Discrete Particle Swarm Optimization Algorithm for Solving Graph Coloring Problem

Kai Zhang[1,2], Wanying Zhu[1], Jun Liu[1,2], and Juanjuan He[1,2](✉)

[1] School of Computer Science, Wuhan University of Science and Technology, Wuhan 430081, People's Republic of China

[2] Hubei Province Key Laboratory of Intelligent Information Processing and Real-time Industrial System, Wuhan, China

hejuanjuan@wust.edu.cn

Abstract. Graph coloring problem is a well-known NP-complete problem in graph theory. Because GCP often finds its applications to various engineering fields, it is very important to find a feasible solution quickly. In this paper, we present a novel discrete particle swarm optimization algorithm to solve the GCP. In order to apply originally particle swarm optimization algorithm to discrete problem, we design and redefine the crucial position and velocity operators on discrete state space. Moreover, the performance of our algorithm is compared with other published method using 30 DIMACS benchmark graphs. The comparison result shows that our algorithm is more competitive with less chromatic numbers and less computational time.

Keywords: Graph coloring problem · Discrete particle swarm optimization algorithm · NP-complete problem

1 Introduction

The graph coloring problem (GCP) is an assignment of colors to each vertex such that any pair of adjacent points of edge have different colors. The challenge is to find the least number of colors, which is a NP-complete problem. In real life, the GCP has numerous practical applications including for instance, scheduling [1], register assignment in timetabling [2], register allocation in compilers [3], frequency assignment in mobile networks [4], and noise reduction in VLSI circuits [5]. So, it is necessary to design efficient algorithms that are reasonably fast and that can be used successfully in practice.

In recent years, a wide variety of algorithms have been proposed for Solving GCP, such as well-known RLF [6] and DSATUR [7] and heuristic algorithms including genetic algorithm [8,9] ant colony optimization algorithm [10,11], tabu search algorithm [12], neural network algorithm [13], etc. However, enormous amount of computation time still unsatisfactory for many applications, where a solution is required in a limited amount of time.

In this paper, a novel discrete particle swarm optimization algorithm for GCP is proposed. The particle swarm optimization (PSO) algorithm [14,15] is a stochastic global optimization method [16,17] based on swarm intelligence. The PSO has obvious advantages in optimization problem of continuous-valued space because of its fast convergence rate, simple computation and easy realization. It is originally developed for continuous problem, so it does not effectively solve the discrete combination optimization problem such as GCP. In order to apply originally particle swarm optimization algorithm to discrete problem, we design and redefine the crucial arithmetic position and velocity operators on discrete state space, including addition, subtraction and multiplication. Moreover, the discrete PSO algorithm has been implemented on Visual Studio 2013 and tested using 30 benchmark graphs provided by the DIMACS computational symposium web site http://mat.gsia.cmu.edu/color/instances.html. In compare to other published method, the comparison result shows that our algorithm is more competitive with less chromatic numbers and less computation time.

2 Graph Coloring Problem

Given a graph $G = (V, E)$ with n vertices and m edges, a coloring of G is an assignment of colors to each vertex such that any pair of adjacent points of edge have different colors. The graph coloring problem is to determine the smallest number of colors $\chi(G)$ and to find a coloring of G that uses $\chi(G)$ colors.

In this algorithm, vertex numbers range from 1 through n. The algorithm try to color n vertices with given k colors, numbered $\{0, 1, ..., k-1\}$. If v_i is a vertex, x_i is a variable that means vertex v_i has color x_i, $x_i \in \{0, 1, ..., k-1\}$. The position vector X of each particle P is defined as following vertex color sequence.

$$X = (x_1, x_2, \cdots, x_i, \cdots, x_n), \quad x_i \in \{0, 1, \ldots, k-1\} \tag{1}$$

A proper coloring formula as sequence $X(x_1, x_2, ..., x_n)$, which should guarantee that each vertex v_i must receive exactly one color x_i. And each sequence must be checked with all edges constraints conditions $x_i \neq x_j$, if $e(v_i, v_j) \in E$. A simple example is shown in Fig. 1.

3 Discrete PSO for Graph Coloring Problem

3.1 Particles Positions and State Space

In this section, we present a discrete PSO algorithm for graph coloring problem. The algorithm try to color n vertices with given k colors. The search space is defined as follows: the finite set of all vertex color sequences k^n. For example, an undirected graph G with 3 vertices and 2 colors, the state space is 2^3. All of binary color sequences with length 3 constitutes a solution space. Figure 2 illustrates its associated state-space.

If the smallest number of colors $\chi(G)$ increase to 3, the search state space also increase to 3^3 candidate particle positions. As shown in Fig. 3.

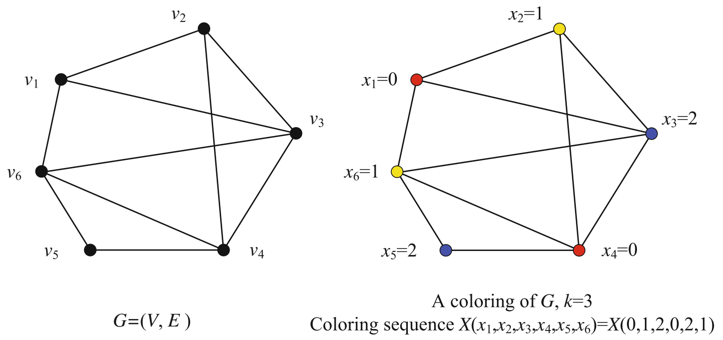

Fig. 1. The Graph G has 6 vertices and can be colored with 3 colors (Color figure online).

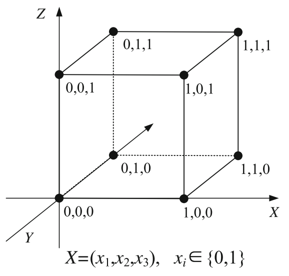

Fig. 2. Particles positions and state space for a graph G with 3 vertices and 2 colors.

3.2 Fitness Function

The algorithm define a fitness function $f(X)$ to evaluate candidate particle position X. Given a graph $G = (V,\ E)$ with n vertices and m edges, the fitness function $f(X)$ is the sum of conflicting vertices and conflicting edges. The mathematical description is as shown in Eq. 2.

$$
\begin{aligned}
&f(X) = a \times \sum_{i=1}^{n} ConflictV(x_i) + \sum_{i=1}^{m} ConflictE(x_i, x_j) \\
&ConflictV(x_i) = \begin{cases} 1, & \text{if } v_i \text{ have conflict edge} \\ 0, & \text{else} \end{cases} \\
&ConflictE(x_i, x_j) = \begin{cases} 1, & \text{if } e(v_i, v_j) \in E,\ x_i = x_j \\ 0, & \text{else} \end{cases}
\end{aligned}
\tag{2}
$$

where X is the position vector of particle P. $ConflictV(x_i)$ is the number of conflicting vertices. $ConflictE(x_i, x_j)$ is the number of conflicting edges. Sometimes the number of edges is far more than the number of vertices. To avoid the influence of the number of conflicting vertices being ignored, the number of conflicting vertices is multiplied by a positive coefficient a. It is obvious that $f(X) \geq 0$. The aim of the optimizing process is to minimize $f(X)$ until reaching $f(X) = 0$ for a fixed k colors, which corresponds to a valid k-coloring.

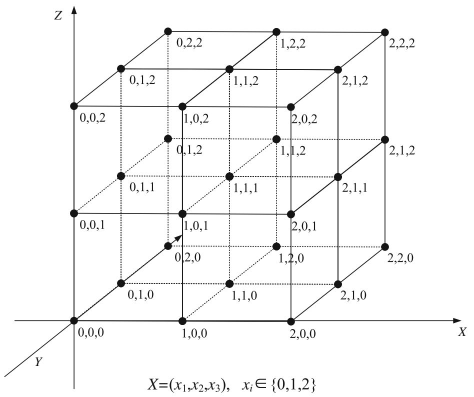

Fig. 3. Particles positions and state space for a graph G with 3 vertices and 3 colors.

3.3 Velocity Vectors

In our PSO algorithm, V denotes the velocity of the particle, and velocity is also defined a vector, as shown in Eq. 3.

$$V = (v_1, v_2, \cdots, v_i, \cdots, v_n), \quad v_i \in Z \tag{3}$$

In addition, the velocity V of a particle at timestep $t+1$ is updated as Eq. 4.

$$V_{t+1} = \omega \times V_t + c_1 \times r_1 \times (P_{pbest} - X_t) + c_2 \times r_2 \times (P_{gbest} - X_t) \tag{4}$$

where P_{pbest} is personal best position of the particle, and P_{gbest} is the best position of its neighborhood, respectively. ω is an inertia weight, c_1 and c_2 are two acceleration coefficients, and r_1 and r_2 are two random real numbers in [0,1].

However, in this equation the value of V_{t+1} is not an integer vector, but a real vector. In order to guarantee that the V_{t+1} is an integer vector, the algorithm round off V_{t+1} to an integer, as shown in Eq. 5.

$$V_{t+1} = \mathrm{INT}(\omega \times V_t) + \mathrm{INT}(c_1 \times r_1 \times (P_{pbest} - X_t)) + \mathrm{INT}(c_2 \times r_2 \times (P_{gbest} - X_t)). \tag{5}$$

3.4 Subtraction Between Position and Position

The subtraction of two positions is a vector of velocity. The $P_{pbest} - X_t$ is the difference between personal best position and current position of the particle. The $P_{gbest} - X_t$ is the difference between particle position X_t with the best position of its neighborhood, as shown in Fig. 4.

3.5 External Multiplication Between Real Number and Velocity

In Eq. 5, the velocity vector is multiplied by another real number, which is the product of acceleration coefficients and random number, as shown in Eq. 6.

$$\begin{array}{c} c_1 \times r_1 \times (P_{pbest} - X_t) \\ c_2 \times r_2 \times (P_{gbest} - X_t) \end{array} \tag{6}$$

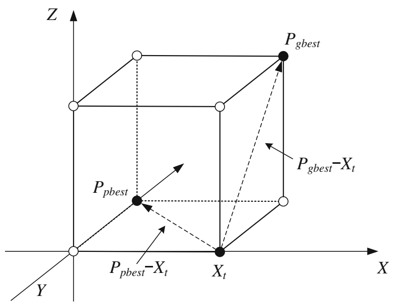

Fig. 4. Subtraction between position P_{best} and position X_i.

For the given coefficients $c(c_1$ or $c_2)$ and the random number $r(r_1$ or $r_2)$ can lead to the value change of velocity, meanwhile keep the direction same, as shown in Fig. 5.

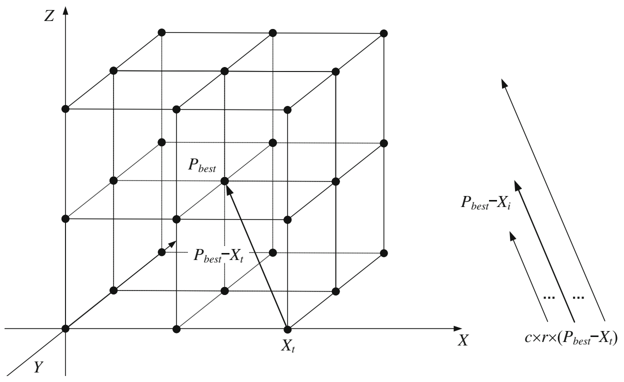

Fig. 5. External multiplication between real number and velocity.

3.6 Addition Between Velocity and Velocity

The velocities addition have meaningful geometric explanation, which is composed of three components, known as inertia current velocity, cognitive component personal best position and social component neighbor best position, respectively. The velocity addition operator is shown in Fig. 6.

3.7 Move by Position Plus Velocity

The sum of position X_t and velocity V_t of particle P_t is a new position X_{t+1}. However, sometimes the value of $X_t + V_t$ is bigger than the given color k. That means the new position would be out of the solution state, as shown in Fig. 7.

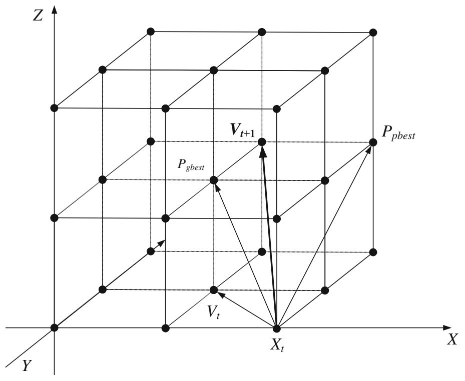

Fig. 6. Addition between velocity and velocity.

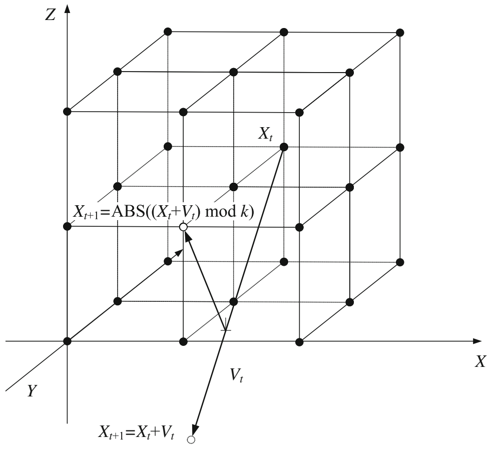

Fig. 7. Move by position X_t plus velocity V_t.

If the particle position X_t is out of given color k, we think that the particle collides with the boundary, then the particle returns to the interior of the solution state. So we take the modular operation of $X_t + V_t$ and get its absolute value. If the particle collides with the boundary, the components of the velocity vector reverse to the opposite direction, as shown in Eq. 7.

$$\begin{aligned} &X_{t+1} = \text{Abs}((X_t + V_t) \mod k) \\ &x_{(t+1)d} = \text{Abs}((x_{td} + v_{td}) \mod k) \\ &v_{(t+1)d} = (-1) \times v_{(t+1)d}, \quad \text{if } (x_{td} + v_{td}) > k \text{ or } (x_{td} + v_{td}) < 0. \end{aligned} \tag{7}$$

3.8 Algorithm Flowchart

The algorithm starts with an initialize q particles with random position and velocity vectors. For each position vector $X_t(x_1, x_2, ..., x_n)$, the component of

x_i is assigned from [0,$\chi(G)$-1] randomly. Then every particles position X_t is optimized and evaluated by fitness function $f(X_t)$. If the fitness value of position is better than the particle personal best fitness value, the particle best position should be replaced by current position. If all the particles are evaluated, the best of particles P_{pbest} positions is chose and replaced the P_{gbest} position. If the value of fitness $f(X_{gbest})$ decrease to 0, the position X_{gbest} is the optimization solution. Otherwise, the algorithm should repeat updating the particles velocities and positions until reach the maximum iterate. The flowchart is shown in Fig. 8.

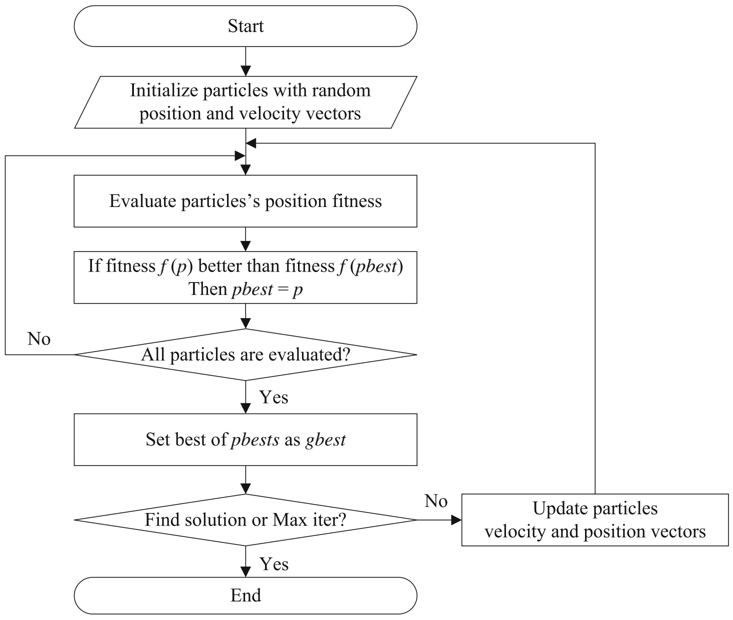
Start
Initialize particles with random position and velocity vectors
Evaluate particles's position fitness
If fitness f (p) better than fitness f (pbest) Then pbest = p
All particles are evaluated?
No
Yes
Set best of pbests as gbest
Find solution or Max iter?
No
Update particles velocity and position vectors
Yes
End

Fig. 8. The flowchart of discrete particle swarm optimization algorithm.

4 Results and Discussion

The algorithm has been implemented on C++ language of Visual Studio 2013. In this section, the compute results of our discrete particle swarm optimization algorithm are tested on 30 DIMACS benchmark graphs. The 'Best Known' column indicate the best known upper and lower bounds on the chromatic number. The HPGA is a hierarchical parallel genetic approach for GCP [8]. 'HPGA Time' column is the average running times in seconds.

In this algorithm, the initial particles number is 200. The inertia weight ω is set to be 0.5, acceleration coefficients c_1 and c_2 are set to be 0.6 and 0.7 respectively. The values of parameters are selected based on some preliminary trials. The comparison results are summarized in Table 1.

Table 1. Comparison results on DIMACS graph coloring challenge instances

Graph instances	Vertex numbers	Edge numbers	Best known	HPGA colors	HPGA time	DPSO colors	DPSO time
1-FullIns-5	282	3247	6			6	1.5571
2-FullIns-4	212	1621	6			6	0.7638
2-FullIns-5	852	12201	7			7	15.62
1-Insertions-4	67	232	5	5	2.4476	5	0.1038
1-Insertions-5	202	1227	4–5			6	0.3933
1-Insertions-6	607	6337	4–7			7	6.8907
2-Insertions-3	37	72	4	4	1.8964	4	0.0388
2-Insertions-4	149	541	4	5	6.4666	5	0.1387
2-Insertions-5	597	3936	3–6			6	11.171
3-Insertions-3	56	110	4	4	3.347	4	0.0265
anna	138	986	11	11	2.9016	11	1.7026
david	87	812	11	11	2.5522	11	0.101
games120	120	1276	9	9	5.0038	9	0.1055
huck	74	602	11	11	4.6278	11	0.0413
jean	80	508	10	10	4.398	10	0.0436
miles250	128	774	8	8	5.8828	8	0.1423
miles500	128	2340	20	20	6.0624	20	0.8848
miles1500	128	10396	73	73	10.1062	73	0.695
mug88-1	88	146	4	4	2.173	4	0.0492
mug88-25	88	146	4	4	1.8002	4	0.0488
mug100-1	100	166	4	4	2.016	4	0.06
mug100-25	100	166	4	4	2.5286	4	0.0607
myciel3	11	20	4	4	0.753	4	0.013
myciel4	23	71	5	5	0.865	5	0.075
myciel5	47	236	6	6	1.7608	6	0.0246
myciel6	95	755	7	7	1.479	7	0.347
myciel7	191	2360	8	8	5.3352	8	4.008
queen5-5	25	160	5	5	2.3717	5	0.0154
queen6-6	36	290	**7**	**8**	2.5062	**7**	2.039
queen7-7	49	476	**7**	**8**	2.8361	**7**	0.9938

For each instance in Table 1, 10 independent runs of algorithm are carried out. The comparison results show that our algorithm runs faster than HPGA, meanwhile the chromatic numbers are same. In addition, our algorithm can find less chromatic numbers for some graph instances such as queen6-6 and queen7-7 by using less computational time. Moreover, our algorithm can solve larger graph instance with more vertices and edges, such as 2-FullIns-5 and 1-Insertions-6. In the experiment, our algorithm adopted the same values of parameters for all benchmark graphs. It's possible and promising to improve the solution of coloring by fine-tuning values of parameters for each graph.

5 Conclusions

In this paper, we present a novel discrete particle swarm optimization algorithm to solve the GCP. We have improved originally continuous PSO algorithm to discrete problem. In order to apply originally particle swarm optimization algorithm to discrete problem, we design and redefine the crucial position and velocity operators on discrete state space. Therefore, the Discrete PSO algorithm is applied to discrete problem and maintains advantages in optimization problem because of its fast convergence rate, simple computation and easy realization. Moreover, the performance of our algorithm is compared with the other graph coloring method by using 30 DIMACS benchmark graphs. The comparison results show that our algorithm is more competitive with less computation time. Moreover, our algorithm can find less chromatic numbers and solve larger graph instances.

The human brain has rich computation intelligences obtained from millions of years evolution, which has provided plenty of ideas to construct powerful artificial intelligent computing models. Recently, neural-like computing models, see, e.g. [6,9,15,16,22,23], have been used in solving computational hard problems. It is of interests to use neural-like computing models to solve graph coloring problem.

Acknowledgments. This work was supported by the National Natural Science Foundation of China (Nos. 61402187, 61472293, 61273225, 61403287 and 31201121), and the Natural Science Foundation of Hubei Province (Grant No. 2015CFB335), and the Youth Foundation of Wuhan University of Science and Technology (Grant No. 2015xz017).

References

1. Bianco, L., Caramia, M., Olmo, P.D.: Solving a preemptive project scheduling problem with coloring technique. In: Weglarz, J. (ed.) Project Scheduling Recent Models, Algorithms and Applications, pp. 135–145. Kluwer Academic Publishers, US (1998)
2. Werra, D.D.: An introduction to timetabling. Eur. J. Oper. Res. **19**, 151–162 (1985)
3. Kannan, S., Proebsting, T.: Register allocation in structured programs. J. Algorithms **29**, 223–237 (1998)
4. Smith, K., Palaniswami, M.: Static and dynamic channel assignment using neural networks. IEEE J. Select. Areas Commun. **15**, 238–249 (1997)
5. Maitra, T., Pal, A.J.: Noise reduction in VLSI circuits using modied GA based graph coloring. Int. J. Control Autom. **3**(2), 37–44 (2010)
6. Song, T., Pan, L., Jiang, K., et al.: Normal forms for some classes of sequential spiking neural P systems. IEEE Trans. NanoBiosci. **12**(3), 255–264 (2013)
7. Leighton, F.T.: A graph coloring algorithm for large scheduling problems. J. Res. National Bureau Stan. **84**, 489–505 (1979)

8. Brlaz, D.: New methods to color vertices of a graph. Commun. ACM **22**, 251–256 (1979)
9. Song, T., Pan, L.: Spiking neural P systems with rules on synapses working in maximum spikes consumption strategy. IEEE Trans. NanoBiosci. **14**(1), 38–44 (2015)
10. Mouhoub, M.: A hierarchical parallel genetic approach for the graph coloring problem. Appl. Intell. **39**(3), 510–528 (2013)
11. Hong, B.: Generic algorithm of color planar graph. J. Guizhou Univ. (Nat Seil) **11**(16), 232–297 (1999)
12. Wang, X.H., Zhao, S.M.: Ant algorithms for solving graph coloring. J. Inner Mongolia Agric. Univ. **9**(26), 79–82 (2005)
13. Salari, E., Eshghi, K.: An ACO algorithm for graph coloring problem. IEEE Serv. Center **1**, 20–21 (2005)
14. Hertz, A., Werra, D.: Using tabu search techniques for graph coloring. Computing **39**, 345–351 (1987)
15. Song, T., Pan, L., Păun, G.: Asynchronous spiking neural P systems with local synchronization. Inf. Sci. **219**, 197–207 (2013)
16. Song, T., Pan, L., Wang, J., et al.: Normal forms of spiking neural P systems with anti-spikes. IEEE Trans. NanoBiosci. **11**(4), 352–359 (2012)
17. Wang, X.H., Wang, Z.O., Qiao, Q.L.: Artificial neural network with transient chaos for four-coloring map problems and k-colorability problems. Syst. Eng. Theory Pract. **5**, 92–96 (2002)
18. Kumar, P., Singh, A.K., Srivastava, A.K.: A novel optimal capacitor placement algorithm using Nelder-Mead PSO. Int. J. Bio-Inspired Comput. **6**(4), 290–302 (2014)
19. Ram, G., Mandal, D., Kar, R., Ghoshal, S.P.: Optimal design of non-uniform circular antenna arrays using PSO with wavelet mutation. Int. J. Bio-Inspired Comput. **6**(4), 424–433 (2014)
20. Zhang, X., Tian, Y., Jin, Y.: A knee point driven evolutionary algorithm for many-objective optimization. IEEE Trans. Evol. Comput. (2014). doi:10.1109/TEVC.2014.2378512
21. Zhang, X., Tian, Y., Cheng, R., Jin, Y.: An efficient approach to non-dominated sorting for evolutionary multi-objective optimization. IEEE Trans. Evol. Comput. **19**(2), 201–213 (2015)
22. Song, T., Pan, L.: Spiking neural P systems with rules on synapses working in maximum spiking strategy. IEEE Trans. NanoBiosci. **14**(4), 465–477 (2015)
23. Zhang, X., Pan, L., Paun, A.: On the universality of axon P systems. IEEE Trans. Neural Netw. Learn. Syst. (2015). doi:10.1109/TNNLS.2015.2396940