Neural Comput & Applic (2013) 22:63–70
DOI 10.1007/s00521-011-0750-z

Learning gene regulatory networks using the bees algorithm

Gonzalo A. Ruz · Eric Goles

Received: 16 February 2011 / Accepted: 20 September 2011 / Published online: 14 October 2011

Abstract Learning gene regulatory networks under the threshold Boolean network model is presented. To accomplish this, the swarm intelligence technique called the bees algorithm is formulated to learn networks with predefined attractors. The resulting technique is compared with simulated annealing through simulations. The ability of the networks to preserve the attractors when the updating schemes is changed from parallel to sequential is analyzed as well. Results show that Boolean networks are not very robust when the updating scheme is changed. Robust networks were found only for limit cycle length equal to two and specific network topologies. Throughout the simulations, the bees algorithm outperformed simulated annealing, showing the effectiveness of this swarm intelligence technique for this particular application.

Keywords Swarm intelligence · The bees algorithm · Simulated annealing · Boolean networks · Attractors

1 Introduction

In recent years, learning of gene regulatory network (GRN) has caught the attention of the machine learning community. The problem can be described in a simple way: given a set of gene expressions (data), find the most plausible network that can contain those expressions. There are several aspects that need to be defined when solving this problem. An important one is, what model will be used to represent the GRN. For this, researches have proposed several mathematical models: Boolean networks [10], Probabilistic Boolean networks [26], Petri nets [28], Bayesian networks [32], recurrent neural networks [14], differential equations [8], linear regression [33], etc. The robustness of the model should be considered as well to see how it behaves in the presence of noise or external perturbations.

For this work, a very simple deterministic model introduced by Stuart Kauffman [10] called Boolean networks will be considered to represent GRN. In this network, each node represents a gene that can take the values of 1 or 0 (active or inactive respectively). The edges represent the direct influences between the genes. Also, each node has associated a Boolean function that is used to compute the dynamics of the network. Although this model was introduced more than 40 years ago, new theoretical and algorithmic results are constantly been reported [3, 5–7, 24]. An interesting characteristic of the Boolean network are the attractors that are the states of the network where each of the possible 2^n states, where n is the number of nodes in the network, converge after some updates of the network. Kauffman associated the attractors to different cell types, and in other works [9], they have been considered as cancer cells.

The construction (learning) of these models from data is typically referred to as the reverse engineering problem [1, 15]. It is a difficult task, since the structure of the network as well as the Boolean function for each node must be learned simultaneously. Also, the number of different structures that can be found grow exponentially with the number of genes as well as the number of states. So, given the large search space, a meta-heuristic approach is of interest. In the context of reverse engineering of GRN,

G. A. Ruz (✉) · E. Goles
Facultad de Ingeniería y Ciencias, Universidad Adolfo Ibáñez,
Av. Diagonal las Torres, 2640 Peñalolén, Santiago, Chile
e-mail: gonzalo.ruz@uai.cl

E. Goles
e-mail: eric.chacc@uai.cl

evolutionary computation is the meta-heuristic technique that has had the most attention. Several developments have been reported, for example, genetic algorithms (GA) with GRN modeled by an S-system [25], genetic programming (GP) with GRN modeled by a combination of kinetic equations, and recurrent neural networks [14], GA, with GRN modeled by a multistate discrete network allowing each node (gene) to have more than two states [21]. A comparison of evolutionary algorithms (GA, GP, evolution strategy, evolutionary programming, and differential evolution) with GRN modeled by the S-system and neural networks was carried out in [27].

Other meta-heuristic methods such as simulated annealing (SA) have been used as well. In [16], SA is used to learn GRN modeled by Bayesian networks, and GRN modeled by combinations of kinetic parameters that produce a desired behavior is presented in [29]. Threshold Boolean networks constructed by SA is reported in [23]; results from this work showed that there is a power law between the frequency of the networks that the SA found and the number of the updating sequences which the network could preserve the cycle attractor. Swarm intelligence has also been used for inference of GRN from data. Particle swarm optimization (PSO) is used for the reconstruction of gene networks modeled by recurrent neural networks (RNN) in [30, 34]. Also, a hybrid of differential evolution and PSO (DEPSO) for training RNN is investigated in [31]. Less work has been reported on the use of ants and bees. In [11], an ant system is implemented to generate candidate network structures, and in [22], the bees algorithm is used to generate Boolean network examples to support a theorem presented in that work.

The bees algorithm (BA) was first introduced in [19] as a swarm intelligence optimization technique that is inspired in the food search strategies used by the honeybees. Since then, it has been widely used for many engineering and manufacturing applications such as: motion planning for a robot arm [2], power dispatch [13], mechanical design optimization [20], workload balancing in the examination job assignment problem [4], etc.

In this paper, learning of GRN modeled by threshold Boolean network using the bees algorithm is presented. In particular, the bees algorithm is used to construct synthesizing networks with predefined properties, such as attractors and limit cycle lengths. The proposed approach will be used to conduct experiments to explore the behavior of the networks when the updating scheme is changed from parallel to sequential. Also, data of a known network, reported in [17], will be used to construct networks with predefined fixed points. The paper is organized as follows. Section 2 gives a description of the Boolean network as well as the bees algorithm. Section 3 presents the formulation of the bees algorithm to construct threshold Boolean networks with predefined properties. The experiment set up is described in Sect. 4, and the results are shown in Sect. 5 The discussion of the results appear in Sect. 6 and the main conclusions of the work are presented in Sect. 7.

2 Background

2.1 Boolean networks

Let $\boldsymbol{x}$ be a finite set of n variables, $\boldsymbol{x} = \{x_1, \ldots, x_n\}$, with $x_i \in \{0,1\}$ for $i = 1, \ldots, n$. A Boolean network is a pair (G, F), where $G = (\mathbf{V}, \mathbf{E})$ is a finite directed graph; $\mathbf{V}$ being the set of n nodes and $\mathbf{E}$ the set of edges. F is a Boolean function, $F: \{0,1\}^n \rightarrow \{0,1\}$ composed of n local functions $f_i: \{0,1\}^n \rightarrow \{0,1\}^n$. Furthermore, each local function f_i depends only on variables belonging to the neighborhood $V_i = \{j \in \mathbf{V} | (j,i) \in \mathbf{E}\}$. The indegree, K, of vertex i is $|V_i|$.

The updating schemes are repeated periodically, and since the hypercube is a finite set, the dynamics of the network converges to attractors that are fixed points or limit cycles, defined by

- Fixed point: $x_i^{t+1} = x_i^t$ for $\{i = 1, \ldots, n\}$.
- Limit cycle: $x_i^{t+p} = x_i^t$ for $\{ i = 1, \ldots, n\}$.

where $p > 1$ is a positive integer called the limit cycle length.

For this work, the updates of each node will be computed by

$$x_i(t+1) = f_i(\mathbf{x}) = u\left(\sum_{j=1}^{n} \omega_{ji} x_j(t) - \theta_i\right) \tag{1}$$

$$u(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{if } x \leq 0 \end{cases} \tag{2}$$

with $\omega_{ji} \in \{-1, 0, 1\}$ the weight of the edge coming from node j into the node i, and $\theta_i = 0$ for all i. This model will be called threshold Boolean network.

There are many ways of updating the values of a Boolean network, for this paper, two are of interest:

1. Parallel or synchronous mode: where every node is updated at the same time.
2. Sequential updating mode: where in every time step, every node is updated in a defined sequence.

2.2 The bees algorithm

The bees algorithm (BA) uses a swarm intelligence approach for function optimization and combinatorial optimization problems. It was first introduced by [19] and recently studied and compared to other meta-heuristic

Table 1 The bees algorithm control parameters

ns	Number of scout bees
ne	Number of elite sites
nb	Number of best sites
nre	Recruited bees for elite sites
nrb	Recruited bees for best sites
ngh	Neighborhood size
maxi	Maximum number of iterations

algorithms in [18]. The algorithm is based on the honeybees' food foraging process, where initially a group of scout bees go out into the field to search for promising flower patches rich in nectar. Then, they return to the beehive to recruit other bees to collect the food; this is carried out through the waggle dance process, which gives the location (distance, direction) of the flower patch and the amount of food available there. In the BA, each bee represents a candidate solution, the flower patch represents a local search area, and the amount of food the bee collects from the flower patch is the fitness value. The control parameters, which need to be specified by the user, are shown in Table 1.

3 Methods

There are three parts that need to be defined when implementing the BA.

3.1 Coding of solutions

The solutions, networks (edges and weight values), are coded using an adjacency matrix Ω, as shown in Fig. 1. The initial Ω_j, $j = 1, \ldots, ns$ in the BA are generated randomly, subject to any restriction, such as, indegree value K.

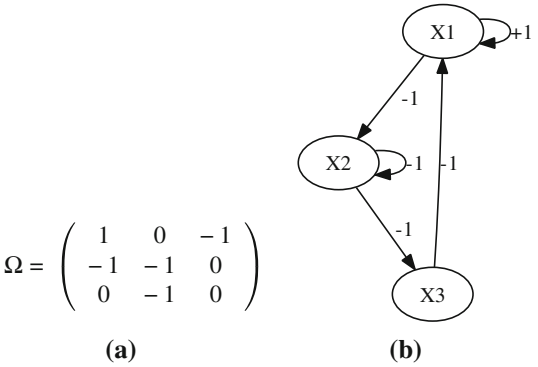

Fig. 1 **a** Adjacency matrix Ω of a network, and **b** resulting network

3.2 Definition of the fitness function

The fitness function for the Boolean network B is given by the deviation of the network output, defined by o_i for each node i, and the target value c_i (cycle) for each node i, which is computed by

$$\text{fitness}(B) = \frac{1}{pn}\sum_{t=1}^{p}\sum_{i=1}^{n}(o_i(t) - c_i(t))^2 \quad (3)$$

where n is the number of nodes in the network, and p is the cycle length.

3.3 Neighborhood search strategy

During the local search, each recruited bee for the elite or best sites must generate a candidate solution, Ω_{new}, based on the current solution, Ω_{old}, corresponding to the scout bee that found that site. For this, three simple neighborhood search strategy are proposed. The pseudocodes appears in Algorithms 1, 2, and 3. The *ngh* parameter of the BA specifies the number of nodes that will be modified by the search strategy.

Algorithm 1 Network local search A

Input: ngh, n, Ω_{old}
Output: Ω_{new}
$\Omega_{new} \leftarrow \Omega_{old}$;
for $i = 1, \ldots, ngh$ **do**
  $x1 \leftarrow 1 + \text{round}((n - 1) * \text{rand})$;
  $x2 \leftarrow 1 + \text{round}((n - 1) * \text{rand})$;
  $y1 \leftarrow 1 + \text{round}((n - 1) * \text{rand})$;
  $y2 \leftarrow 1 + \text{round}((n - 1) * \text{rand})$;
  If $rand > 0.5$ **then**
  $\Omega_{new}(x1, y1) \leftarrow 1$;
  else
  $\Omega_{new}(x1, y1) \leftarrow -1$;
  end
  $\Omega_{new}(x2, y2) \leftarrow 0$;
end

Using the above definitions, the general bees algorithm is shown in Algorithm 4.

3.4 Simulated annealing

During the simulations, the BA will be compared with simulated annealing. The simulated annealing (SA) algorithm [12] is based on the annealing treatment of solids consisting in the physical process of heating up a solid until it melts and cooling it down slowly until it crystallizes,

Algorithm 2 Network local search B

Input: ngh, n, Ω_{old}
Output: Ω_{new}
$\Omega_{new} \leftarrow \Omega_{old}$;
for $i = 1, \ldots, ngh$ **do**
  $x \leftarrow 1 + \text{round}((n-1) * \text{rand})$;
  $y1 \leftarrow 1 + \text{round}((n-1) * \text{rand})$;
  $y2 \leftarrow 1 + \text{round}((n-1) * \text{rand})$;
  If $rand > 0.5$ **then**
    $\Omega_{new}(x, y1) \leftarrow \Omega_{old}(x, y_2)$;
    $\Omega_{new}(x, y2) \leftarrow \Omega_{old}(x, y1)$;
  else
    $\Omega_{new}(x, y1) \leftarrow -\Omega_{old}(x, y2)$;
    $\Omega_{new}(x, y2) \leftarrow \Omega_{old}(x, y1)$;
  end
end

Algorithm 3 Network local search C

Input: ngh, n, Ω_{old}
Output: Ω_{new}
$\Omega_{new} \leftarrow \Omega_{old}$;
for $i = 1, \ldots, ngh$ **do**
  $x \leftarrow 1 + \text{round}((n-1) * \text{rand})$;
  $y \leftarrow 1 + \text{round}((n-1) * \text{rand})$;
  for $j = 1, \ldots, n$ **do**
  $\Omega_{new}(x, j) \leftarrow 0$
  end
  if $rand > 0.5$ **then**
    $\Omega_{new}(x, y) \leftarrow 1$
  else
    $\Omega_{new}(x, y) \leftarrow -1$
  end
end

changing the properties of the solid. The SA algorithm is used to solve optimization problems that can be formulated as the annealing process, where the states of the solid represents the feasible solutions of the optimization problem, the energy of the states correspond to the objective function value of the solutions, and the minimum energy corresponds to the optimal solution to the problem. To formulate the present problem into the annealing process, four parts are required to be defined.

1. Coding of solutions: the same as for BA
2. Definition of the fitness function: the same as BA
3. Neighborhood search strategy: the same as BA, but additionally, at each iteration, m, of the SA algorithm, the ngh decreases, following

Algorithm 4 The bees algorithm for learning threshold Boolean networks

Input: ns, ne, nb, nre, nrb, ngh, $maxi$, $cycle$
Output: Threshold Boolean network Ω
Initialize a random population Ω_i, for $i = 1, \ldots, ns$
Evaluate each candidate solution Ω_i in the fitness function;
$iter \leftarrow 0$;
While $iter < maxi$ **do**
  Search for nre candidates in the neighborhood of the first ne fittest solutions;
  Search for nrb candidates in the neighborhood of the $[ne + 1, ne + nb]$ fittest solutions;
  Search for $ns - (nre + nrb)$ candidates randomly;
  Evaluate each new candidate solution in the fitness function;
  if *the fittest solution* $= 0$ **then**
    Break;
  end
  $iter \leftarrow iter + 1$;
end

$$ngh = \text{round}\left(ngh \exp\left(\frac{-\Delta E}{mT}\right)\right) \tag{4}$$

where ΔE is the difference between the fitness value of the current and the newly candidate solution. If $ngh < 1$, then ngh is set to 1.

4. Cooling schedule: For this work, the standard geometric cooling rule is used. This rule updates the temperature T by

$$T(t+1) = \alpha T(t) \tag{5}$$

where $\alpha < 1$ is the cooling rate which is constant. In this work, the temperature is not decreased after each iteration, instead, (5) is applied every 10 iterations.

4 Experiment setup

To evaluate the performance of learning GRN under the threshold Boolean network model using the bees algorithm, two different simulations are carried out.

4.1 Simulation 1

In this simulation, the BA is used to learn threshold Boolean networks with predefined fixed point attractors. The data in this simulation are the six attractors of the model of floral morphogenesis in *Arabidopsis thaliana* proposed in [17]. The model consists in twelve interacting chemical species, designated by EMF1, TFL1, LFY, AP1, CAL, LUG, UFO, BFU, AG, AP3, PI, and SUP. The BFU species is a dimer of the AP3 and PI proteins, all the rest are proteins as well. The six attractors are (000000001000,

000000011110, 000100000000, 000100010110, 110000000000, 110000010110). 1,000 runs of the BA and the SA algorithm were carried out, and the resulting network of each run was recorded to verify whether it was capable of learning the six fixed points. The parameters for the BA and the SA were found empirically after several runs based on the effectiveness of learning the network and the execution time. For the BA: $ns = 20$, $ne = 5$, $nb = 10$, $nre = 20$, $nrb = 10$, $ngh = 2$, $maxi = 100$. For the SA algorithm: $m = 1{,}000$, $ngh = 11$, $\alpha = 0.7$, $T(0) = 100$, where $T(0)$ is the initial temperature. For both algorithms, network local search A was used. No restriction on the indegree K is given. The fitness value was computed as the sum of the evaluation of each fixed point in (3).

4.2 Simulation 2

This simulation is oriented to study the robustness of the networks. We are interested in exploring what happens to the limit cycle attractors of a network found in a parallel updating mode, when the updating scheme is changed to a sequential one. Is the limit cycle always preserved? Is it always destroyed? Does the network topology (indegree K) influence the outcome? These are some of the questions we are looking to answer through this simulation.

In this case, networks of $n = 6$ are learned with predefined attractor cycles of length $p = 2$, 3, 4, and 5. Also $K = 1$, 2, 3, 4, 5, and 6 are considered. The output o of the candidate solution, to be used in (3), is computed using a parallel updating scheme. For each pair of p and K, 100 different limit cycles, one per network, are considered to be learned. The limit cycles are generated randomly, but each variable in the limit cycle is non-constant. Then, the networks, which were capable of learning the limit cycle, are tested using a sequential updating mode where all the possible sequences are considered, that is, $6! = 720$, and the networks, which preserved the limit cycle (at least in one of all the possible permutations), are recorded.

The parameters of the BA are the same as in simulation 1. The parameters for the SA algorithm are: $m = 500$, $ngh = 5$, $\alpha = 0.7$, and $T(0) = 100$.

For both algorithms, when indegree $K = 1$, network local search C was used, while, for $K > 1$ network local search B was used.

5 Results

5.1 Results for simulation 1

The six attractors that needed to be learned by the threshold Boolean networks in simulation 1 are composed of 72 bits (6 fixed points × 12 nodes). Out of the 1,000 runs of the BA and the SA algorithm, the results are shown in Fig. 2. The x-axis represent the number of bits the networks failed to predict out of the 72, and the y-axis represents the frequency of the networks found. Notice how the BA algorithm was capable of learning the 6 attractors for the 1,000 runs. The BA obtained networks that had indegree average of 4.5, whereas the SA algorithm had indegree average of 6.2.

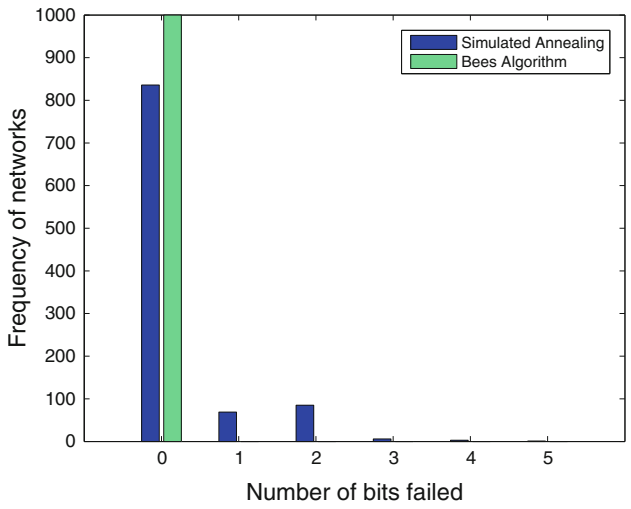

Fig. 2 Frequency of networks found by the SA algorithm and the BA for the six fixed points of the *Arabidopsis thaliana* model

5.2 Results for simulation 2

For the resulting figures in this simulation, the x-axis represent the indegree K of the network and the y-axis represents the frequency of the networks found.

5.2.1 Limit cycle length $p = 2$

Figure 3 shows the results for this case. P/S represents the networks that were capable of preserving the limit cycle when the updating scheme is changed from parallel to sequential. Figure 4 shows a resulting P/S network from the BA for $K = 3$. This network has the limit cycle (101001,010110) in parallel updating mode, and it can preserve it when the updating mode is changed to sequential, for the following updating sequence: 6-5-4-2-3-1. Figure 5 shows another resulting P/S network for $K = 5$. In this case, the network has the limit cycle (000010,111101) in parallel updating mode and for the sequential mode, for the following sequence: 4-1-6-3-5-2.

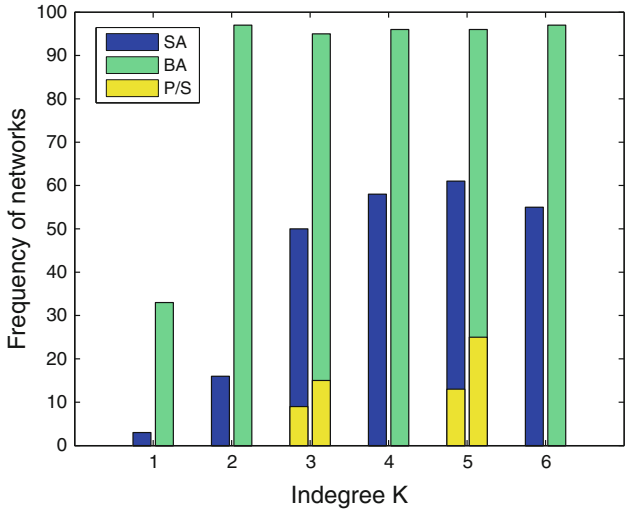

Fig. 3 Frequency of networks found with limit cycle length $p = 2$. *SA* simulated annealing, *BA* bees algorithm, *P/S* networks that preserve the cycle when the updating mode is changed from parallel to sequential

5.2.2 *Limit cycle length p = 3*

In this case, no networks were found that could preserve the limit cycle when the updating mode is changed from parallel to sequential (P/S), as shown in Fig. 6.

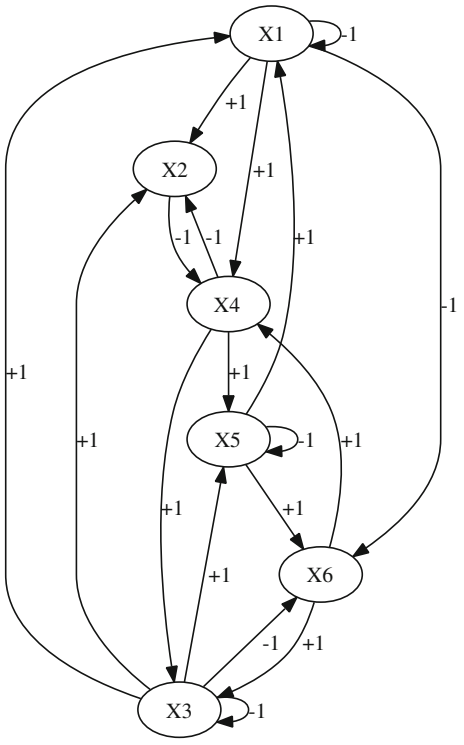

Fig. 4 Resulting network with $K = 3$ using the BA. This network has the same limit cycle (101001,010110) in parallel and in sequential mode for the updating sequence: 6-5-4-2-3-1

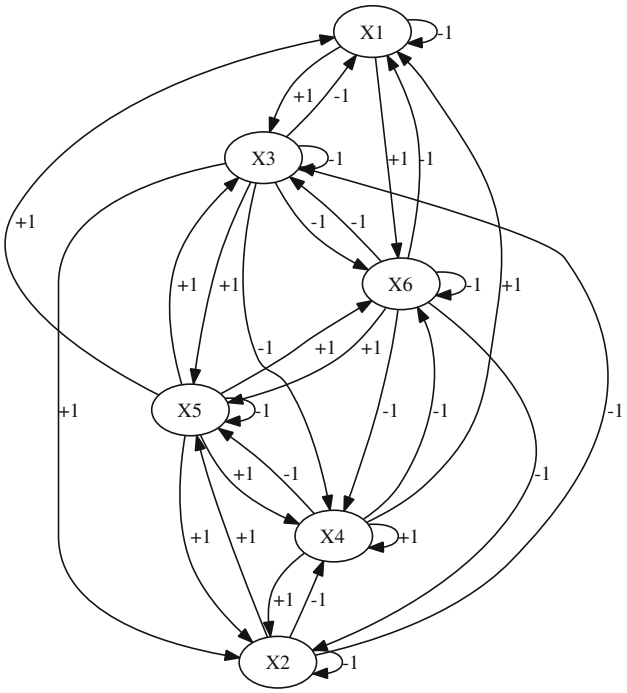

Fig. 5 Resulting network with $K = 5$ using the BA. This network has the same limit cycle (000010,111101) in parallel and in sequential mode for the updating sequence: 4-1-6-3-5-2

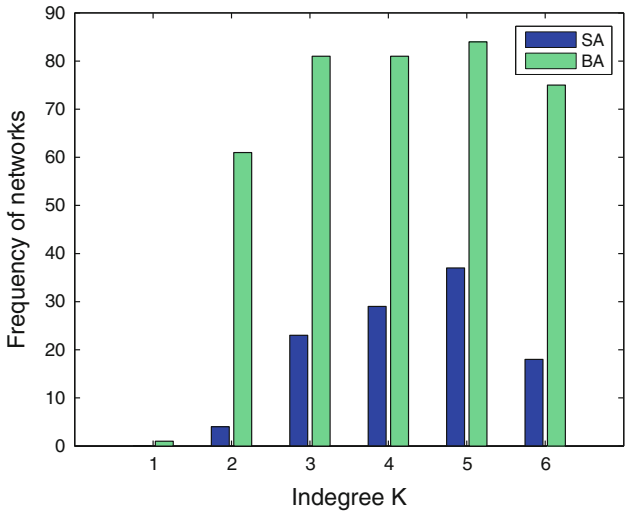

Fig. 6 Frequency of networks found with limit cycle length $p = 3$. *SA* simulated annealing, *BA* bees algorithm

5.2.3 *Limit cycle length p = 4*

Results for this case are shown in Fig. 7. No P/S network were found.

5.2.4 *Limit cycle length p = 5*

Similar to the previous simulations, no P/S networks are found as shown in Fig. 8.

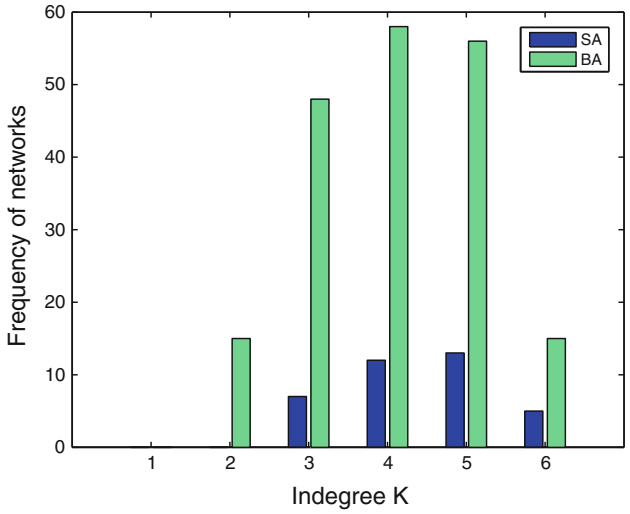

Fig. 7 Frequency of networks found with limit cycle length $p = 4$. *SA* simulated annealing, *BA* bees algorithm

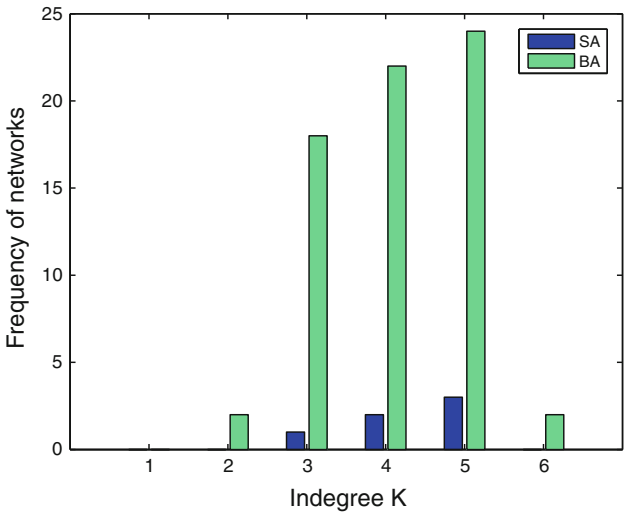

Fig. 8 Frequency of networks found with limit cycle length $p = 5$. *SA* simulated annealing, *BA* bees algorithm

6 Discussion

In general, the BA outperformed the SA algorithm in the task of learning threshold Boolean networks with predefined attractors (fixed points and limit cycles) as seen in the results of simulation 1 and 2. This is interesting because, although both algorithms share the same coding of solutions, local search strategies, fitness function, the BA obtained significantly better results, both in the actual number of solutions found as well as the solutions itself. Notice how in the results of simulation 1, the average indegree of the networks found by the BA were less than the SA algorithm, meaning that more compact solutions were found using BA than SA, requiring less edges to accomplish the same results. This shows that for this application, a swarm intelligence approach (many bees) for searching the solution in each iteration obtains better results than the SA approach that considers only one candidate solution in each iteration.

For simulation 1, since the attractors were fixed points, no exploration was carried out regarding the effect of changing the updating mode. It is straightforward to show that fixed points are preserved for any updating mode.

Analyzing the results of simulation 2, one can point out that the fact that no P/S networks were found for $K \leq 2$ is in accordance with the theorem proved in [22], which states that given a Boolean network $B = (G, F)$ of n variables with G strongly connected and $K \leq 2$, any limit cycle of B found in a parallel updating mode of length $p \geq 2$ where each variable in the limit cycle is non-constant cannot be preserved if the updating mode is switched to sequential mode. It is important to point out that during the simulations, only strongly connected networks were considered.

P/S networks were only found for $p = 2$ with $K = 3$ and $K = 5$. Also, P/S networks are <30% of the total networks found in $K = 3$ and $K = 5$ for $p = 2$. It is clear from the figures that as the limit cycle grew, the frequency of networks found reduced. Both SA and BA had difficulties to find networks when $K = 1$ for $p > 2$. In fact SA could not find any and BA only found one network for $p = 3$.

7 Conclusion

The formulation of the bees algorithm for learning GRN under the threshold Boolean network is presented. This is the first time, to our knowledge, that the algorithm is fully detailed for this application and used to carry out simulations of many Boolean networks. Comparisons with simulated annealing showed that the bees algorithm obtained more solutions, making this swarm intelligence technique promising for this application. In general, the threshold Boolean networks are not very robust, when the updating scheme is changed, it was shown that most of the time, the limit cycle is destroyed when the network changed its updating mode from parallel to sequential.

Future research will consider comparisons with other techniques, as well as studying robustness with less restrictions, for example, instead of preserving the exact limit cycle, we will consider vectors of attractors and explore what happens to each vector when changing the updating scheme.

Acknowledgments The authors would like to thank Conicyt-Chile under grant Fondecyt 3100044 (G.A.R.), Fondecyt 1100003 (E.G.), Basal (Conicyt)-CMM (E.G.), and ANILLO ACT-88 for financially supporting this research.

References

1. Akutsu T, Miyano S, Kuhara S (1999) Identification of genetic networks from a small number of gene expression patterns under the Boolean network model. In: Pac Symp Biocomput pp 17–28
2. Ang MC, Pham DT, Ng KW (2009) Minimum-time motion planning for a robot arm using the bees algorithm. In: 7th IEEE International Conference on Industrial Informatics, 2009. INDIN, pp 487–492
3. Aracena J, Goles E, Moreira A, Salinas L (2009) On the robustness of update schedules in Boolean networks. Biosystems 97:1–8
4. Baykasoglu A, Ozbakir L, Tapkan P (2009) The bees algorithm for workload balancing in examination job assignment. Eu J Ind Eng 3:424–435
5. Demongeot J, Goles E, Morvan M, Noual M, Sené S (2010) Attraction basins as gauges of robustness against boundary conditions in biological complex systems. PLoS ONE 5(8):e11,793
6. Goles E, Salinas L (2008) Comparison between parallel and serial dynamics of Boolean networks. Theor Comput Sci 396:247–253
7. Goles E, Salinas L (2010) Sequential operator for filtering cycles in Boolean networks. Adv Appl Math 45:346–358
8. Hoon MD, Imoto S, Miyano S (2003) Inferring gene regulatory networks from time-ordered gene expression data of bacillus subtilis using differential equations. In: Pac Symp Biocomput pp 17–28
9. Huang S (2006) Cell state dynamics and tumorigenesis in Boolean regulatory networks. In: Minai AA, Bar-Yam Y (eds) Unifying themes in complex systems. Springer, Berlin, pp 293–305
10. Kauffman SA (1969) Metabolic stability and epigenesis in randomly constructed genetic nets. J Theor Biol 22:437–467
11. Kentzoglanakis K, Poole M, Adams C (2008) Incorporating heuristics in a swarm intelligence framework for inferring gene regulatory networks from gene expression time series. In: Dorigo M, Birattari M, Blum C, Clerc M, Sttzle T, Winfield A (eds) Ant colony optimization and swarm intelligence, lecture notes in computer science, vol 5217. Springer Berlin, Heidelberg, pp 323–330
12. Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220:671–680
13. Lee JY, Darwish AH (2008) Multi-objective environmental/economic dispatch using the bees algorithm with weighted sum. In: Yoo SD (eds) EKC2008 Proceedings of the EU-Korea conference on science and technology, Springer proceedings in physics, vol 124. Springer Berlin, Heidelberg, pp 267–274
14. Lee W, Yang K (2008) Applying intelligent computing techniques to modeling biological networks from expression data. Genom Proteom Bioinform 6:111–120
15. Liang S, Fuhrman S, Somogyi R (1998) Reveal, a general reverse engineering algorithm for inference of genetic network architectures. In: Pac Symp Biocomput, pp 18–29
16. Liu G, Feng W, Wang H, Liu L, Zhou C (2009) Reconstruction of gene regulatory networks based on two-stage Bayesian network structure learning algorithm. J Bionic Eng 6:86–92
17. Mendoza L, Alvarez-Buylla ER (1998) Dynamics of the genetic regulatory network for arabidopsis thaliana flower morphogenesis. J Theor Biol 193:307–319
18. Pham DT, Castellani M (2009) The bees algorithm: modelling foraging behaviour to solve continuous optimization problems. Proc IMechE Part C: J Mech Eng Sci 223:2919–2938
19. Pham DT, Ghanbarzadeh A, Koc E, Otri S, Rahim S, Zaidi M (2006) The bees algorithm, a novel tool for complex optimisation problems. In: Proceedings of the Second International Virtual Conference on Intelligent production machines and systems (IPROMS 2006), pp 454–459
20. Pham DT, Ghanbarzadeh A, Otri S, Ko E (2009) Optimal design of mechanical components using the bees algorithm. Proc Inst Mech Eng Part C: J Mech Eng Sci 223:1051–1056
21. Repsilber D, Liljenstrom H, Andersson SGE (2002) Reverse engineering of regulatory networks: simulation studies on a genetic algorithm approach for ranking hypotheses. Biosystems 66:31–41
22. Ruz GA, Goles E (2010a) Cycle attractors for different deterministic updating schemes in Boolean regulation networks. In: Proc. of the IASTED International Conference on Computational Bioscience (Comp-Bio 2010), pp 620–625
23. Ruz GA, Goles E (2010b) Learning gene regulatory networks with predefined attractors for sequential updating schemes using simulated annealing. In: Proceeding of IEEE the Ninth International Conference on Machine Learning and Applications (ICMLA 2010), pp 889–894
24. Serra R, Villani M, Barbieri A, Kauffman SA, Colacci A (2010) On the dynamics of random Boolean networks subject to noise: attractors, ergodic sets and cell types. J Theor Biol 265:185–193
25. Shin A, Iba H (2003) Construction of genetic network using evolutionary algorithm and combined fitness function. Genome Inform 14:94–103
26. Shmulevich I, Dougherty ER (2009) Probabilistic Boolean networks: the modeling and control of gene regulatory networks. SIAM-Society for Industrial and Applied Mathematics, Philadelphia
27. Sirbu A, Ruskin H, Crane M (2010) Comparison of evolutionary algorithms in gene regulatory network model inference. BMC Bioinform 11:59
28. Steggles LJ, Banks R, Wipat A (2006) Modelling and analysing genetic networks: from Boolean networks to petri nets. Comput Methods Syst Biol 4210:127–141
29. Tomshine J, Kaznessis YN (2006) Optimization of a stochastically simulated gene network model via simulated annealing. Biophys J 91:3196–3205
30. Xu R, Venayagamoorthy G, Wunsch D (2006) A study of particle swarm optimization in gene regulatory networks inference. In: Wang J, Yi Z, Zurada J, Lu BL, Yin H (eds) Advances in neural networks—ISNN 2006, lecture notes in computer science, vol 3973. Springer, Berlin, pp 648–653
31. Xu R, Venayagamoorthy G, Wunsch D (2007) Modeling of gene regulatory networks with hybrid differential evolution and particle swarm optimization. Neural Netw 20:917–927
32. Yu J, Smith VA, Wang PP, Hartemink AJ, Jarvis ED (2004) Advances to Bayesian network inference for generating causal networks from observational biological data. Bioinformatics 20:3594–3603
33. Zhang S, Ching W, Tsing N, Leung H, Guo D (2010) A new multiple regression approach for the construction of genetic regulatory networks. Artif Intell Med 48:153–160
34. Zhang Y, Xuan J, de los Reyes B, Clarke R, Ressom H (2009) Reverse engineering module networks by pso-rnn hybrid modeling. BMC Genom 10:S15