On the Fly PSO Inspired Algorithm For Graph Distribution

Djamel Eddine Saïdouni, Ahmed Chawki Chaouche, Jean-Michel Ilié

On the fly PSO inspired algorithm for graph distribution

SAIDOUNI Djamel Eddine[#1], CHAOUCHE Ahmed Chawki[#2], ILIE Jean-Michel[*3]

[#]*MISC Laboratory, Mentouri University*
Constantine, Algeria
[1]saidouni@misc-umc.org
[2]ac.chaouche@misc-umc.org

[*]*LIP6, Pierre and Marie Curie University - Paris 6*
Paris, France
[3]jean-michel.ilie@upmc.fr

Abstract— **This paper deals with clustering applications, it proposes an on the fly algorithm for distributing graphs. The proposed algorithm is based on the meta-heuristic "Particle Swarm Optimization (PSO)". For the sake of presentation, the idea of adapting the PSO algorithm for graphs distribution is developed firstly by assuming that the graph is already generated. After that a distributed algorithm is proposed for concurrently generating and distributing graphs. This distribution should ensure the workload balancing property and the minimization of the number of distributed inter-site edges.**

Keywords— **Distributed system; Distributed algorithms; Graphs distribution; PSO;**

I. INTRODUCTION

Nowadays networks communications are more and more used in almost domains. In particular, they are used for improving the performance of applications. In fact, distributed versions of several applications are proposed for augmenting the storage capacity and accelerating the computing. As an example of such applications, e.g. this is often the case formal verification of systems. The main problem of this application is the combinatorial explosion of state space graph and the related computing. One solution consists in the distribution of the graph.

The graph distribution in the several sites is not an easy task because several factors should be taken into account to have a good distribution. Among the most important of them are the workload balancing of sites in terms of the sub-graphs sizes, and minimization of the distributed inter-site edges belonging distinct sites. This last factor directly contributes to the reduction of the inter-site messages during the distributed verification operation. Hence saving network communication capacity and also time computations.

Among the distribution approaches, the hash coding function MD5 (*MD5*: cryptographic hash function that produces a 128-bit, it has been employed in a wide variety of security application) is largely used over cluster (*Cluster*: group of linked computers, working together closely thus in many respects forming a single computer). In practice, it offers a good compromise between power calculation, used memory and workload [2], [3], [10], [11].

As an important drawback, it generates an excessively rate of the edges linking sub-graphs being distributed. This is due to the distribution function which is generally insensitive with guarding to this parameter. This generates a considerable slowdown of verification stages over the distributed graph. In addition, practical experiment showed that both the workload balancing and reduction distributed inter-site edges [14], [16].

In this paper, we propose an *on the fly* algorithm based on the distribution the system state. We propose to use the meta-heuristic "Particle Swarm Optimization PSO" as an adaptive technique. The aim is to compare with the MD5 algorithm while reducing the number of inter-site edges.

The PSO approach is a well-known approach in optimization areas, that to knowledge, its use for graph distribution is a premier experience.

The paper is scheduled as follows. In section II, the PSO meta-heuristic is reviewed. The section III introduces our formalization of distributed graphs. In Section IV, we show how to re-express the PSO concepts in the context of graph distribution. For the sake of clarity, centralized approach is used first. In section V, we propose an on the fly distributed algorithm for graph distribution. Section VI is dedicated to comparisons again the MD5-based approach. The last section is our conclusion and perspectives.

II. PARTICLE SWARM OPTIMIZATION (PSO)

The particle swarm optimization (PSO) is a class of algorithms inspired from social behavior of animals evolving on swarm, such as migratory birds or fishes. The particles of one same swarm communicate directly together in order to search the best solution. The search process is based on their collective experience [1], [5], [7], [9], [17].

At the initialization step of the algorithm, a swarm is randomly distributed in the search space. Each particle has a random speed too [5]. Then, at each process step:

- Each particle is able to valuing the quality of its position and to remember its best performance, i.e. the best position that it has achieved so far (that can in fact be sometimes the current position) and its quality (the

function value in this position). In what follows, these behaviors can be classified [6], [8]:

- The **"adventurous trend"**, consists to continue according the current speed,
- The **"conservative trend"**, bringing more or less to the best position already found,

- Each particle is able to ask a certain number of its own kind (its neighbors, including itself) and to obtain for each of them its own best performance (the related quality), it is called **"panurgian trend"**.
- At each computing step, each particle chooses the best between best's performances of its neighborhood modify its speed according to this information and its own data and move consequently.

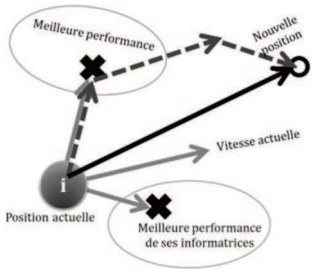


Fig. 1. Particle movement diagram

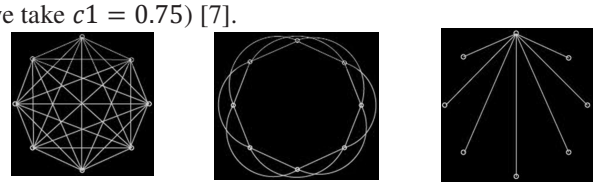
Neighborhood The neighborhood can be defined according by different topologies relating the particles, such that ring, star and beam topologies (Figure 2).

A. PSO algorithm

1) Notation: In the algorithm, the dimension of search space is D. The current position of a particle at the moment t is given by a vector $x(t)$, of D components. Its current speed is $v(t)$. The best position found until now by this particle is given by a vector $p(t)$. Finally, the best of those found by the neighbors of the particle is referred by using a vector $g(t)$ (when t is known by the context, we denote x, v, p and g). The d^{th} component of any of these vectors is indicated by the index d (for example x_d). With these notations, the movement equations of a particle are, for each dimension d:

- $v_d(t+1) \leftarrow c_1 v_d(t) + rand(0, c_{max})(p_d - x_d(t)) + rand(0, c_{max})(g_d - x_d(t))$
- $x_d(t+1) \leftarrow x_d(t) + v_d(t+1)$

For improving the convergence, the values of c_1 and c_{max} must not be chosen independently. In practice, the first must be slightly less than 1 and the second can be calculated by the equation $c_{max} = (2 / 0.97725) \times c_1$. The more c_1 is closed to 1, the good is the exploration of the search space (generally we take $c1 = 0.75$) [7].



(a) Star (b) Ring (c) Beam

Fig. 2. Particles neighborhood topologies

2) The algorithm: [12]

Algorithm 1 *Basic PSO*

```
t ← 0 ;
for Particle p do
    for Dimension d do
        Initialize x_d(t);
        Initialize v_d(t);
    end for
end for
repeat
    Select the particle g_d(t) that has the best fitness in the current iteration;
    for Particle p do
        for Dimension d do
            Calculate the speed v_d(t + 1);
            Update the vector position x_d(t + 1);
            Calculate the value of fitness f(x_d(t));
            If f(x_d(t)) > f(P_d(t)) then
                P_d(t + 1) ← P_d(t);
            end if
        end for
    end for
    t ← t + 1 ;
until (termination criteria verified)
```

Termination criteria if the following two conditions are verified:

- $t = t_{max}$
- $Global\ best\ perf - best\ perf\ of\ P \leq 0.00001(eps)$

3) Algorithm parameters:

- The problem dimension D,
- The number of particles N,
- The size of neighborhood K,
- The values of coefficients C_1, C_{max},
- The search space interval $[X_{min}, X_{max}]_D$,
- The fitness function f.

III. Distributed graphs

Definition 1: An directed graph G is defined by $G = \langle V, E \rangle$ such that:

- V : a finite set of nodes.
- E : a finite set of edges connecting the nodes such that :
  - $E \subseteq V^2$.
  - α and β are two applications of E in V such that every edge $e = (v, v')$ denotes: $\alpha(e) = v$, is the origin of the edge and $\beta(e) = v'$ its starget.

Notice 3.1: The nodes v and v' of V are said adjacent iff $(v, v') \in E$ or $(v', v) \in E$.

Let $W = \{W_k\}_{k=1..N}$ be N sites, a distributed graph (namely *DiG*), is a graph associated with a partial distribution function f^k.

$$DiG = \langle G, f^k \rangle_{k=1..N}$$

such that :

- $G = \langle V, E \rangle$ is an directed graph.
- $f^k : G \longrightarrow G_k$ is an application of G in G_k, such that $G_k = \langle V_k, E_k \rangle$.

Notation 3.1: $\{G_k\}_{1 \leq k \leq N}$ is a set of subsets called fragments G_k, such that $\uplus V_k = V$ and $\uplus E_k = E$.

Definition 2: A fragment G_k is defined by $G_k = \langle V_k, E_k \rangle$ such that :

- $V_k \subseteq V$: V_k is a fragment of nodes of V in the site W_k.
- $E_k = E_k^L \cup E_k^R$ such that $E_k^L \cap E_k^R = \emptyset$ is the set of intra-site and inter-site edges with :
  - $E_k^L \subseteq V_k^2$ is the set of edges between nodes belonged in the same site W_k (Local edges).
  - $E_k^R \subseteq V_k \times (V \setminus V_k) = \{(v_k, v'_k)$such that $v_k \in V_k$ and $v'_k \notin V_k \}$ is the set of edges whose the origins are in the local sites and the stargets are in the remote sites (Remote edges).
  - α_k et β_k are two applications of E_k in V such that for every edges $e = (v, v') \in E$:
    - $\alpha_k(e) = v \in V_k$ indicates the origin of the edge e.
    - $\beta_k(e) = v' \in V_k$ if $e \in E_k^L$ and $\beta_k(e) = v' \notin V'_k$ else.

IV. Graphs distribution based on PSO

A naive application of the PSO algorithm for the optimal distribution of graphs assumes the existence of all the possible solutions (the set of possible distributions of the graph). Thus, a solution can be defined by the set of sites each one with a sub-graph, the optimal solution will be denoted by the set $\{(W_k, G_k^{op}), k = 1..N\}$ (Figure 3). In our context a particle will be defined by a site and a sub-graph that it contains. The movement of particles of the swarm begin from the initial state defined by $\{(W_k, \emptyset), k = 1..N\}$. The moving of a particle P_k ($P_k = (W_k, G_k)$) hence result in the nodes of added to its sub-graph G_k. After this movement the particle is so in the state $(W_k, G_{k'})$ such that $G_k = G_k + new\ nodes$.

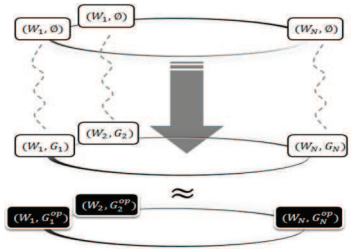


Fig. 3. The algorithm evolution

The adaptation of the PSO algorithm to graph distribution problem requires the definition of the current speed of particle moving as well as its new speed.

Definition 3: The current speed of a particle P_k ($P_k = (W_k, G_k)$) is defined by the more suitable adjacent node in term of connectivity with the sub-graph. This node constitutes its best performance.

In Figure 4, the current speed of the particle $P_2 = (W_2, G_2)$ is the adjacent node v_3. Assuming that this node is attributed to W_2, its new speed will be the node v_2. The dynamic evolution of the algorithm is explained through the example of the section IV-A.

Notation 4.1: Let a particle $P_k = (W_k, G_k)$:

- $Adj(P_k)$: adjacent nodes of P_k,
- $Adj^+(P_k)$: current speed of P_k,
- $N^+(v, P_k)$: defines the particle P_j to which the node v will be added. The computation of this particle is started by the particle P_k (P_j can be P_k).

In the example of Figure 4:

- $Adj(P_2) = \{v_1, v_2, v_3\}$,
- $Adj^+(P_2) = \{v_3\}$,
- $N^+(v_3, P_2) = P_2$,

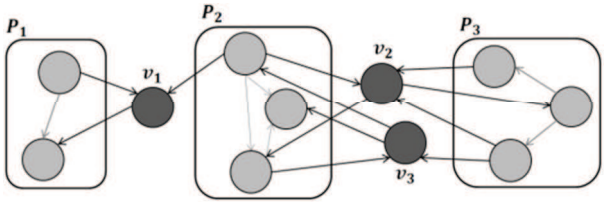
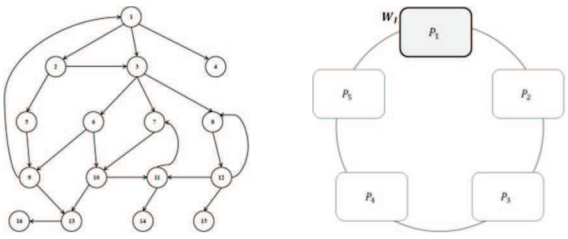


Fig. 4. Particles speeds

A. *Illustrative example*

By considering centralized treatment, the particles of the swarm are assimilated to sites. Each one W_k with sub-graph G_k.

The next example illustrates the different steps of the algorithm for the distribution of a graph. We have 5 particles in the nets ($\{(W_1, G_1), \dots, (W_5, G_5)\}$). The initiator site of calculation is W_1 (Figure 5.b). The graph to be distributed is composed of 16 $nodes$ ($V = \{v_1, v_2, \dots, v_{16}\}$) (Figure 5.a).



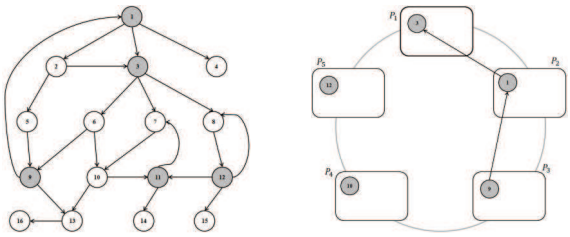
(a) (b)

Fig. 5. Initial state

In the initial state, the particles are defined by $\{W_i, G_i^0, i = 1..N\}$ such that $G_i^0 = \emptyset$.

All the sites are idle with the exception of the initiator site W_1 that ensures the computing (centralized execution of the algorithm). Thus, W_1 defines the structure of 5 $subgraphs$ $(G_1, \dots, G_5)$ that will be associated to sites $(W_1, \dots, W_5)$ which constitute the final states of particles $(P_1, \dots, P_5)$. Indeed, at each iteration, it chooses the 5 best nodes in term of connectivity with the others nodes of the graph. The selected nodes are attributed respectively to the 5 particles (the 5 sub-graphs G_k) (Figure 6.a). The initialization phase is followed

by the creation of local and remote edges related to each particle (P_k). The selected nodes are removed from the set of remaining nodes R in W_1(Figure 6.b).



(a) (b)

Fig. 6. Initialization step

As a result of the initialization step, the cardinality of G_k^1 is similarly to one. The initiator site W_1 treats the remaining nodes according to the order defined by the circulation of the token in the virtual ring linking the particles ($P_1, \dots, P_5$) (Figure 7).

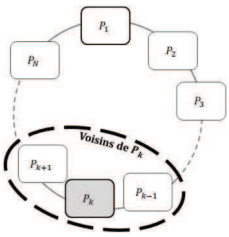


Fig. 7. P_k Neighbourhood

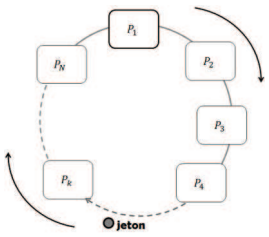
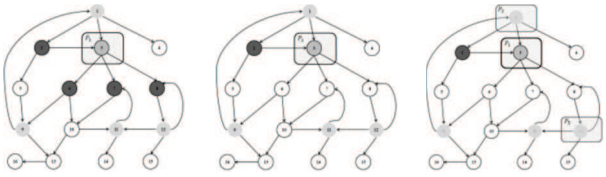


Fig. 8. Virtual ring

For each G_k, the initiator site calculates $Adj(G_k)$ (whatever incoming or outgoing edges) (Figure 9.a), and chooses the best node defined by $Adj^+(G_k)$. In case of equality between nodes, the choice is arbitrary (Figure 9.b). At the last step, the selected node is attributed to the most convenient sub-graph according to the adjacency degree implied by the sub-graphs nodes (the related sub-graphs are G_k and its two neighbors) (see Figure 9.c).

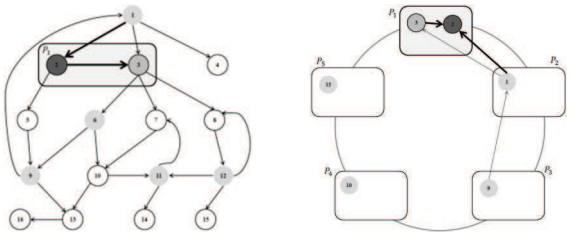


(a) (b) (c)

Fig. 9. 1st iteration of the algorithm according to Figure 5

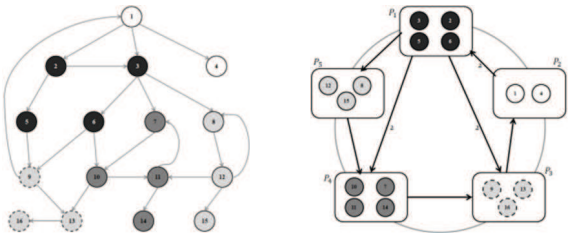
The addition of a node to a sub-graph G_k changes the adjacence relation of G_k (Figure 10).

The distributed graph is given by Figure 11.a. So that sub-graphs can be distinguished. Figure 11.b illustrates the repartition of the graph on the different sites. The expected result relates to the workload balancing and the minimization of the inter-site edges. These can be through the example.



(a) (b)

Fig. 10. Updating after the add of a node on P_1



(a) (b)

Fig. 11. Distributed graph representation

B. PSOGD algorithm

PSOGD: PSO inspired Graph Distribution

1) Hypotheses :

H1. $W = \{W_k\}_{1 \le k \le N}$: is a finite set of N sites,

H2. $W_I \in W$: is the initiator site,

H3. $|S| \gg |W|$: is the number of nodes to distribute (the number of graph nodes is assumed to be high w.r.t. the number of sites),

H4. $\{G_k\}_{1 \le k \le N}$: is the set of graph fragments in the sites W_k. At first, the graph fragments are empty,

H5. The graph G is generated and initially stored at the site W_I.

Algorithm 2 $PSOGD(G,W)$

Constants : $N = |W|$: number of sites;
Variables : R : set of remaining nodes not attributed;

1: ***Initialization()***;
2: **repeat**
3: **for** $k = 1..N$ **do**
4: $v_k \leftarrow Adj^+(G_k)$;
5: $G_j \leftarrow N^+(v_k, G_k)$;
6: $\boldsymbol{Add(v_k, G_j)}$;
7: $R \leftarrow R \setminus \{v_k\}$;
8: **end for**
9: **Until** $(NP = \emptyset)$
10: Stock G_k to W_k;

V. On the fly graph distribution

In this section, we propose an on the fly algorithm which distributes graphs during their generations. This solution includes two components: on the one hand, the PSOGD algorithm is used for the initialization. At this step M nodes are assigned to each site as seen in section IV. On the other

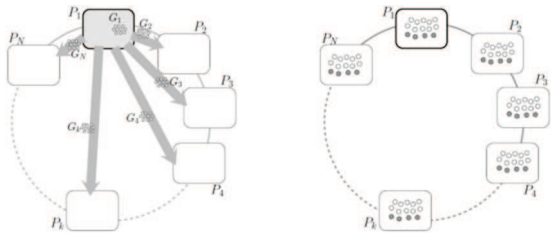
hand, the on the fly algorithm run concurrently over each site (Figure 12.b).

Initialization()

Select N nodes $\{v_k\}_{k=1..N} \in V$;
for $k = 1..N$ **do**
  $V_k \leftarrow \{v_k\}$;
  $E_k^L \leftarrow \{(v_k, v_k)\}$ **if** $(v_k, v_k) \in E$;
  $E_k^R \leftarrow \{(v_k, v_j)\}$ **if** $(v_k, v_j) \in E$ et $v_j \in (V \setminus V_k)$;
  $R \leftarrow V \setminus \{v_k\}_{k=1..N}$;
end for

Add (v_k, G_j)

$V_k \leftarrow V_k \cup \{v\}$;
$\forall e \in E$ such that $(e = (v, v') \vee e = (v', v))$,
if $v' \in V_k$ **then**
  $E_k^L \leftarrow E_k^L \cup \{e\}$;
else
  if $v' \in V_j$ such that $j \neq k$ **then**
    $E_k^R \leftarrow E_k^R \cup \{e\}$;
  end if
end if



(a) (b)

Fig.12. Initialization phase

A. PSOFGD algorithm

PSOFGD : PSO inspired on the Fly Graph Distribution

1) Hypotheses: Assume that H1, H2, H3 and H4 (section IV-B1) are maintained moreover consider the hypotheses H5 and H6:

H5. A part of G with $M \times N$ nodes is generated and distributed by the initiator site W_I using the PSOGD algorithm 2 (N is the sites number while M is the average nodes number in each site after the initialisation step).

H6. A generation process $Succ(v)$ is associated to each site.

Notice 5.1: $Succ(v)$ is a function implemented in each site W_k such that $Succ(v): V \rightarrow P(V)$

For a node v_i, $Succ(v_i) = \{v_i^1, v_i^2, \dots, v_i^j\}$: is the finite set of successor nodes of v_i.

Algorithm 3 $PSOFGD(G, W)$

Constants :
$\boldsymbol{N} = |W|$: number of sites;
$\boldsymbol{M}$: average number of nodes put over each site after the initialisation step;
Variables:
$\boldsymbol{NE} = \{v$ such that $succ(v) \in V = \emptyset\}$: the remaining nodes to be explored;
$\boldsymbol{NE_k}$: the subset of NE in the site W_k;
$\boldsymbol{B}$: array of [2] Boolean; % neighbors responses
$\boldsymbol{NA_k} = \{(v, v', B)\}$: the set of remaining nodes such that v' is the remaining node, v is its predecessor and B is the neighbours responses;

Initialization();
for $W_k \in W$ **do**
  for $v \in NE_k$ **do**
    $NE_k \leftarrow NE_k \setminus \{v\}$;
    for $v' \in succ(v)$ **do**
      if $v' \in V_k$ **then**
        $E_k^L \leftarrow E_k^L \cup \setminus\{(v, v')\}$;
      else
        $NA_k \leftarrow NA_k \cup \{(v, v', \emptyset)\}$
        Send $\boldsymbol{Request(v', k)}$to W_{k+1}; %left neighbor
        Send $\boldsymbol{Request(v', k)}$to W_{k-1}; %right neighbor
      end if
  end for
end for
end for

Initialization()

W_I generates and distributes a graph parts such that $|S| = M \times N$, using the algorithm 2;
Assign each sub-graph G_k to W_k;
$NE_k \leftarrow NE \cap V_k$;

At the reception of $\boldsymbol{Request(v', j)}$

if $(\exists\, v''$ such that $v' \approx v'' \wedge v'' \in V_k)$ **then**
  Send $\boldsymbol{Response(true, v', k)}$ to W_j;
else
  Send $\boldsymbol{Response(false, v', k)}$ to W_j;
end if

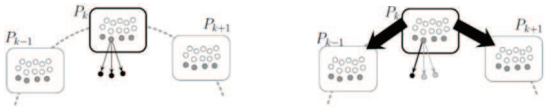
At the reception of $\boldsymbol{Response(exist, v', j)}$

for $(v, v', B) \in NA_k$ **do**
  $B \leftarrow B \cup \{exist\}$;
  if $(exist = true)$ **then**
    $E_k^R \leftarrow E_k^R \cup \{(v, v', W_j)\}$;
    $NE_k \leftarrow NE_k \setminus \{v'\}$;
  end if
  if $(|B| = 2 \wedge (\forall\, b \in B, b = false))$ **then**
    $E_k^L \leftarrow E_k^L \cup \{(v, v')\}$;
    $NE_k \leftarrow NE_k \setminus \{v'\}$;
  end if
end for

2) Algorithm illustration

Initially, W_I generates a sub graph composed of $M \times N$ nodes. These nodes are distributed on the N sites according to the algorithm 2. The initialization step is followed by the exploration of nodes successors. This task is done concurrently by the network sites using the distributed $Succ(v)$ function. The variable NE denotes the set of non explored nodes. This set is distributed on the network sites. $\{NE_k\}_{k=1..N}$ denotes the non explored nodes stored at site W_k (Figure 12.b).

After the initialization phase, which is fully centralized and executed on W_I, the computing is distributed on the network. All sites W_k computes each remaining nodes to be explored sequentially $v \in NE_k$. Given a node v in site W_k, its successors are $Succ(v)$ (Figure 13). Let $v' \in Succ(v)$; the site verifies the existence of the node locally, if it is the case a local edge is created. Otherwise W_k queries its neighbours (W_{k-1} and W_{k+1}) for the existence of v' in their nodes sets (respectively V_{k-1} and V_{k+1}), by sending a request (which contains the node sought v' and the index of the interrogator site k).



(a) (b)

Fig. 13. On the fly generation of a remaining node

At the reception of $Request(v', j)$ by W_k, it checks if there is a node equivalent to v' in V_k and it returns a positive or negative response to the interrogator site W_j.

At the reception of $Response(exist, v', j)$ by W_k, the boolean $exist$ equals $true$ means that the node v' exists in W_j. Hence W_k creates the edge linking the predecessor v to the equivalent of v' (Figure 14).

(a) (b)

Fig. 14. Nodes affectation

Otherwise, after the reception of two negative responses from the two neighbors concerning v' ($exist = false$ and $|B| = 2$), then v' is located neither on W_k nor on its neighbors, it is assigned to W_k.

VI. Results

To show the advantages of the proposed approach, we compare it to the hash function (MD5) based algorithm. From study examples treated in the literature in order to approach more closely to the combinatorial explosion problem in the formal verification applications. As an example, let us consider the paradigm resources allocation by studying the problem of the philosophers dinner. The specification and the graph generation is made under the platform FrameKit [13], [18] (FrameKit: software platform for the systems modeling and the formal verification). More examples may be found in [15].

TABLE I
Results according to 6 philosophers

5 sites	$\|V\|$	σ_V	σ_V%	$\|E^L\|$	$\|E^R\|$	τ	Δ%	*Time*
Centralized	729	–	–	3402	–	–	0%	4
MD5	729	24	3,3%	691	2723	24%	0%	5
PSOGD	729	13	1,8%	2344	1058	69%	0%	120
PSOFGD	896	16	2,2%	1941	1958	51%	23%	35

Table I illustrates the various results for 6 philosophers knowing that the states graph is distributed on 5 sites. Such as σ_V represents the standard deviation of the nodes number ($|V_k|$) on each site, $\sigma_V\% = \frac{\sigma_V}{|V|}$. τ is the rate of local edges compared to all the edges, $\tau = \frac{E^L}{|E|}$ and $\Delta\%$ is the rate of replicated nodes compared to all the nodes, $\Delta\% = \frac{|V_{replicas}|}{|V|}$.

Table I shows that the standard deviation (σ_V) of distributed nodes on the various sites is tiny, this means that the network sites are well balanced.

For the rate of the local edges compared to all the edges, we notice that PSOGD and PSOFGD algorithms are very higher than that of the function MD5. This result is a consequence of the PSO based graph distribution.

Nevertheless, we see that the rate of replication nodes compared to all the nodes of PSOFGD algorithm is considerable. This result is caused by the used neighborhood topology.

The reader may remark that the computing time of the proposed algorithm is more grater that the computing time of the MD5 based algorithm. This is due to the communication cost and additional calculation during the attribution of each node.

VII. Conclusion

In this paper, we proposed an on the fly algorithm based on the meta-heuristic PSO for distributing graphs. For that, we showed on the one hand, how to translate the ingredients constituting the meta-heuristic PSO such that swarms, speed, direction and neighborhood may be adapted to a platform made up of N distributed sites and their progressions in time for distributing graphs. On the other hand, integrating the on the fly algorithm in each site to compute concurrently graph states.

The results showed that the proposed algorithm is very powerful in terms of workload balancing and minimization of distributed inter-site edges. Nevertheless, the computing time is grater than MD5 based distribution algorithm, also nodes are replicated on several sites. This replication problem may be studied in future works.

References

[1] G. Berthiau and P. Siarry. Etat de l'art des méthodes d'optimisation globale. *RAIRO-Operations Research*, 35(3):329–365, 2001.

[2] Zine El-Abidine Bouneb. *Vérification Symbolique des Systèmes Critiques: Approche Distribuée*. PhD thesis, Université de Mentouri, Constantine, Algérie, 2011.

[3] Zine El-Abidine Bouneb and Djamel Eddine Saïdouni. Parallel state space construction for a model checking based on maximality semantics.*CISA*, 2009.

[4] Gary Chhartrand and Ping Zhang. *Chromatic graph theory*. Kenneth H. Rosen. Taylor & Francis Group, LLC, 2009.

[5] M. Clerc and James Kennedy. The particle swarm: Explosion, stability, and convergence in a multidimensional complex space. *In Proceedings of the IEEE Transactions on Evolutionary Computation*, 6:58–73, 2002.

[6] Maurice Clerc. L'optimisation par essaim particulaire, france télécom r&d. *Tutoriel pour OEP 2003*, 2003.

[7] Maurice Clerc and Patrick Siarry. Une nouvelle métaheuristique pour l'optimisation difficile : la méthode des essaims particulaires, october 2003.

[8] Philippe Collard, Manuel Clergue, and Sébastien Vérel. Introduction aux systèmes complexes : Optimisation par essaims particulaires. April 2009.

[9] Antoine Dutot and Damien Olivier. Optimisation par essaim de particules: Application au problème des n-reines. 2004.

[10] Hubert Garavel, Radu Mateescu, and Irina Smarandache. Parallel state space construction for model-checking. tome 2057, 2001. springerlink.com/openurl.asp?genre=article&id=63D7C9EN4BJKAYQW.

[11] Alexandre Hamez. *Génération éfficace de grands espaces d'états*. PhD thesis, Université Pierre & Marie Curie Paris 6, Paris, France, december 2009.

[12] James Kennedy. Small worlds and mega-minds: Effects of neighborhood topology on particle swarm performance. *In IEEE Congress on Evolutionary Computation*, 3:1932–1938, 1999.

[13] Fabrice Kordon. Framekit version 1.4 : Manuel de référence. http://www.lip6.fr/framekit.

[14] Flavio Lerda and Riccardo Sisto. Distributed-memory model checking with spin. In *Proc. of the 5th International SPINWorkshop, tome 1680 de LNCS*, Springer-Verlag, 1999.

[15] Djamel Eddine Saïdouni, Ahmed Chawki Chaouche, and Jean-Michel Ilié. Algorithme de distribution de graphes basée sur pso (version étendue). 2012. http://misc-umc.org/documents/4.pdf.

[16] Ulrich Stern and David L. Dill. Parallelizing the murφ verifier. *Computer Aided Verification*, pages 256–278, 1997. http://citeseer.ist.psu.edu/stern97parallelizing.html.

[17] Abida Toumi, Abdelmalik Taleb-Ahmed, Khier Benmahammed, and Naima Rechid. Optimisation par essaim de particules : Application à la restauration supervisée d'image. 2003.

[18] Jean-Baptiste Voron. Projet coloane: environnement graphique de la plate-forme framekit (plug-in eclipse). http://coloane.lip6.fr.