Supporting Information: Inference of Vohradský’s models of genetic networks by solving two-dimensional function optimization problems

Shuhei Kimura et al.
kimura@ike.tottori-u.ac.jp

1 REX\textsuperscript{star}/JGG

REX\textsuperscript{star}/JGG [1] is a real-coded genetic algorithm, a sort of evolutionary algorithm, that uses JGG as a generation alternation model and REX\textsuperscript{star} as a recombination operator. This section will describe each of the operators in greater detail.

1.1 JGG

JGG is a generation alternation model. The following is an algorithm of JGG. In the algorithm described below, a recombination operator uses \( m \) (\( \geq 2 \)) parents to generate offsprings.

[Algorithm: MGG]

1. Initialization
   As an initial population, create \( n_p \) individuals. As REX\textsuperscript{star}/JGG is a real-coded genetic algorithm, these individuals are represented as \( s \)-dimensional real number vectors, where \( s \) is the dimension of the search space. Set \( Generation = 0 \).

2. Selection for reproduction
   Select \( m \) individuals without replacement from the population. The selected individuals, that are expressed here as \( p_1, p_2, \ldots, p_m \), are used as the parents for the recombination operator in the next step.

3. Generation of offsprings
   Generate \( n_c \) children by applying the recombination operator to the parents selected in the previous step. This study uses REX\textsuperscript{star} as the recombination operator.

4. Selection for survival
   Select the best \( m \) individuals from the family containing the \( m \) parents (\( p_1, p_2, \ldots, p_m \)) and their children. Then, replace the \( m \) parents with the selected individuals. In the original JGG, the best \( m \) individuals are selected only from the children. As its optimization process seemed to be unstable, however, this study slightly modified its algorithm.

5. Termination
   Stop if the halting criteria are satisfied. Otherwise, \( Generation \leftarrow Generation + 1 \), and then return to the step 2.

1.2 REX\textsuperscript{star}

REX\textsuperscript{star} is a real-coded crossover operator. REX\textsuperscript{star} uses \( s + 1 \) parents, where \( s \) is the dimension of the search space, and generate \( n_c \) (\( > s + 1 \)) children according to the following algorithm. In the following algorithm, the parents are represented as \( p_1, p_2, \ldots, p_{s+1} \).

[Algorithm: REX\textsuperscript{star}]

1. Generate reflection points, \( p_1, p_2, \ldots, p_{s+1} \), of the parents, i.e.,
   \[
   p_i = 2G - p_i, \tag{1}
   \]
where
\[ G = \frac{1}{s+1} \sum_{i=1}^{s+1} p_i. \]

2. Compute the objective values of the \( s+1 \) reflection points generated in the previous step. In REX\(^{\text{star}}\), these reflection points are treated as the children.

3. From the parents and their reflection points, select the best \( s+1 \) individuals, and then compute the center of the gravity of the selected individuals. This study represents it as \( G^c \).

4. Generate \( n_c - s - 1 \) children by applying the following equation \( n_c - s - 1 \) times. Note that the \( s+1 \) reflection points generated in the step 1 are treated as the children. The total number of the children generated is therefore \( n_c \).

\[ c = G + \text{diag}(\xi_1, \xi_2, \ldots, \xi_s)(G^b - G) + \sum_{i=1}^{s+1} \xi^i(p_i - G), \]

where \( \xi^i \)'s and \( \xi^s \)'s are random numbers drawn from uniform distributions \([0, t]\) and \([-\sqrt{\frac{2}{s+1}}, \sqrt{\frac{3}{s+1}}]\), respectively, where \( t \) is a hyper-parameter named a step-size parameter.

## 2 Back-propagation through time

The discrete form of the Vohradský’s model can be viewed as a recurrent neural network. The existing inference methods [2, 5, 6] have thus designed on the basis of the learning algorithm for the recurrent neural network, i.e., the back-propagation through time [4]. In the back-propagation through time, all of the parameters of the Vohradský’s methods [2, 5, 6] have been designed on the basis of the learning algorithm for the recurrent neural network, i.e., the back-propagation through time [4]. In the back-propagation through time, all of the parameters of the Vohradský’s model are estimated simultaneously by minimizing

\[ S(\alpha, \beta, w, b) = \sum_{n=1}^{N} \sum_{k=2}^{K} \left( X_n|_{t_k} - X_n|_{\text{cal}} \right)^2, \]

where \( \alpha = (\alpha_1, \alpha_2, \ldots, \alpha_N) \), \( \beta = (\beta_1, \beta_2, \ldots, \beta_N) \), \( w = (w_{1,1}, w_{1,2}, \ldots, w_{N,N}) \), and \( b = (b_1, b_2, \ldots, b_N) \) are the model parameters, \( N \) is the number of genes contained in the network, and \( K \) is the number of measurements. \( X_n|_{t_k} \) and \( X_n|_{\text{cal}} \) are the observed and the computed expression level of the \( n \)-th gene at time \( t_k \), respectively. In the back-propagation through time, \( X_n|_{\text{cal}} \) is computed from the discrete form of the Vohradský’s model, i.e.,

\[ X_n|_{t_k} = \alpha_n f \left( \sum_{m=1}^{N} w_{n,m} X_m|_{t_{k-1}} + b_n \right) \Delta t + (1 - \beta_n \Delta t) X_m|_{t_{k-1}}, \]

where \( \Delta t = t_k - t_{k-1} \).

This study constructed two inference methods based on the back-propagation through time, i.e., BPTTLS and BPTTGA. BPTTLS and BPTTGA used a local search, i.e., the conjugate gradient method [3], and an evolutionary algorithm, i.e., REX\(^{\text{star}}/\text{JGG} \) [1], respectively, as function optimization algorithms. The following recommended values were used for the parameters of REX\(^{\text{star}}/\text{JGG} \) applied here: the population size \( n_p \) is 20s, the number of children generated per selection \( n_c \) is 3s, and the step-size parameter \( t \) is 2.5, where \( s \) is the dimension of the search space. Each run was continued until the best objective value did not improved over 2 \( \times \) \( n_p \) generations.

## References

[1] S. Kobayashi, “The frontiers of real-coded genetic algorithms,” Transactions of the Japanese Society for Artificial Intelligence, Vol. 24, pp. 147–162, 2009 (in Japanese).

[2] L. Palafox and H. Iba, “On the use of population based incremental learning to do reverse engineering on gene regulatory networks,” Proc. 2012 Congress on Evolutionary Computation, pp. 1865–1872, 2012.

[3] W.H. Press, S.A. Teukolsky, W.T. Vetterling and B.P. Flannery, Numerical Recipes in C, 2nd Edition, Cambridge University Press, 1995.
[4] R.J. Williams and J. Peng, “An Efficient Gradient-based Algorithm for On-line Training of Recurrent Network Trajectories,” *Neural Computation*, Vol. 2, pp. 490–501, 1990.

[5] R. Xu and G.K. Venayagamoorthy and D.C. Wunsch II, “Inference of gene regulatory networks with hybrid differential evolution and particle swarm optimization,” *Neural Networks*, Vol. 20, pp. 917–927, 2007.

[6] R. Xu, D.C. Wunsch II and R.L. Frank, “Inference of genetic regulatory networks with recurrent neural network models using particle swarm optimization,” *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, Vol. 4, pp. 681–692, 2007.