This article can be cited as H. Zapata, N. Perozo, W. Angulo and J, Contreras, A Hybrid Swarm Algorithm for Collective Construction of 3D Structures, International Journal of Artificial Intelligence, vol. 18, no. 1, pp. 1-18, 2020.

A Hybrid Swarm Algorithm for Collective Construction of 3D Structures

Henry Zapata[1], **Niriaska Perozo**[1], **Wilfredo Angulo**[2] **and Joyne Contreras**[1]

[1]Artificial Intelligence Research Center, Decanato de Ciencias y Tecnología
Universidad Centroccidental Lisandro Alvarado
Barquisimeto 3001-Venezuela
Telf.: +582512591720, Fax: +582512591718
henryzapata16@hotmail.com; nperozo@ucla.edu.ve; joynecontreras@ucla.edu.ve

[2]Facultad de Ciencias Naturales y Matemáticas
ESPOL Polytechnic University, Escuela Superior Politcnica del Litoral, ESPOL
Campus Gustavo Galindo Km. 30.5 Va Perimetra, P.O. Box 09-01-5863
Guayaquil, Ecuador
wangulo@espol.edu.ec
**Corresponding Author: Niriaska Perozo, nperozo@ucla.edu.ve*

ABSTRACT

In order to decrease the time of construction of self-assembly algorithm based on the construction of wasp nests, with respect to the number of iterations, this work proposes a novel hybrid swarm algorithm combining strengths of self-assembly and the particle swarm optimization. The paper also considers integration of adaptive values of inertia to further balance exploration and exploitation for improving the construction process. According to the obtained results, it is shown experimentally with two types of benchmark structures that the convergence speed of our hybrid swarm algorithm is considerably improved because each structure is built complete on a lower number of iterations compared with the classical algorithm.

Keywords: Cooperative Construction, Multiagent Systems, Swarm Intelligence, Particle Swarm Optimization, Bio-Inspired Algorithm.

2000 Mathematics Subject Classification: 68T05.

Computing Classification System (CCS): I.2.11.

1 INTRODUCTION

Self-assembly is the ubiquitous process by which objects autonomously assemble into complexes. Nature provides many examples: Atoms react to form molecules. Molecules react to form crystals and supramolecules. Cells coalesce to form organisms, (Adleman, 1999). Self-assembly processes are prevalent in nature at various levels of scale, from the subatomic to the galactic, one can observe the seemingly spontaneous formation of structures that are far

more complex than their individual components. Simulation of self-assembling systems has more relevance if construction is to take place in an environment that is dangerous, or altogether inaccessible for human beings. In particular, when the self-assembly is inspired by the construction of wasps nest is called collective construction, (Theraulaz and Bonabeau, 1999). This algorithm is a model for the construction of complex structures from simple micro-rules in a 3D space, and in a multiagent environment that introduces the qualitative stigmergy as a paradigm of communication, (Theraulaz and Bonabeau, 1999). In this paper will be called classic self-assembly algorithm (CSA).

Nowadays self-assembly has important challenges for the construction of objects autonomously at the software and hardware level in different areas: *artificial life, collective robotic, intelligent manufacturing, civil engineering, and so on*, in order to solve real world problems such as: robotic devices and data management in telecommunication systems, applications of solar power systems in outer space, bases on the lunar surface, and various structures at the nanoscale. Also a large class of interesting structures resembling real world objects e.g., building, pyramid, staircases, etc. can indeed self-assemble under local behaviors, and moreover, automatically applying self-assembly as a useful construction tool.

In the CSA, agents move randomly for determining their position. Agents explore a lot by locating the construction area, instead of exploiting the established construction area. This means that the number of iterations needed to build a complete structure is generally very high.

For this reason, it is necessary to include a mechanism to reduce the construction cost of a complete structure as a function of the number of iterations. In this research, the particle swarm optimization (PSO) is used as a mechanism of navigation that keeps the agents moving towards the areas that require construction activity. Thus a trade-off between exploration and exploitation is achieved through the use of PSO with an adaptive inertia to improve convergence speed of CSA.

2 RELATED WORKS

PSO has been employed by many researchers in various areas and has achieved considerable success when is combined with other algorithms, (Blum and Li, 2008), (Zhang, Lee, Yu and Lau, 2017) and (Precup and David, 2019). In swarm intelligence (SI) area, this combination or hybridization is to invent new SI algorithms as a solution necessary in overcoming certain shortcomings observed during the use of classical algorithms. These new algorithms could have new promising features for solving some optimization problems such as (Precup and David, 2019),(Shams, Rashedi, Dashti and Hakimi, 2017), (Vaščák, 2012), (Vrkalovic, Lunca and Borlea, 2018). Others interesting works combined with PSO for taking advantages of its strengths like our proposal are: (Badamchizadeh, Nikdel and Kouzehgar, 2010), (El-Hefnawy, 2014), (Esmin, Lambert-Torres and Alvarenga, 2006), (Niu and Li, 2008), (Gupta, Tazi, Jain et al., 2014), (Holden and Freitas, 2008), (Li and Liang, 2011), (Liu, Niu, Tang and Jiao, 2011), (Mirjalili, Wang and Coelho, 2014), (Yazdani, Nasiri, Azizi, Sepas-Moghaddam and

Meybodi, 2013), (Wen-Jun and Xiao-Feng, 2003) and (Lai and Zhang, 2009). Besides, we can mention other key works for our research: In (Niu and Li, 2008) is implemented the model of self-assembly through a multiagent system, where the agents of the swarm communicate indirectly to achieve the self-assembly of a global structure, they can only perceive the available information, stored or modified through the environment. Also, the behavior of the stigmergy in the 3D space as a system of discreet events is shown. In (Castro, 2011) is developed a set of libraries using Compute Unified Device Architecture (CUDA), for particle swarm optimization and a genetic algorithm, establishing comparisons among the two methods. This work was important for defining objective function required in our work to improve the process of construction of self-assembly with PSO.

3 CSA BASED ON THE COLLECTIVE BEHAVIOR OF THE WASPS

In this model, the wasps can build simple and complex structures socially, their mechanism of construction is based on a natural program formed by sequences of stimulus-response, which each wasp can continue the work of another, in this way creating the stigmergy (indirect communication). For building a structure is needed: *an 3D environment, a multiagent system, a set of microrules*, that is, all stimulating configurations and associated actions, and a lookup table comprising all its microrules that govern the construction of a structure, (Bonabeau, Dorigo and Theraulaz, 1999). The summarized algorithm for self-assembly is shown on Table 1. When starting the algorithm, it is necessary to build the lookup table for the agents. This table provides the rules that the agents follow to deposit a block. Then, the first block is placed in a predefined site. Agents are randomly located in unoccupied sites in the environment. In each iteration all agents evaluate their environment and if their configuration is present in the lookup table, they deposit the specified block in the table, otherwise they do not deposit the block. After that, agents move randomly to a new unoccupied position and repeat actions. Some key elements when implementing the algorithm are described in (Bonabeau et al., 1999) and (Theraulaz and Bonabeau, 1999):

Local Configuration o neighborhood, refers to the adjacent cells to the agent's position. In Figure 1, we can see two examples of neighborhoods in three plane, z + 1, z and z-1. The agent-plane is called z. The agent-plane is called z and the position of the agent is located in the center of the lattice. The location of the agent is represented with a black block. In Figure 2, it is observed that each cell in a plane (z + 1, z, z-1) has a number, which represents its position in it.

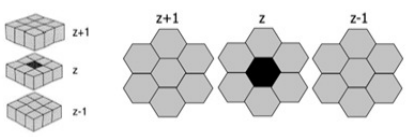

Figure 1: Neighborhood with cubic and hexagonal lattice.

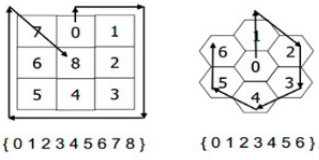

Figure 2: Representation of the positions of the cells.

Algorithm 1: Algorithm of Classical Self-Assembly

```
1  Initialization
2  Construct lookup table; m = Number of Agents;
3  Put one initial block at predefined site;
4  for k=1 to m do                          /* for to distribute the m agents */
5      site=rand();
6      Set agent k in site
7  end
8  for t=1 to t_max do                  /* Iterations to be performed by agents  */
9      for k = 1 to m do                  /* Actions accomplished by each agent */
10         Evaluate local configuration ;
11         if Local configuration is in lookup table then
12             Build the block specified by the lookup table ;        /* Draw new block */
13         end
14         site=rand() ;                  /* randomly selected unoccupied site */
15         Move to site
16     end
17 end
```

Table 1: Algorithm of Classical Self-Assembly

State of Cells, are the possible values that a cell could take on Table 2.

State	Type	Representation
-1	1	Agent
0	2	Unoccupied
1	3	Block type A
2	4	Block type B

Table 2: Example of possible cell states.

Microrule, it is the logical representation of the neighborhood, they are built with the cell states in 3 dimensions, i.e., x = {states in z + 1, states in z, states in z-1}. The agent value is not taken into account. Table 3 shows an example of neighborhood with cubic blocks for which the rule would be: x=0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.

z+1									
Cell	0	1	2	3	4	5	6	7	8
State	0	0	0	0	0	0	0	0	0
Z									
Cell	0	1	2	3	4	5	6	7	8
State	0	0	0	0	0	0	0	0	
z-1									
Cell	0	1	2	3	4	5	6	7	8
State	0	0	0	0	0	0	0	0	1

Table 3: Microrule Construction Example.

Lookup Table, it has the number of the rule, the microrule and the next state of the cell when is evaluated by an agent.

4 PARTICLE SWARM OPTIMIZATION

The PSO algorithm was originally proposed by (Eberhart and Kennedy, 1995), motivated by

intelligent collective behavior such as flocks of birds, schools of fish and even human social behavior. This algorithm simulates the movement of a set of particles, also known as a swarm, in $d_$dimensional search space under the predetermined rules in order to find the optimal position. Each particle is characterized by its position vector $x_i(t) \in \mathbb{R}^d$, velocity vector $v_i(t) \in \mathbb{R}^d$ and associated objective value (associated fitness value) $f(x_i(t))$, (Zhang et al., 2017), (Nickabadi, Ebadzadeh and Safabakhsh, 2011).

Apart from those three attributes, each particle also memorizes its best previous position $p_i \in \mathbb{R}^d$ (local-optimal position) and the best previous position of the swarm $p_g \in \mathbb{R}^d$ (global-optimal position). The particles are randomly placed in the search space at the beginning. Whenever movement occurs, the particles update their velocity vector and position vector according to the following equations:

$$v_i(t+1) = w\, v_i(t) + C_1 R_1(p_i - x_i(t)) + C_2 R_2(p_g - x_i(t)) \quad (4.1)$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (4.2)$$

where $v_i(t)$ is the current velocity vector of the *i*th particle; $x_i(t)$ is the current position vector of $i_$th particle, w is *inertia weight* which control the impact of previous velocity $v_i(t)$ in calculating the new velocity $v_i(t+1)$ of a particle; C_1 and C_2 are the *acceleration coefficients* which control the relative proportion of cognition and social interaction in the swarm (impact of personal and global best position); R_1 and R_2 are two random values uniformly distributed in [0,1], (Bonabeau et al., 1999) and (Miranda, 2005), (Nickabadi et al., 2011), (Precup and David, 2019).

The first term in equation (4.1) represents inertia of particle (it provide information about previous direction of the particle); the second term represents memory of particle (the particle is attracted to the best point in its trajectory, its local-optimal position); the third term represents social interaction or cooperation (the particle is attracted to the best previous position of the swarm, the global-optimal position), (Miranda, 2005).

The fitness of each particle is evaluated using the objective function $f(x)$ and its current position. Then, the best previous position of each particle p_i and the best previous position of the swarm p_g can update according to the following equations:

$$p_i = x_i(t) \quad \text{if} \quad f(x_i(t)) < f(p_i) \quad (4.3)$$

$$p_g = x_i(t) \quad \text{if} \quad f(x_i(t)) < f(p_g) \quad (4.4)$$

The procedure of PSO algorithm is described in table 4.

5 SELF-ASSEMBLY ALGORITHM WITH PSO (SAPSO)

In the CSA, agents move randomly in the search space looking for construction area and then all agents evaluate their environment for build the block specified by the lookup table. Due to

Algorithm 2: The Process of the PSO Algorithm.

Initialize *the position of particles* $x_i(t)$ *randomly,* $v_i(t) = 0$, $p_i = x_i(t)$.
Evaluate *the particles in objective function and initialize* p_g *using equation (4.4)*
while *the termination conditions not met* **do**
    for *all the particles* **do**
        Update the velocity of particles, equation (4.1);
        Update the position of particles, equation (4.2);
        Evaluate particles in objective function;
        Update the local best positions, equation (4.3);
        Update the global best position, equation (4.4);
    end
    Find *the best particle*
end

Table 4: The Process of the PSO Algorithm.

the randomness in movement of agents, they explore a lot by locating the construction area, instead of exploiting the promising area and build structure. This means that the number of iterations needed to build a complete structure is generally very high and the convergence speed of this algorithm could be affected for the large number of iterations required for building a complete structure.

In order to decrease the time of construction of CSA algorithm with respect to the number of iterations, this work proposes a novel hybrid swarm algorithm combining strengths of self-assembly and PSO. It is choosing PSO for its movement rule which are used as a navigation mechanism for agents and also for its similarities with self-assembling, in particular their intelligent behavior: aggregating according to (Zhang et al., 2017).

In the context of **SAPSO** algorithm, the particles of PSO represent the build agents in self-assembly. The purpose is to keep the agents near the construction area and avoid an excessive exploration of the search space, so they can build more blocks in each iteration, this is achieved with PSO.

The optimization problem for PSO is minimizes the distance between the agents and the construction area. The objective function f is defined as the usually Euclidean distance between two points: $\vec{x_i}$ and $\vec{A}$ in 3D space, where $\vec{x_i} = (x_{i1}, x_{i2}, x_{i3})$ represent agent position and $\vec{A} = (a_1, a_2, a_3)$ represent reference point position of the construction area. So the optimization problem is formulated as:

$$\begin{aligned} Min \quad & f(x_i) = \sqrt{(a_1 - x_{i1})^2 + (a_2 - x_{i2})^2 + (a_3 - x_{i3})^2} \\ s.t. \quad & x_i \in D \subset \mathbb{R}^3 \end{aligned} \tag{5.1}$$

where D is the search space.

Initially reference point $\vec{A}$ is the position of the initial block in Self-Assembly. In each iteration of SAPSO the agent moves from their current position towards the reference point $\vec{A}$ using the movement rule of PSO, then they evaluate their environment for building the block specified by the lookup table. If the distance defined in 5.1 is minimized into some iteration t of SAPSO, the swarm must change direction and move towards a new reference point of the construction area. The new reference point will be last built block position denoted $\vec{B}$ and defined by $\vec{B} = \vec{x_i}$. So the optimization problem (5.1) is implemented under the following condition:

$$if\ f(x_i) < 1\ then\ \vec{A} = \vec{B} \tag{5.2}$$

the condition (5.2) allows agents exploit another zone of the construction area and continue building the structure.

The SAPSO algorithm incorporates the PSO movement rule (using the objective function (5.1)) into the self-assembly algorithm to improve the agents exploration. The SAPSO algorithm consists of the following steps (figure 3):

Step 1. Initialization. Define 3D environment. Select microrules and construct lookup table for self-assembly. Put initial block at predefined site (initial position of reference point $\vec{A}$), set iteration index $t = 0$, set the self-assembly process iteration limit t_{max}, set agents index $k = 1$, set the number of agents m, define the *acceleration coefficients* C_1 and C_2, define inertia weight parameter w, initialize randomly the agents position vector and put the agents at their position (show agents at their position into the grid).

Step 2. Evaluate the agents local configuration (it refers to the adjacent cells to the agent's position).

Step 3. Compare if agent's local configuration is in lookup table then go to step 3.1, else go to step 4.

Step 3.1. If local configuration is in lookup table then agent built the block specified by lookup table and update the position of last block $\vec{B} = \vec{x_i}$.

Step 4. Evaluate the agents fitness using the objective function $f(x)$ defined in optimization problem, equation (5.1)

Step 5. Update best position vector of agent according to equation (4.3).

Step 6. Update best global swarm position vector according to equation (4.4).

Step 7. Update velocity vector and position vector of agent according to equations (4.1) and (4.2).

Step 8. Compare, if agent position is an unoccupied site then go to step 9, else go to step 8.1.

Step 8.1. Compare, if agent fitness < 1 then update position of reference point $\vec{A} = \vec{B}$ according to condition (5.2) and go to step 4, else go to step 4.

Step 9. Move the agent at its position (show agent at its position into the grid).

Step 10. Increase k and go to step 2 until $k = m$.

Step 11. Increase t and go to step 2 until $t = t_{max}$. At the end, terminate and show of structure.

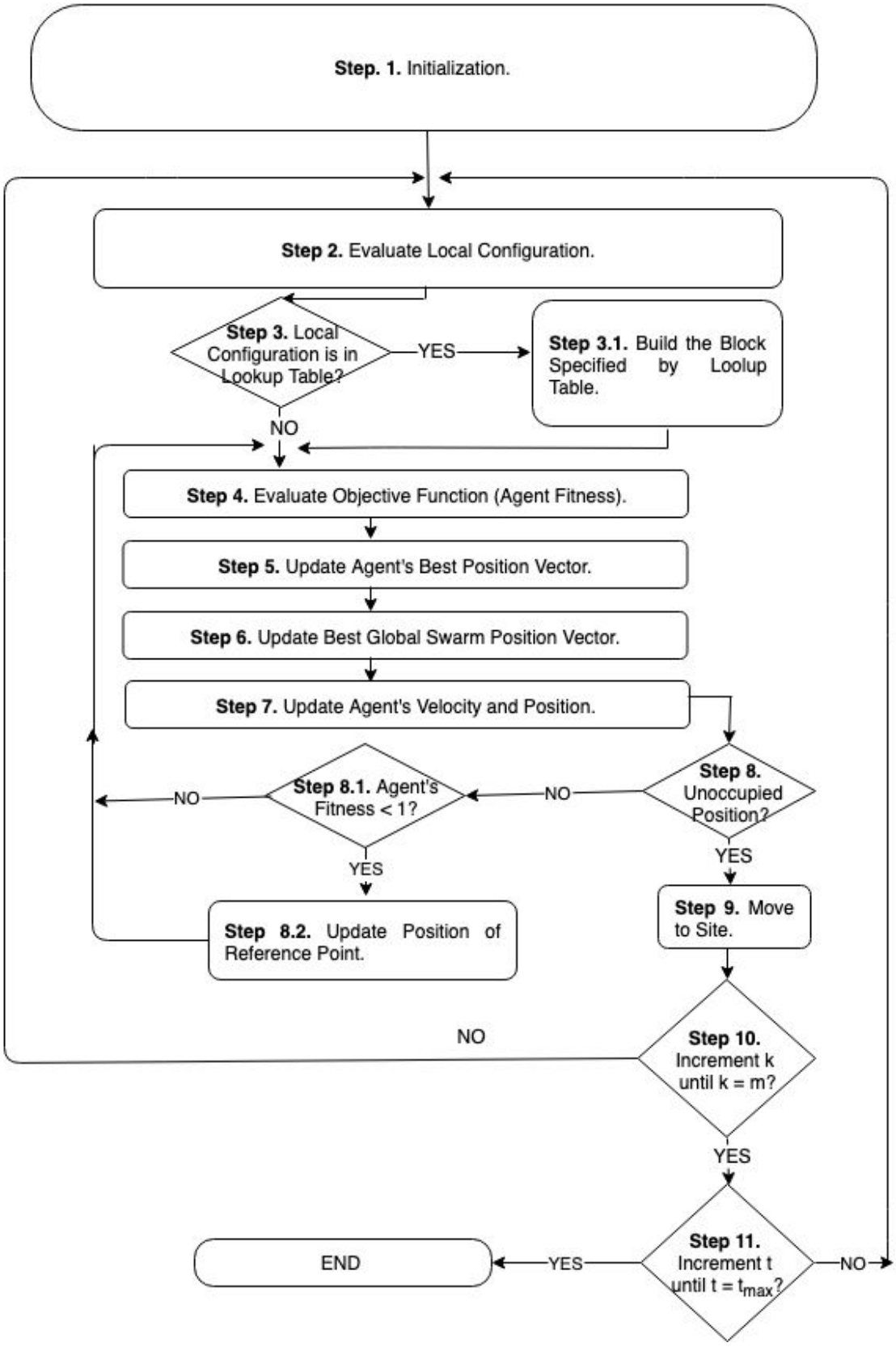

Figure 3: Flowchart of hybrid SAPSO.

5.1 EXPLORATION AND EXPLOITATION CONTROL IN THE SAPSO

Commonly there is some difficulties for choosing a proper value for inertia parameter in 4.1, because its effects are a problem in the trade-off between exploration (time spent by agents in searching) and exploitation (time spent by agents in building), (Nickabadi et al., 2011). Besides, this trade-off between exploration and exploitation is a key issue when adapting any SI algorithm, it could either avoid wasting too much time in unpromising search area or evade the premature phenomenon according to (Zhang et al., 2017). An additional advantage our implementation is that we can dynamically calibrate of inertia parameter through equations (5.3)-(5.9), to control the exploration and exploitation within the construction space defined by the micro-rules, using some mechanisms proposed by (Nickabadi et al., 2011).

$$w(it) = 0.5 + \frac{\tilde{r}}{2} \tag{5.3}$$

$$w(it) = \left(\frac{it_{max} - it}{it_{max}}\right)(w_{max} - w_{min}) + w_{min} \tag{5.4}$$

$$w(it) = \left(\frac{it_{max} - it}{it_{max}}\right)^2(w_{max} - w_{min}) + w_{min} \tag{5.5}$$

$$w(it) = \left(\frac{it_{max} - it}{it_{max}}\right)(w_{max} - w_{min}) + 4\tilde{r}(\tilde{r} - 1)w_{min} \tag{5.6}$$

$$w(it) = \frac{1-\beta}{1-s\beta};\ \beta = \frac{it}{it_{max}};\ s > -1 \tag{5.7}$$

$$w(it) = \left(\frac{2}{it}\right)^{0.3} \tag{5.8}$$

$$w(it) = w_{initial}u^{it};\ \forall u \in [1.0001, 1.0005] \tag{5.9}$$

For all expression about Dynamic Inertia equations (5.3)-(5.9): it denotes the iteration; $w(it)$ denotes the inertia per iteration; $\tilde{r} \in [0, 1]$ is a random number; it_{max} is the maximum number of iterations; w_{max} is the maximum inertia; w_{min} is the minimum inertia and $w_{initial}$ is the initial inertia. A random value of Inertia weight is used to enable the PSO to obtain a trade-of between global and local search in terms of optimization (Nickabadi et al., 2011).

5.2 SOFTWARE FOR SIMULATION

A computational tool developed in MATLAB is used in this work for implementing SAPSO with some features such as: a friendly and interactive user interface which lets visualize in 3D the structure that emerges via self-assembly with and without PSO, it processes and shows graphically the results obtained in the experiments, in relation to the number of built blocks and the distance between the agents and the reference point, (Zapata and Perozo, 2014).

5.3 SETTINGS FOR THE EXPERIMENTATION

Convergence speed is commonly measured using the number of iterations or the CPU time, which is indirectly related to the computational cost, (Zhang et al., 2017). For our experimentation, CSA and SAPSO are compared taking into account the number of built blocks in a

0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1
0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,1,0,2
0,0,0,1,0,0,0,0,0,2,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,2
0,1,0,0,0,0,0,0,0,0,1,2,0,0,0,0,0,0,1,0,0,0,0,0,0,0,2
0,0,0,0,0,1,0,0,0,0,0,2,0,1,0,0,2,0,0,0,0,0,1,0,0,0,2
0,0,0,0,0,0,0,0,1,0,0,0,0,0,2,0,2,1,0,0,0,0,0,0,0,0,1,2
0,0,0,0,0,0,0,1,0,0,0,0,0,0,2,0,1,2,0,0,0,0,0,0,0,1,0,0,2
1,0,0,0,0,0,0,0,0,1,2,0,2,0,0,0,0,1,0,0,0,0,0,0,0,0,2
0,0,1,0,0,0,0,0,0,0,0,2,1,0,2,0,0,0,0,0,1,0,0,0,0,0,0,2

Table 5: Set of 10 micro-rules of the structure that represents the setting 1.

determined number of iterations for measuring the convergence speed, which reflects directly the time of construction of the entire structure. For this, two settings are proposed with two set of micro-rules for building a simple (see table 5 and figure 4) and complex benchmark structure with both algorithms (see table 6 and figure 5).

Setting 1:

A set of micro rules is selected that define a simple structure that grows on the z axis, and evolves in its perimeter according to them (see table 5). It is important to highlight that the figure 4 should emerge after simulation with these micro-rules.

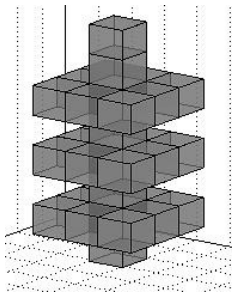

Figure 4: Image of the structure that represents setting 1.

Setting 2:

For this case, a more complex structure is proposed that will evolve in three axes (x, y, z) (see figure 5), depending on the conditions established in the micro-rules (see table 6).

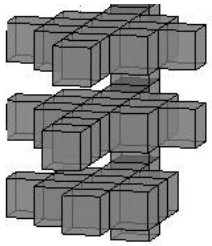

Figure 5: Image of the structure that represents setting 2.

0,0,0,0,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1
0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,2
0,0,0,0,0,2,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,2,0,0,0,2
0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,0,0,0,0,0,0,0,0,0,2
0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,2,0,0,0,0,0,0,0,0,0,0,0,2
0,0,0,0,0,0,0,0,0,0,0,2,2,2,0,0,0,0,0,0,0,0,0,0,0,0,0,2
0,0,0,0,0,0,0,0,0,0,0,0,2,2,2,2,2,0,0,0,0,0,0,0,0,0,0,2
0,0,0,0,0,0,0,0,0,0,0,0,2,2,2,2,2,0,0,0,0,0,0,0,0,0,0,2

Table 6: Set of 9 micro-rules of the structure that represents the setting 2.

6 ANALYSIS OF THE RESULTS

A set of experiments are performed with our computational tool for calculating the quantity of built blocks in a determined number of iterations with CSA and SAPSO respectively to build the structures proposed in settings 1 and 2. The evaluation of each setting is done in two parts: First, it is about building the structure with CSA (table 1); second, the structure is built using SAPSO (figure 3).

6.1 RESULTS OBTAINED IN SETTING 1

Calibration Process for SAPSO. Though different experiments are determined all ideal values of key parameters of PSO such as: inertia (w), cognitive component (C_1), and the social component (C_2), that allow a better construction activity of the agents with SAPSO. To achieve this, two parameters are fixed and one of them is varied in magnitude from 0.1 to 0.9, until finding the values that improve the capacity of construction of the agents (calibration process). Moreover, a set of experiments were performed for testing the mechanisms for dynamic inertia described in the equations (5.3)-(5.9). According to results, the swarm manages to build the structure in most cases, with dynamic inertia faster than fixed inertia, since capabilities of exploration and exploitation of agents take advantage, through to vary the values of inertia. Ideal values obtained for key parameters are: Dynamic Inertia (5.9) with $w_{initial} = 0.9$, $C_1 = 0.3$ and $C_2 = 0.4$.

Structure 1 with CSA. Initial parameters are following: Iterations: 250; Number of Agents: 10; Position of First Block (10, 10, 5). The graphs of each experiment represent per iteration: on the top part, the Euclidean distance between the agent and the reference point (initial block), and on the bottom part, the number of built blocks. According to figure 6, the Euclidean distance between the agents and the reference point was high, thus, there are two built blocks, since the agents were exploring away from the construction area and not building.

Structure 1 with SAPSO. Ideal values are obtained in the calibration process for key parameters. According to setting 1 (see figure 6 and 7), it is shown that in 250 iterations the CSA only built two blocks of the proposed structure, while the agents with SAPSO, in the same experimental conditions, built the entire structure of 32 blocks. which allows us to conclude, that the SAPSO improves the convergence speed and minimize the construction time of this algorithm with respect to the number of iterations considered.

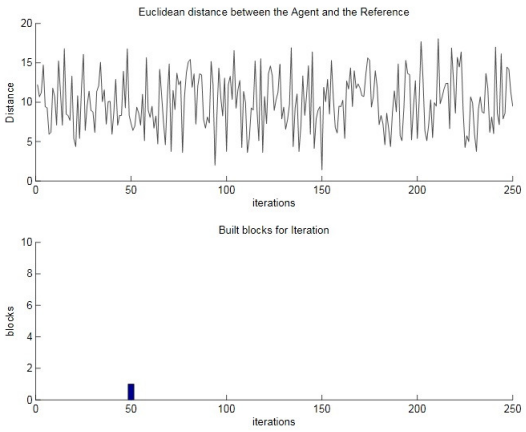

Figure 6: Construction activity of the agents for structure 1 with CSA.

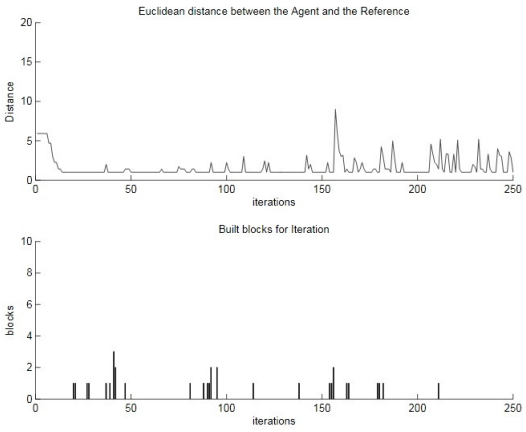

Figure 7: Construction activity of the agents for structure 1 with SAPSO.

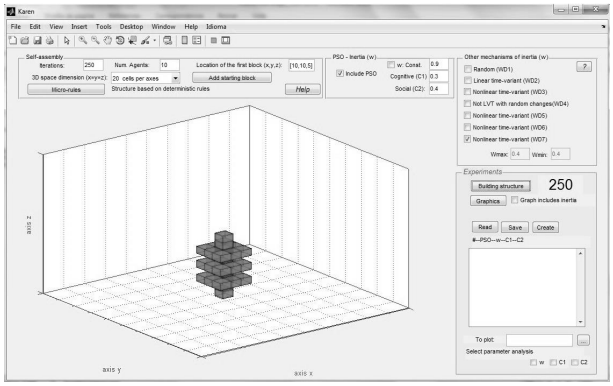

Figure 8: Structure 1 built with SAPSO.

6.2 RESULTS OBTAINED IN SETTING 2

Calibration Process for SAPSO. In this case, the ideal values found in setting 1 will be taken as initial parameters, then the parameters are changed according to table 7, (see from figure 10 through figure 25), besides, the parameter (w) is varied using equations (5.3)-(5.9) for the dynamic inertia. In this manner we can dynamically obtain a value for the inertia that achieves a trade-off between exploration and exploitation in our hybrid algorithm. According to the results obtained in the experimentation, it can be seen that it is feasible to select several options for dynamic inertia to build structure 2 (see figures 19, 21 and 23 respectively), since the values found in setting 1 for social and cognitive component, have a similar behavior when building structure 2. However, the parameters used in figure 19 were selected for this setting with Dynamic Inertia (5.3) with $w_{initial} = 0.9$, $C_1 = 0.3$ and $C_2 = 0.4$. In this case, agents stay near the construction area, and also manage to build efficiently (higher number of built blocks, see figure 26).

Structure 2 with CSA. Initial parameters are following: Iterations: 500; Number of Agents: 10; Position of First Block (10, 10, 5). According to figure 9, it is possible to see that the agents kept exploring away from the construction area, thus, there are few blocks built.

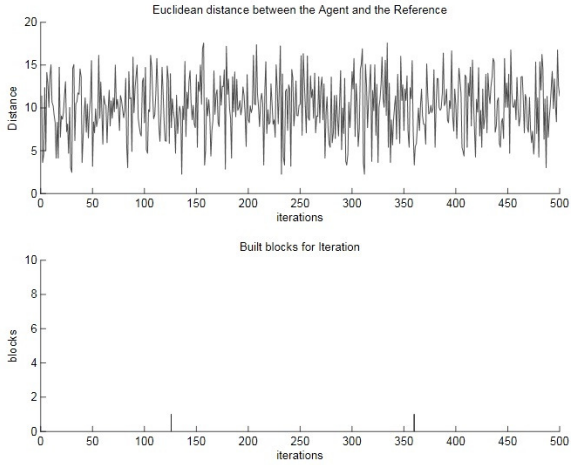

Figure 9: Construction activity of the agents for structure 2 with CSA.

Parameters			Figure	Parameters			Figure	Parameters			Figure
w	C_1	C_2	**N°**	**W**	C_1	C_2	**N°**	**W**	C_1	C_2	**N°**
0.9	0.3	0.4	9	0.9	0.7	0.4	15	Eq. 5.6	0.3	0.4	21
0.7	0.3	0.4	10	0.9	0.3	0.2	16	Eq. 5.7	0.3	0.4	22
0.5	0.3	0.4	11	0.9	0.3	0.6	17	Eq. 5.8	0.3	0.4	22
0.3	0.3	0.4	12	Eq. 5.3	0.3	0.4	18	Eq. 5.9	0.3	0.4	24
0.9	0.1	0.4	13	Eq. 5.4	0.3	0.4	19				
0.9	0.5	0.4	14	Eq. 5.5	0.3	0.4	20				

Table 7: Calibration of the key parameters when building structure 2 with SAPSO.

Structure 2 with SAPSO. In figure 26, it is possible to see a higher quantity of built blocks than figure 9. Besides with SAPSO, agents stay near the construction area (promising areas)

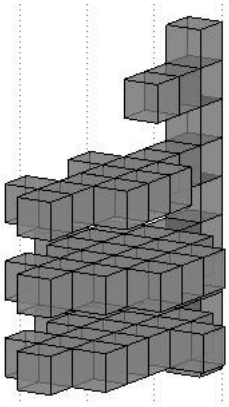

Figure 10:

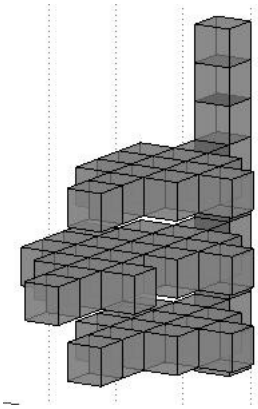

Figure 11:

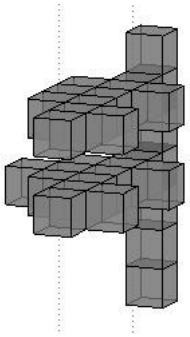

Figure 12:

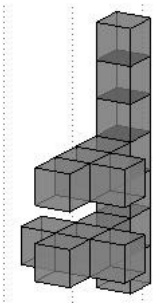

Figure 13:

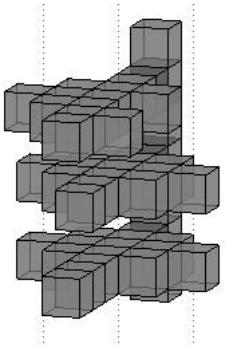

Figure 14:

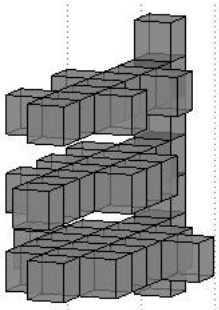

Figure 15:

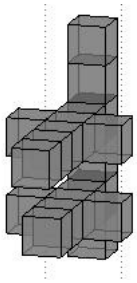

Figure 16:

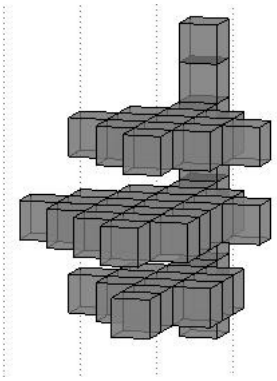

Figure 17:

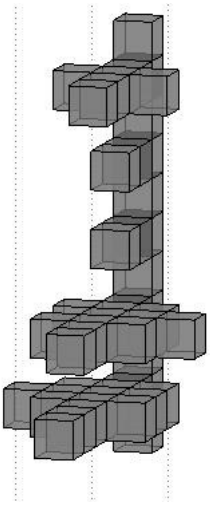

Figure 18:

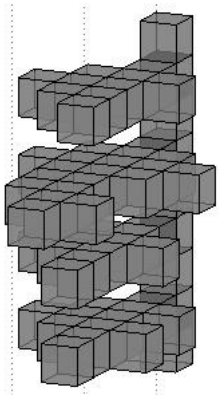

Figure 19:

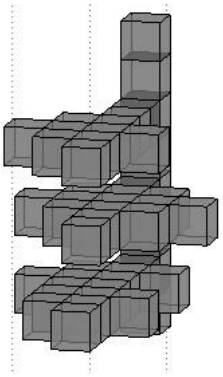

Figure 20:

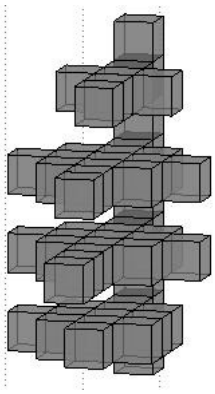

Figure 21:

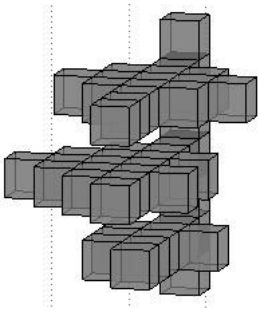

Figure 22:

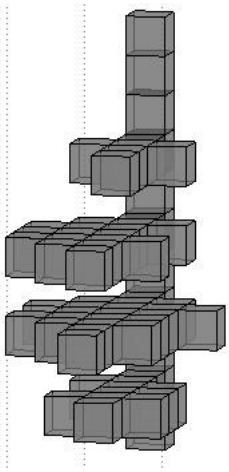

Figure 23:

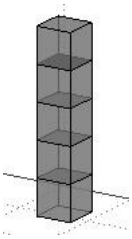

Figure 24:

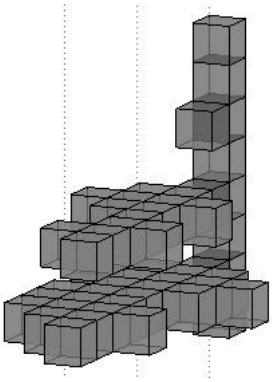

Figure 25:

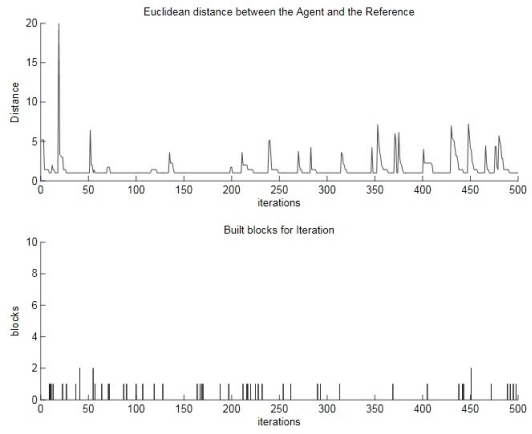

Figure 26: Construction activity of the agents for structure 2 through SAPSO.

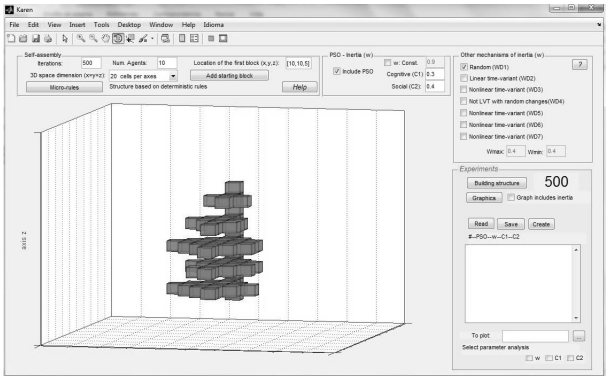

Figure 27: Structure 2 obtained with SAPSO.

for building more efficiently. On the other hand, it is shown how structure 2 emerges through the coordination that agents reach due to qualitative stigmergy (see figure 27).

7 CONCLUSIONS AND FUTURE WORK

In this work, 3D structures were built using SAPSO which is a combination of two emerging techniques (Self-assembly and PSO) with the purpose of improving the process of construction of 3D structures and the convergence speed based on decreasing the number of iterations for building any structure. With a computational tool developed in MATLAB by (Zapata and Perozo, 2014), the process of construction of the agents is simulated with both algorithms, showing that SAPSO builds both structures almost completely in less than 500 iterations, while the CSA does not build any structure in the same number of iterations, thus, the process of collective construction of 3D structures is improved through the particle swarm optimization.

Certain heuristics are established in SAPSO,to allow the swarm to navigate near the area of construction,for example, the objective function is defined to minimize the Euclidean distance

between agents and a reference point selected in promising areas, with the purpose of increasing the capacity of construction of the agents with a balance between exploration and exploitation. Two types of benchmark structures are built in our experiments: a simple one (setting 1) and a more complex one (setting 2) to verify the process of construction of the agents with SAPSO. According to obtained results, the proposed algorithm depends on the calibration of the key parameters: inertia, cognitive and social components and not on complexity of structure for improving the convergence speed, i.e. the time of construction of the agents.

Finally, it is recommended to apply SAPSO in real world problems in order to further evaluate the efficiencies in solving them. Besides, it could be important to consider for further works:

a) The implementation of the self-assembly for the virtual construction of nano structures, adjusting the geometric shape of the cell in the grid to represent the different geometric elements used in nano structures and, additionally, define through micro-rules, what geometric shape could be built according to the conditions of neighborhood, since the micro-rules could be defined according to the laws that rule the behavior of the particles, like the ones found in quantum mechanics.

b) To take the self-assembly to its highest potential, demands computational requirements that increases as the size of the grid does. Because of this, we propose to improve the computational requirements using architecture like CUDA by (Castro, 2011).

c) The analysis of computational complexity of SAPSO is another promising future topic.

References

Adleman, L. 1999. Toward a mathematical theory of self-assembly, *Technical Report, University of Southern California* .

Badamchizadeh, M., Nikdel, N. and Kouzehgar, M. 2010. Comparison of genetic algorithm and particle swarm optimization for data fusion method based on kalman filter, *International Journal of Artificial Intelligence* **5**(10): 67–78.

Blum, C. and Li, X. 2008. *Swarm Intelligence in Optimization*, pp. 43–85.

Bonabeau, E., Dorigo, M. and Theraulaz, G. 1999. From natural to artificial swarm intelligence.

Castro, I. 2011. Parallelization optimization algorithms based on populations through gpgpu, *Oxford University Press* .

Eberhart, R. and Kennedy, J. 1995. A new optimizer using particle swarm theory, *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Japan*, IEEE, pp. 39–43.

El-Hefnawy, N. 2014. Solving bi-level problems using modified particle swarm optimization algorithm, *International Journal of Artificial Intelligence* **12**(2): 88–101.

Esmin, A. A. A., Lambert-Torres, G. and Alvarenga, G. B. 2006. Hybrid evolutionary algorithm based on pso and ga mutation, *2006 Sixth International Conference on Hybrid Intelligent Systems (HIS'06), Brazil*, IEEE, pp. 57–61.

Gupta, M., Tazi, S., Jain, A. et al. 2014. Edge detection using modified firefly algorithm, *2014 International Conference on Computational Intelligence and Communication Networks. India*, IEEE, pp. 167–173.

Holden, N. and Freitas, A. A. 2008. A hybrid pso/aco algorithm for discovering classification rules in data mining, *Journal of Artificial evolution and Applications* **2008**.

Lai, X. and Zhang, M. 2009. An efficient ensemble of ga and pso for real function optimization, *2009 2nd IEEE International Conference on Computer Science and Information Technology. China*, IEEE, pp. 651–655.

Li, Y. and Liang, J. 2011. A hybrid multi-swarm co-evolutional particle swarm optimizer, *International Journal of Artificial Intelligence* **7**(11): 274–287.

Liu, R., Niu, M., Tang, L. and Jiao, L. 2011. Adaptive particle swarm optimization based artificial immune network classification algorithm, *International Journal of Artificial Intelligence* **7**(11): 151–164.

Miranda, V. 2005. Evolutionary algorithms with particle swarm movements, *Proceedings of the 13th International Conference on, Intelligent Systems Application to Power Systems. USA*, IEEE, pp. 6–21.

Mirjalili, S., Wang, G.-G. and Coelho, L. d. S. 2014. Binary optimization using hybrid particle swarm optimization and gravitational search algorithm, *Neural Computing and Applications* **25**(6): 1423–1435.

Nickabadi, A., Ebadzadeh, M. M. and Safabakhsh, R. 2011. A novel particle swarm optimization algorithm with adaptive inertia weight, *Applied Soft Computing* **11**(4): 3658–3670.

Niu, B. and Li, L. 2008. A novel pso-de-based hybrid algorithm for global optimization, *International Conference on Intelligent Computing. China*, Springer, pp. 156–163.

Precup, R.-E. and David, R.-C. 2019. *Nature-Inspired Optimization Algorithms for Fuzzy Controlled Servo Systems*, Butterworth-Heinemann.

Shams, M., Rashedi, E., Dashti, S. and Hakimi, A. 2017. Ideal gas optimization algorithm, *International Journal of Artificial Intelligence* **15**(2): 116–130.

Theraulaz, G. and Bonabeau, E. 1999. A brief history of stigmergy, *Artificial life* **5**(2): 97–116.

Vaščák, J. 2012. Adaptation of fuzzy cognitive maps by migration algorithms, *Kybernetes* **41**(3/4): 429–443.

Vrkalovic, S., Lunca, E.-C. and Borlea, I.-D. 2018. Model-free sliding mode and fuzzy controllers for reverse osmosis desalination plants, *International Journal of Artificial Intelligence* **16**(2): 208–222.

Wen-Jun, Z. and Xiao-Feng, X. 2003. Depso: hybrid particle swarm with differential evolution operator, *Conference Proceedings. 2003 IEEE International Conference on Systems, Man and Cybernetics. Conference Theme - System Security and Assurance (Cat. No.03CH37483)*, Vol. 4, pp. 3816–3821.

Yazdani, D., Nasiri, B., Azizi, R., Sepas-Moghaddam, A. and Meybodi, M. 2013. Optimization in dynamic environments utilizing a novel method based on particle swarm optimization, *International Journal of Artificial Intelligence* **11**(13): 170–192.

Zapata, H. and Perozo, N. 2014. Simulation of cooperative self-assembly 3d structures, *II National Conference of Computing and Systems (CoNCISa). Venezuela*, pp. 33–40.

Zhang, S., Lee, C., Yu, K. and Lau, H. 2017. Design and development of a unified framework towards swarm intelligence, *Artificial intelligence review* **47**(2): 253–277.