0.0648/0.0484
5	3.156/2.9437	0.256/0.1962	0.05/0.1791	0/0	0/0	0.354/0.2354

TABLE VII

INFERRED REGULATIONS FOR THE *E. coli* SOS NETWORK

	Suppressed By	Activated By
uvrD	lexA	uvrA
lexA	lexA	recA
umuD	uvrD, lexA	
recA	lexA	polB
uvrA	lexA	
polB	lexA	

For validation of this work, we will compare our results with works like those in [29] and [38] using a receiver operating characteristic (ROC) plot. Most of the previous researchers have done inference on six of the eight original genes. These are uvrD, lexA, umuD, recA, uvrA, and polB. Since the two excluded genes (uvrY and ruvA) have little involvement in the regulatory process, we decided to use only the six genes, making it easier to compare our work with the results of the other groups. We will use the microarray data, provided by the group, to generate the S-System model of the network. Those parameters will allow us to know the regulations taking place in these networks.

1) *Experimental Setup:* We used a population of 20 particles and set the search space based on previous results by [38]. Thus, we used a search region of $[-2, 2]$ for the kinetic orders g_{ij}, h_{ij} and $[0, 5]$ for the rate constants α and β. The remaining parameters were the same to those used in the previous experiments.

2) *Results:* After performing 10 000 iterations, the PSO inferred the inhibitory effect of other genes by lexA, which is in line with the previous works using *E. coli* [11], [38]. In this paper, however, we found the known regulative factor of recA over lexA that some approaches had difficult finding. Yet, the algorithm found three additional regulations that might be FPs.

Since we found many differences in the numerical values of the parameters for different iterations, presenting the results like Tables II, III, and V is difficult. Instead, we present, in Table VII, the interactions found by the algorithm. We found that most of the regulatory patterns were correct at least 70% of the times we ran the algorithm. The remaining 30% was less precise, but it still found 80% of the correct regulations.

Since most of the other works present their results as false and positive regulations, when comparing our approach with

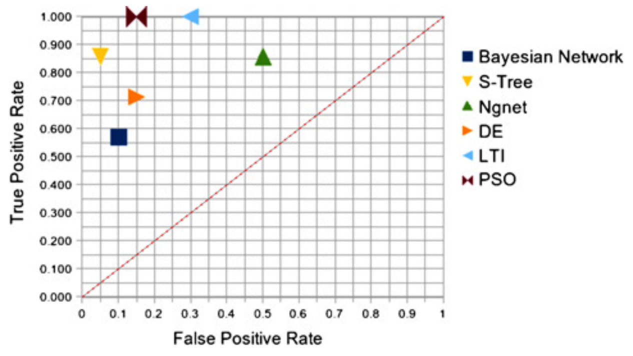


Fig. 5. ROC plot of the results in the SOS network.

other works, it is difficult to calculate a mean error. Thus, we calculated the number of TPs, FPs, TNs, and FNs, given the connections they reported and the known regulations for the SOS net. We took seven true regulations (the regulation of lexA over all the others and the regulation of recA over lexA) that are the consensus of true regulations. If their works reported self-regulations of the genes or regulation of recA over all the variables, we did not count them as FPs or TPs. Since recA regulates lexA, an inference algorithm may find these regulations. In fact, we found that most of the works reported a regulation of recA over the other genes.

In Table VIII, we compare the sensitivity and specificity of our algorithm with other approaches. PSO has a perfect sensitivity, finding all the true regulations, and its performance in specificity is among the top results in the table.

We also calculated the F1-score (also known as F-score) to score the algorithms without looking at the tradeoff sensitivity–specificity. The PSO presents a high F-score, which means it is good for predicting the true regulations of the network. It ranks second, after the S-tree, which is the best algorithm to calculate the true regulations in the SOS net.

To visualize these results, we show an ROC plot (Fig. 5), where the x-axis is the FP rate (1-specificity) and the y-axis is the TP rate (sensitivity). Since we do not have more information from the comparative works, it is difficult to create comparable ROC curves for all of them. In an ROC plot, the best points are far up-left from the identity function ($x = y$). In the ROC plot, we see that, by having a high TP rate and a low FP rate, PSO is clearly among the best algorithms.

TABLE VIII
SPECIFICITY, SENSITIVITY, AND F-SCORE FOR DIFFERENT MODELS OF THE *E. coli* SOS NETWORK

	#Regulations	#TP	#FP	#TN	#FN	Sensitivity	Specificity	F-Score
Bayesian network [39]	6	4	2	18	3	0.571	0.900	0.615
S-tree [38]	7	6	1	19	1	0.857	0.950	0.857
LTV [11]	13	7	6	14	0	1.000	0.700	0.7
NGnet [40]	16	6	10	10	1	0.857	0.500	0.521
DE [29]	8	5	3	17	2	0.714	0.850	0.667
L1–DPSO	10	7	3	17	0	1.000	0.850	0.823

TABLE IX
SPECIFICITY, SENSITIVITY, AND F-SCORE FOR DIFFERENT MODELS OF THE *E. coli* SOS NETWORK

	Total Runtime (h)
Bayesian network [39]	0.01
S-tree [38]	35
LTV [11]	0.1
NGnet [40]	0.04
DE [29]	0.3
L1–DPSO	4

3) *Runtime Comparison:* To compare the effectiveness of our approach, we compare the runtimes of different methods to find the regulations in the SOS network, as reported in Table VIII.

Table IX shows that our method takes longer to find good results, although the S-tree, the method with the best F-score, takes even longer. Methods like Bayesian networks and NGnet are very fast; however, their F-score is low compared with other methods.

H. Preliminary Test With a Medium-Sized Network

While the current tests present good results for small networks, we also test the performance of larger networks. For our preliminary test, we used the artificial network NET2 proposed by Noman and Iba [30]. This network's nodes and connections are sampled randomly, limiting the maximum number of connections. Since it has known regulation parameters, it is often convenient to use an artificial model for these kinds of tests. We provide the artificial network that is available at http://www.iba.t.u-tokyo.ac.jp/~leon/Net2/NET2.csv

Table X shows the dynamic parameters of the network.

For this model's inference, we used a population of 40 particles, a search region of $[-1, 1]$ for the kinetic orders g_{ij}, h_{ij} and [0, 10] for the rate constants α and β. The restart parameters and the values of the η and λ parameters were the same as for the NET1 inference, 0.05 and 5, respectively.

We increased the number of particles since the size of the feature vector also increased. We did experiments for 0% noise rate and 10% noise and calculated TP, FP, and FN, as done previously.

1) *Results:* Table XI shows that, compared with classic DE [30], the DPSO–L1 approach has an inferior performance on noiseless medium-sized networks. When the system has noise, however, the system does restrict the FPs it finds, which improves its performance compared with DE.

The restriction imposed by the regularizer is the reason for this behavior. The system does not find as many TPs as the DE implementation, but the TPs it finds are more reliable than the ones found by DE.

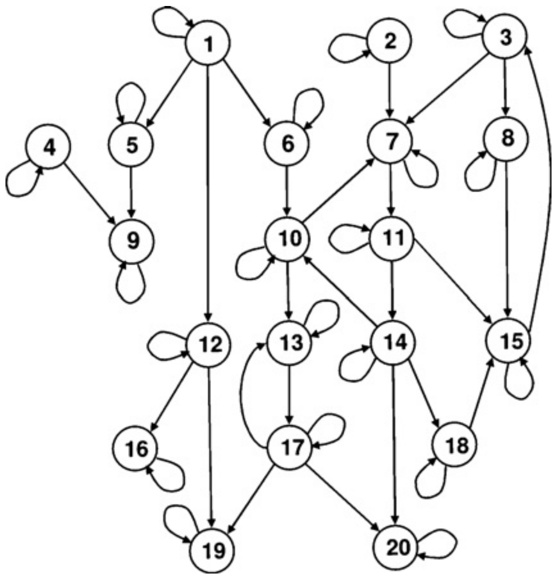


Fig. 6. Artificial network NET2 with 20 genes.

Regarding the processing time, our system took 32.3 h on average to reach the convergence state. This number is subjective to the number of possible interactions of the network. In contrast, using DE, the authors took 15.5 h to find the regulations.

IV. DISCUSSION

Recently, many researchers have used different techniques to solve the problem of gene inference in bioinformatics. There is, however, a need to increase the accuracy and decrease the FPs in a correct interaction for small networks.

Our hypothesis is that PSO, a good evolutionary optimizer, along with the use of a regularizer, should increase or match the best inference techniques' performance. A regularizer will keep the small values from increasing, thus avoiding the FPs' creation, while promoting a sparse set of parameters.

For the first set of experiments using the NET1 artificial network without noise, the use of fewer time series than similar results by Noman and Iba [31] and Kabir *et al.* [11] prove that PSO with a regularizer is enough to get good results by decreasing the time series.

In the noisy series, Table V shows how this approach tops models using LTV [11] and memetic algorithms [33]. The model also well presented a high robustness against noise, which the two previous approaches did not report for 25% data. The approach, however, did not manage to reduce the computation time drastically enough to make the S-System feasible for larger networks.

TABLE X
TARGET S-SYSTEM PARAMETERS FOR NETWORK MODEL NET2

α_i, β_i	10.0
$g_{i,j}$	$g_{3,15} = -0.7$, $g_{5,1} = 1.0$, $g_{6,1} = 2.0$, $g_{7,2} = 1.2$, $g_{7,3} = -0.8$, $g_{7,10} = 1.6$, $g_{8,3} = -0.6$, $g_{9,4} = 0.5$, $g_{9,5} = 0.7$, $g_{10,6} = -0.3$, $g_{10,14} = 0.9$, $g_{11,7} = 0.5$, $g_{12,1} = 1.0$, $g_{13,10} = -0.4$, $g_{13,17} = 1.3$, $g_{14,11} = -0.4$, $g_{15,8} = 0.5$, $g_{15,11} = -1.0$, $g_{15,18} = -0.9$, $g_{16,12} = 2.0$, $g_{17,13} = -0.5$, $g_{18,14} = 1.2$, $g_{19,12} = 1.4$, $g_{19,17} = 0.6$ $g_{20,14} = 1.0$, $g_{20,17} = 1.5$, other $g_{i,j} = 0.0$
$h_{i,j}$	1.0 if $(i = j)$, 0.0 otherwise

TABLE XI
COMPARISON BETWEEN DE AND DPSO–L1 FOR THE MEDIUM-SIZED NET2

	0.0%					10.0%				
	TP	FP	FN	TN	Std. Dev.	TP	FP	FN	TN	Std. Dev.
DE [30]	46	0	0	354	–	44.8	53.8	1.2	300.2	0.421
DPSO–L1	43	0	3	354	0.56	32.8	9.7	0.3	354	2.32

Next, we analyzed a real synthetic network, the yeast fermentation pathway; this network has been extensively studied by Cho *et al.* [38], Tsai and Wang [8], and Voit [35]. It is a good network to test a reconstruction algorithm's feasibility. The results by these researchers, however, are not comparable with one another, since they followed different inference techniques and used different metrics to report their results. It is possible, though, to survey the results' validity by looking at the statistical differences in Table VI. Since the network is a larger NET1, this test just corroborates that our approach is good when we increase the size of the net.

The study of the SOS network provided much more insight into the capabilities and obstacles for this particular approach. The SOS net is probably the most studied network for small net inference. Multiple works have presented good inference results of the SOS net. These works, however, have yet to do a perfect inference of all the parameters. Some of them, like the LTV approach by Kabir *et al.* [11], found all the true regulations, but also found many false regulations. Only the S-tree approach by Cho *et al.* [38] was capable of reducing to one of the number of false regulations, yet it still failed to find one of the true regulations.

Using our model, we found each of the true regulations, which is a good result compared with similar works. Yet, even though we used the regularizer, we found three false regulations. The reason for this might be that the amount of noise is too high for the algorithm to find only the true regulations. These three false regulations are still within the best three of the group presented in Table VIII.

When we compare the F-score, which reflects the tradeoff between precision and recall, we see how the PSO is slightly worse than the S-tree. Given that PSO is a highly random process, a different set of runs might present a better set of results.

Our model, however, does not top any of the previous methods in processing time, and the use of fewer time series is only an advantage compared with PSO itself; the method's processing time, though, was faster than the S-tree's, the best inference method so far.

When we tested the system in a larger network, we found that our implementation gives good preliminary results for the 20-gene network NET2. This approach, however, was less accurate for noiseless systems but had a better recognition rate in a noisy inference.

V. CONCLUSION

We presented a PSO approach to solve the inference problem in a gene network using the S-System, using a variation of the PSO with a chaotic operator to prevent the algorithm from reaching a local minimum. We also used an L1 regularizer to avoid overshooting and encouraging the parameters' sparsity in large networks, thus avoiding a large rate of FPs. We obtained satisfactory results that converge in a reasonable time using fewer candidates than other methods.

The algorithm proved to be robust to added noise up to 25%, which is better than previous works using DE and other models. We also found that, in the real-world test set, it detected all the regulations and had few mistakes compared with reported approaches.

We also found that increasing the number of inferred genes affected the required time for obtaining good results. Increasing it by a few genes proved to be almost intractable using the S-System with PSO, indicating a need for a different model if we wish to infer larger networks.

Further work may include using generative models in the experimental data to increase the available training data. If we wish to keep using the S-System, we can use parallelization of the PSO to increase the speed of the inference and to be able to use larger swarms to increase the performance of the algorithm. Also, we will try a different approach than PSO, like the recently popularized estimation of distribution algorithms.

REFERENCES

[1] J. Quackenbush, "Microarray data normalization and transformation," *Nat. Genet.*, vol. 32, pp. 496–501, Dec. 2002.

[2] I. Shmulevich, E. R. Dougherty, S. Kim, and W. Zhang, "Probabilistic Boolean networks: A rule-based uncertainty model for gene regulatory networks," *Bioinformatics*, vol. 18, no. 2, p. 261, 2002.

[3] M. I. Davidich and S. Bornholdt, "Boolean network model predicts cell cycle sequence of fission yeast," *PLoS One*, vol. 3, no. 2, p. e1672, 2008.

[4] T. Akutsu, S. Miyano, and S. Kuhara, "Identification of genetic networks from a small number of gene expression patterns under the Boolean network model," in *Proc. Pacific Symp. Biocomput.*, vol. 4. 1999, pp. 17–28.

[5] A. V. Werhli, M. Grzegorczyk, and D. Husmeier, "Comparative evaluation of reverse engineering gene regulatory networks with relevance networks, graphical Gaussian models and Bayesian networks," *Bioinformatics*, vol. 22, no. 20, p. 2523, 2006.

[6] N. Friedman, M. Linial, I. Nachman, and D. Pe'er, "Using Bayesian networks to analyze expression data," *J. Computat. Biol.*, vol. 7, nos. 3–4, pp. 601–620, 2000.

[7] M. A. Savageau, *Biochemical Systems Analysis: A Study of Function and Design in Molecular Biology*. Reading, MA: Addison-Wesley, 1976.

[8] K. Y. Tsai and F. S. Wang, "Evolutionary optimization with data collocation for reverse engineering of biological networks," *Bioinformatics*, vol. 21, no. 7, p. 1180, 2005.

[9] N. Noman and H. Iba, "Accelerating differential evolution using an adaptive local search," *IEEE Trans. Evol. Computat.*, vol. 12, no. 1, pp. 107–125, Feb. 2008.

[10] P. K. Liu and F. S. Wang, "Inference of biochemical network models in S-system using multiobjective optimization approach," *Bioinformatics*, vol. 24, no. 8, p. 1085, 2008.

[11] M. Kabir, N. Noman, and H. Iba, "Reverse engineering gene regulatory network from microarray data using linear time-variant model," *BMC Bioinform.*, vol. 11, Suppl. 1, p. S56, Jan. 2010.

[12] M. Al-Rashidi and M. El-Hawary, "A survey of particle swarm optimization applications in electric power systems," *IEEE Trans. Evol. Computat.*, vol. 13, no. 4, pp. 913–918, Apr. 2009.

[13] R. Eberhart, "Recent advances in particle swarm," in *Proc. Congr. Evol. Computat.*, 2004, pp. 90–97.

[14] R. Poli, "Analysis of the publications on the applications of particle swarm optimisation," *J. Artif. Evol. Applicat.*, vol. 2008, no. 2, pp. 1–10, 2008.

[15] R. Xu, I. I. D. Wunsch, and R. Frank, "Inference of genetic regulatory networks with recurrent neural network models using particle swarm optimization," *IEEE/ACM Trans. Computat. Biol. Bioinform.*, vol. 4, no. 4, pp. 681–692, Oct.–Dec. 2007.

[16] X. F. Xie, W. J. Zhang, and Z. L. Yang, "Dissipative particle swarm optimization," in *Proc. CEC*, vol. 2. 2002, pp. 1456–1461.

[17] J. Vesterstrom and R. Thomsen, "A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems," *Evol. Computat.*, vol. 2, pp. 1980–1987, Jun. 2004.

[18] M. A. Savageau, "Biochemical systems analysis: I. Some mathematical properties of the rate law for the component enzymatic reactions," *J. Theoretic. Biol.*, vol. 25, no. 3, pp. 365–369, 1969.

[19] M. A. Savageau, "Biochemical systems analysis: II. The steady-state solutions for an n-pool system using a power-law approximation," *J. Theoretic. Biol.*, vol. 25, no. 3, pp. 370–379, 1969.

[20] M. A. Savageau, "Biochemical systems analysis: III. Dynamic solutions using a power-law approximation," *J. Theoretic. Biol.*, vol. 26, no. 2, pp. 215–226, 1970.

[21] D. Tominaga, N. Koga, and M. Okamoto, "Efficient numerical optimization algorithm based on genetic algorithm for inverse problem," in *Proc. Genet. Evol. Computat. Conf.*, vol. 251. 2000, p. 258.

[22] N. Noman and H. Iba, "Inference of gene regulatory networks using S-system and differential evolution," in *Proc. Conf. Genet. Evol. Computat.*, 2005, p. 439.

[23] O. R. Gonzalez, C. Küper, K. Jung, P. C. Naval, and E. Mendoza, "Parameter estimation using simulated annealing for S-system models of biochemical networks," *Bioinformatics*, vol. 23, no. 4, pp. 480–486, Feb. 2007.

[24] A. Y. Ng, "Feature selection, L1 vs. L2 regularization, and rotational invariance," in *Proc. 21st ICML*, 2004, pp. 78–86.

[25] L. Palafox and H. Hashimoto, "4W1H and particle swarm optimization for human activity recognition," *J. Adv. Computat. Intell. Intell. Inform.*, vol. 15, no. 7, pp. 793–799, 2011.

[26] I. C. Trelea, "The particle swarm optimization algorithm: Convergence analysis and parameter selection," *Inform. Process. Lett.*, vol. 85, no. 6, pp. 317–325, 2003.

[27] S. Kimura, K. Ide, A. Kashihara, M. Kano, M. Hatakeyama, R. Masui, N. Nakagawa, S. Yokoyama, S. Kuramitsu, and A. Konagaya, "Inference of S-system models of genetic networks using a cooperative coevolutionary algorithm," *Bioinformatics*, vol. 21, no. 7, p. 1154, 2005.

[28] S. Kikuchi, D. Tominaga, M. Arita, K. Takahashi, and M. Tomita, "Dynamic modeling of genetic networks using genetic algorithm and S-system," *Bioinformatics*, vol. 19, no. 5, p. 643, 2003.

[29] N. Noman and H. Iba, "Reverse engineering genetic networks using evolutionary computation," *Genome Inform.*, vol. 16, no. 2, pp. 205–214, Jan. 2005.

[30] N. Noman and H. Iba, "Inferring gene regulatory networks using differential evolution with local search heuristics," *IEEE/ACM Trans. Computat. Biol. Bioinform.*, vol. 4, no. 4, pp. 634–647, Oct.–Dec. 2007.

[31] N. Noman and H. Iba, "On the reconstruction of gene regulatory networks from noisy expression profiles," in *Proc. IEEE Int. Conf. Evol. Computat.*, no. 1. Sep. 2006, pp. 2543–2550.

[32] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," in *Proc. CEC*, 1999, pp. 1945–1950.

[33] M. M. Hasan, N. Noman, and H. Iba, "A prior knowledge based approach to infer gene regulatory networks," in *Proc. ISB*, 2010, p. 1.

[34] M. Ronen, R. Rosenberg, B. I. Shraiman, and U. Alon, "Assigning numbers to the arrows: Parameterizing a gene regulation network by using accurate expression kinetics," *Proc. Nat. Acad. Sci.*, vol. 99, no. 16, p. 10555, 2002.

[35] E. O. Voit, *Computational Analysis of Biochemical Systems: A Practical Guide for Biochemists and Molecular Biologists*. Cambridge, U.K.: Cambridge Univ. Press, 2000.

[36] J. Vera, P. De Atauri, M. Cascante, and N. V. Torres, "Multicriteria optimization of biochemical systems by linear programming: Application to production of ethanol by Saccharomyces cerevisiae," *Biotechnol. Bioeng.*, vol. 83, no. 3, pp. 335–343, 2003.

[37] P. M. Schlosser, G. Riedyf, and J. E. Bailey, "Ethanol production in bakers yeast: Application of experimental perturbation techniques for model development and resultant changes in flux control analysis," *Biotechnol. Progress*, vol. 10, no. 2, pp. 141–154, Mar.–Apr. 1994.

[38] D.-Y. Cho, K.-H. Cho, and B.-T. Zhang, "Identification of biochemical networks by S-tree based genetic programming," *Bioinformatics*, vol. 22, no. 13, pp. 1631–1640, 2006.

[39] B.-E. Perrin, L. Ralaivola, A. Mazurie, S. Bottani, J. Mallet, and F. D'Alche-Buc, "Gene networks inference using dynamic Bayesian networks," *Bioinformatics*, vol. 19, Suppl. 2, pp. ii138–ii148, Oct. 2003.

[40] S. Kimura, K. Sonoda, S. Yamane, H. Maeda, K. Matsumura, and M. Hatakeyama, "Function approximation approach to the inference of reduced NGnet models of genetic networks," *BMC Bioinform.*, vol. 9, p. 23, Jan. 2008.

Leon Palafox (M'06) received the B.Sc. degree from the National Autonomous University of Mexico, Mexico City, Mexico, and the M.Sc. degree from the Graduate School of Engineering, University of Tokyo, Tokyo, Japan, both in electronic engineering, where he is currently pursuing the Ph.D. degree in electric engineering and information sciences.

His current research interests include evolutionary computation, machine learning, and bioinformatics.

Nasimul Noman received the B.Sc. and M.Sc. degrees in computer science from the University of Dhaka, Dhaka, Bangladesh, and the Ph.D. degree in frontier informatics from the Graduate School of Frontier Sciences, University of Tokyo, Tokyo, Japan.

Since March 2002, he has been a Faculty Member with the Department of Computer Science and Engineering, University of Dhaka. He is also a Joint Researcher with the Department of Information Science in Technology, Tokyo University. His current research interests include evolutionary computation, computational biology, and bioinformatics.

Hitoshi Iba (M'99) received the Ph.D. degree from the University of Tokyo, Tokyo, Japan, in 1990.

From 1990 to 1998, he was with the Electrotechnical Laboratory, Ibaraki, Japan. Since April 1998, he has been with the University of Tokyo, where he is currently a Professor with the Graduate School of Information Science and Technology. His current research interests include evolutionary computation, genetic programming, bioinformatics, foundation of artificial intelligence, machine learning, and robotics.

Dr. Iba is an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and the *Journal of Genetic Programming and Evolvable Machines*.