Synchronous Distributed Load Balancing on Totally Dynamic Networks

Jacques M. Bahi[1], Raphaël Couturier[1] and Flavien Vernier[2]

[1]Laboratoire d'Informatique de l'Université de Franche-Comté (LIFC),
IUT de Belfort-Montbéliard, BP 527,
90016 Belfort CEDEX, France.
email: {jacques.bahi, raphael.couturier}@iut-bm.univ-fcomte.fr

[2]LISTIC - Polytech'Savoie - Université de Savoie,
Domaine Universitaire,
BP 80439,
74944 Annecy le Vieux cedex, France.
email: flavien.vernier@univ-savoie.fr

Abstract

In this paper, first order diffusion load balancing algorithms for totally dynamic networks are investigated. Totally dynamic networks are networks in which the topology may change dynamically. Some edges or nodes can appear, disappear or move during the time. In our previous works on dynamic networks, the dynamism was limited to the edges. The main result of this study consists in proving that the load balancing algorithms reduce the unbalance on arbitrary dynamic networks. Notice that the hypotheses of our result are realistic and that for example the network does not have to be maintained connected. To study the behavior of these algorithms, we compare the load evolution by several simulations.

load balancing, totally dynamic networks, iterative algorithm.

1 Introduction

One of the most important problems in distributed processing consists in balancing the work load among all processors. In distributed systems, the schedules of the load balancing (LB) problem are iterative in nature and their behavior can be characterized by iterative methods derived from the linear systems theory. Local iterative LB algorithms were first proposed by Cybenko in [1], they have been studied and derived by several authors from different points of view [2, 3, 4, 5]. These algorithms iteratively balance the load of a node with its neighbors until the whole network is globally balanced. They have been derived for use on homogeneous or heterogeneous networks [6] with fixed topologies or dynamic topologies but where the dynamism is limited to the edges of the network [7, 8, 9, 10]. But nowadays, with grid, P2P, Ad-Hoc, sensors networks or more generally totally dynamic networks, some nodes appear or disappear in the network during its evolution. More and more applications are dedicated to these networks and the diffusion algoritms can be useful for these applications when no global knowledge can be used.

In this paper, the adaptation on totally dynamic networks of three LB algorithms - first order diffusion (FOS), relaxed diffusion (RFOS) and generalized adaptive exchange (GAE) - are studied. Totally dynamic networks are networks in which the topology may change dynamically. Some edges or nodes can appear, disappear or move during the time. In our previous works on dynamic networks [8, 10], the dynamism was limited to the edges. The main result of this study consists in proving that these algorithms reduce the unbalance of the system on arbitrary totally dynamic networks and converge toward the uniform load distribution if the conditions given in section 3.2 are satisfied. Notice that the hypotheses of our result are realistic (see Theorem 1 and Corollary 1), they are coherent with the behavior of real dynamic networks.

This paper is organized as follows. Section 2 presents

1-4244-0910-1/07/$20.00

the related works, we review the diffusion and the dimension exchange on any static network and dynamic network where the dynamism is limited to the edges. In Section 3, we introduce a graph model for totally dynamic networks and the adaptation of three LB algorithms for these networks. Section 4 illustrates the behavior of FOS on various dynamic topologies and Section 5 concludes this work.

2 Related Works

This section recalls some works on classical diffusion on static networks and the adaptation of three LB algorithms on networks with dynamic edges.

2.1 Classical Diffusion on Static Networks

In [1] Cybenko introduced the diffusion LB algorithm, called First Order Scheme (FOS). This algorithm assumes that a process i balances its load simultaneously with all its neighbors. To balance the load, a ratio $\alpha_{ij} \in]0,1[$ of load difference between the process i and its neighbor j is swapped between i and j. For a process i, the LB step with all its neighbors j is given by

$$w_i^{(t+1)} = w_i^{(t)} + \sum_j \alpha_{ij} \left(w_j^{(t)} - w_i^{(t)} \right),$$

where $w_i^{(t)}$ is the work load done by process i at time t.
In [1] Cybenko also introduced the dimension exchange (DE) algorithm dedicated to binary hypercube topologies. This algorithm assumes that a node i can balance its load with only one of its neighbors at each time step. The choice of a neighbor j is realized using the dimension of the hypercube. This algorithm has been generalized for arbitrary topologies in [4, 5], it is called GDE for Generalized Dimension Exchange. For a process i the GDE algorithm is defined by

$$\begin{array}{ll} w_i^{(t+1)} = w_i^{(t)} + \lambda \left(w_j^{(t)} - w_i^{(t)} \right) & \text{if } i \text{ balances its load} \\ & \text{with a neighbor } j, \\ = w_i^{(t)} & \text{otherwise,} \end{array}$$

where $\lambda \in]0,1[$ is the exchange parameter.

2.2 LB on Networks with Dynamic Edges

Various papers study the LB problem on dynamic networks but they limit their studies to the volatility of edges and do not deal with the volatility of nodes. In [7, 8, 9, 10] the authors introduce two different studies of diffusion LB algorithms on networks with dynamic edges. In [8] a network with dynamic edges is represented by a graph $G^{(t)} = (V, E, E_B^{(t)})$, where V is the set of vertices (processors), E is the set of edges (communication links) and $E_B^{(t)}$ is the set of broken edges at time t. As in static networks, the number of vertices is constant: $|V| = n$. The FOS algorithm on this kind of networks is given by Equation 1.

$$w_i^{(t+1)} = w_i^{(t)} + \sum_j \alpha_{ij}(w_j^{(t)} - w_i^{(t)}) \quad \text{for all living edges } (i,j). \tag{1}$$

A living edge (i,j) at time t is an edge that exists ($(i,j) \in E$) and that is not broken ($(i,j) \notin E_B^{(t)}$). Equation 1 is linear and it expresses the vector Equation 2 that updates load for all nodes at time t.

$$W^{(t+1)} = M^{(t)} W^{(t)} \tag{2}$$

Where $W^{(t)}$ is the vector of $w_i^{(t)}$ and $M^{(t)}$ is defined by

$$m_{ij}^{(t)} = \begin{cases} \alpha_{ij} & \text{if } (i,j) \in E \wedge (i,j) \notin E_B^{(t)} \wedge \\ & i \neq j, \\ 1 - \sum_k \alpha_{ik} & \forall k | (i,k) \in E \wedge (i,k) \notin E_B^{(t)} \\ & \wedge i = j \\ 0 & otherwise. \end{cases}$$

$M^{(t)}$ is the diffusion matrix at time t, it represents the adjacency matrix of the communication graph at this time step. The authors prove that this algorithm converges toward the uniform load distribution under some realistic conditions.

They also give two variants of FOS - relaxed diffusion (RFOS)[8, 7] and generalized adaptive exchange (GAE) [8] - for networks with dynamic links.
The relaxed diffusion algorithm is the diffusion algorithm in which a relaxation parameter $\beta^{(t)}$ is introduced. This parameter speeds up the convergence of the classical diffusion algorithm. The relaxed diffusion algorithm may be described as follows: for a processor i, the exchange of its workload with its reachable neighbor j is executed as Algorithm 3.

$$w_i^{(t+1)} = w_i^{(t)} + \beta^{(t)} \sum_j \alpha_{ij}(w_j^{(t)} - w_i^{(t)}) \quad \text{for all living edges } (i,j) \tag{3}$$

In this algorithm, the diffusion matrix is $(1 - \beta^{(t)})Id + \beta^{(t)} M^{(t)}$. Where Id is the identity matrix, $M^{(t)}$ is equal to the diffusion matrix of FOS and $\beta^{(t)}$ is the relaxation parameter at time t.
The main difference between GAE and diffusion algorithm is that at each LB step, a node balances its load with only one of its neighbors and selects a new neighbor at each time step if it is possible. The choice of a neighbor is free, it can be arbitrary as in GDE algorithm, random or more sophisticated. There is only one condition in the choice of a neighbor j of i: the edge (i,j) must exist and must not be in $E_B^{(t)}$. The GAE algorithm may be described as follows: for

a processor i, the exchange of its workload with a neighbor j is executed as Algorithm 4.

$$\begin{aligned} w_i^{(t+1)} &= w_i^{(t)} + \lambda(w_j^{(t)} - w_i^{(t)}) && \text{if } i \text{ balances its load with its neighbor } j, \\ &= w_i^{(t)} && \text{if } i \text{ does not balance its load at this time.} \end{aligned} \quad (4)$$

In this case the diffusion matrix at time t is given by:

$$m_{ij}^{(t)} = \begin{cases} \lambda & \text{if } i \neq j \wedge i \text{ balances its load with its neighbor } j, \\ 1-\lambda & \text{if } i = j \wedge i \text{ balances its load with a neighbor } k, \\ 1 & \text{if } i = j \wedge i \text{ does not balance its load with a neighbor } k, \\ 0 & otherwise, \end{cases}$$

where j is the chosen neighbor.

In the following section, we study the application of these three algorithms - diffusion, relaxed diffusion and GAE - on totally dynamic networks.

3 LB Algorithms on Totally Dynamic Networks

This section introduces a graph model for totally dynamic networks and the adaptation of the three studied algorithms on these networks.

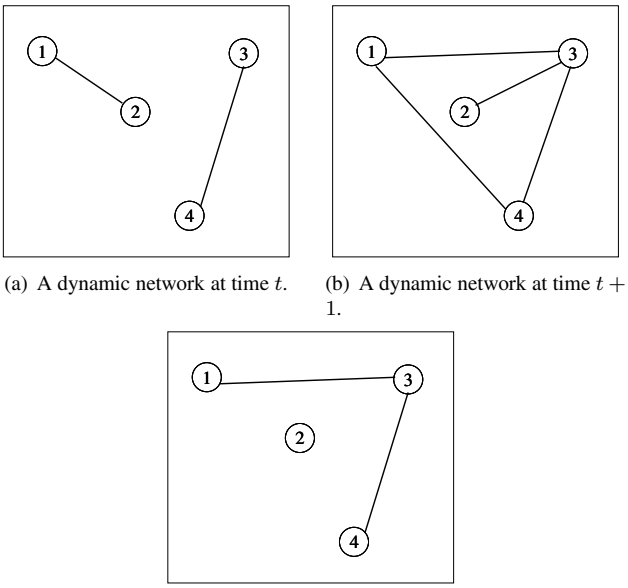
3.1 Graph Model for Totally Dynamic Networks

Classically, a static network topology is represented by a simple undirected connected graph $G = (V, E)$, where V is the set of vertices and E is the set of edges, $E \subseteq V \times V$. Each computing processor is a vertex of the graph and each communication link between two processors i, j is the edge $\{i, j\} \in E$ between the two vertices i and j ($i, j \in V$). By definition, each vertex is labeled from 1 to n where n is the number of processors, thus $|V| = n$. Let m be the number of communication links ($|E| = m$).

A totally dynamic network is a network in which some nodes can appear or disappear and the links can evolve with time, like P2P or ad-hoc networks. So, the graph representation must evolve with time (see Figures 1 and 2). Let us define $V_A^{(t)}$ and $V_D^{(t)}$ respectively the set of nodes appeared at time t and the set of nodes disappeared at time t, and let us define $E_A^{(t)}$ and $E_D^{(t)}$ respectively the set of edges appeared at time t and the set of edges disappeared at time t. At time t a totally dynamic network is represented by a graph $G^{(t)} = (V^{(t)}, E^{(t)})$, where $V^{(t)}$ is the set of vertices at time t and $E^{(t)}$ is the set of edges at time t. Indeed, the number of vertices at time t is given by $n^{(t)}$ ($|V^{(t)}| = n^{(t)}$) and the number of edges is $m^{(t)}$ ($|E^{(t)}| = m^{(t)}$). The evolution of a totally dynamic network can be written as follows.

$$\begin{aligned} G^{(t+1)} &= (V^{(t+1)}, E^{(t+1)}) \\ &= (V^{(t)} \cup V_A^{(t+1)} \setminus V_D^{(t+1)}, \\ &\quad E^{(t)} \cup E_A^{(t+1)} \setminus E_D^{(t+1)}), \end{aligned}$$

where $V^{(t+1)} = V^{(t)} \cup V_A^{(t+1)} \setminus V_D^{(t)}$ and $E^{(t+1)} = E^{(t)} \cup E_A^{(t+1)} \setminus E_D^{(t)}$. In other words, the set of vertices at time $t+1$ is the set of vertices at time t plus the vertices appeared at time $t+1$ minus the vertices disappeared at time $t+1$. It should be noted that $V^{(t)} \cap V_A^{(t+1)} = \emptyset$ - a node can appear only if it is not in the network yet - and $V_D^{(t+1)} \subseteq V^{(t)}$ - a node can disappear only if it is in the network. These two remarks can be applied to the edges, $E^{(t)} \cap E_A^{(t+1)} = \emptyset$ and $E_D^{(t+1)} \subseteq E^{(t)}$.



(a) A dynamic network at time t.

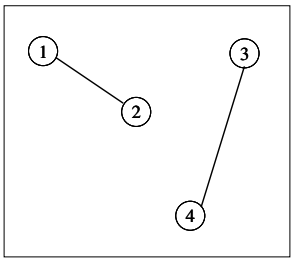
(b) A dynamic network at time $t+1$.

(c) A dynamic network at time $t+2$.

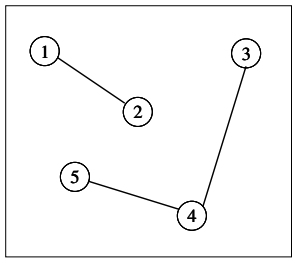
Figure 1. The three first graphs (1(a), 1(b) and 1(c)) illustrate an evolution of a dynamic network. The three last graphs (2(a), 2(b) and 2(c)) illustrate an evolution of a totally dynamic network with an appearance and a disappearance of nodes.

3.2 Diffusion Algorithm

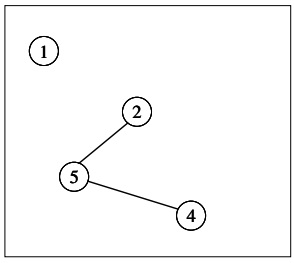
In the context of totally dynamic networks, the standard diffusion scheme requires some adaptations due to the totally dynamic nature of the topology. The main difference



(a) A totally dynamic network at time t with $V^{(t)} = \{1, 2, 3, 4\}$ and $V_A^{(t)} = V_D^{(t)} = \emptyset$.



(b) A totally dynamic network at time $t + 1$ with $V^{(t+1)} = \{1, 2, 3, 4, 5\}$, $V_A^{(t)} = \{5\}$ and $V_D^{(t)} = \emptyset$.



(c) A totally dynamic network at time $t + 2$ with $V^{(t+1)} = \{1, 2, 4, 5\}$, $V_A^{(t)} = \emptyset$ and $V_D^{(t)} = \{3\}$.

Figure 2. The three first graphs (1(a), 1(b) and 1(c)) illustrate an evolution of a dynamic network. The three last graphs (2(a), 2(b) and 2(c)) illustrate an evolution of a totally dynamic network with an appearance and a disappearance of nodes.

lies in a relevant adaptation of the diffusion matrix that needs to dynamically integrate information about the links and nodes modifications.

The diffusion algorithm with totally dynamic networks may be described as follows: for a processor i, the exchange of its workload with its reachable neighbors j is executed as Algorithm 5.

$$w_i^{(t+1)} = w_i^{(t)} + \sum_j \alpha_{ij}^{(t)}(w_j^{(t)} - w_i^{(t)}) \quad \forall (i,j) \in E^{(t)}. \tag{5}$$

Let us note that in Equation 5 α depends on the time, this is due to the dynamism of the network. If the network evolves, α must be able to evolve with it. There exists three classical methods to compute α - Cybenko [1], Boillat [2] or optimal [11] choice - but only one is convenient. Cybenko and optimal choice need a global knowledge of the network, therefore only the Boillat choice is convenient to a totally dynamic distributed system. This method of determining α for a node i only needs a knowledge of neighbors degree. Indeed, $\alpha_{ij}^{(t)}$ can be determined by

$$\alpha_{ij}^{(t)} = \frac{1}{\max(d^t(i), d^t(j)) + 1},$$

where $d^t(i)$ is the degree of node i at time t.

Equation 5 is linear and it expresses the vector Equation 6 that updates load for all nodes at time t.

$$W^{(t+1)} = M^{(t)} W^{(t)} \tag{6}$$

Where $W^{(t)}$ is the vector of $w_i^{(t)}$ and $M^{(t)}$ is defined by

$$m_{ij}^{(t)} = \begin{cases} \alpha_{ij}^{(t)} & \text{if } (i,j) \in E^{(t)}, \\ 1 - \sum_k \alpha_{ik}^{(t)} & \forall k | (i,k) \in E^{(t)} \\ 0 & otherwise. \end{cases} \tag{7}$$

$M^{(t)}$ is the diffusion matrix at time t, it represents the adjacency matrix of the communication graph at this time step. Note that the sizes of $W^{(t)}$ and $M^{(t)}$ can vary with time, they are respectively $n^{(t)}$ and $n^{(t)} \times n^{(t)}$. The difference between Equation 2 and Equation 6 is the construction of $M^{(t)}$. α_{ij} depends on the neighbors number, therefore it depends on t in Equation 7.

The main problem in LB on totally dynamic networks is the volatility of the nodes. If we only consider a volatility of edges, the diffusion algorithm converges towards the uniform load distribution according to the conditions given in [8]. Two cases must be studied for the volatility of nodes, the first one is when a node appears and the second one is when a node disappears.

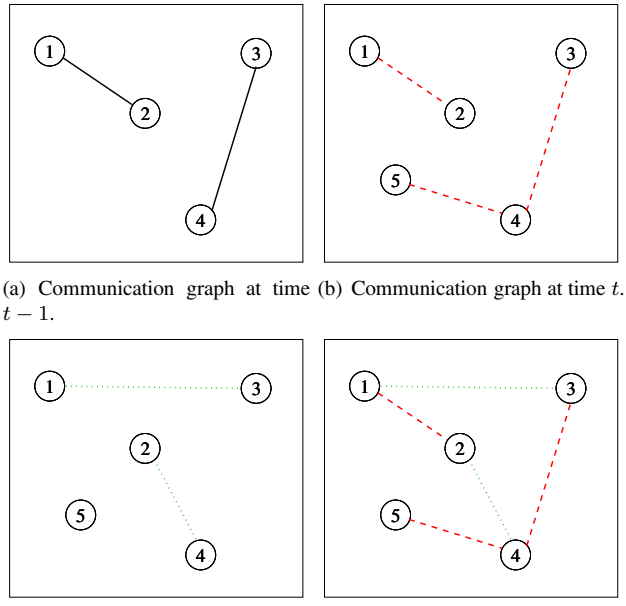
We suppose in this paper that we are in the framework of static load - the global load of the system is constant - on totally dynamic networks. The framework of dynamic load on static networks is studied in [1]. Therefore we consider that a new node has no work load when it connects itself to the network and that the work load of a node that disappears is moved on one or some of its neighbors. It is easy to see that if a node appears or disappears in the network, the new system can be unbalanced due to this node.

Let us note that if nodes appear or disappear, it does not exist a single uniform load distribution. In static and homogeneous networks, the uniform load distribution w^* is given by $w^* = \frac{\sum_i w_i^{(t)}}{n}$, in our case as n depends on t, thus w^* is not constant. Moreover, the two cases presented above show that the networks can be unbalanced when nodes appear or disappear. In fact, our goal on totally dynamic networks is to reduce the over load of the system between two appearances or disappearances of nodes and to reach the uniform load distribution if no node appears or disappears.

To give our main result, we need the following definition:

Definition 1. *A superposed communication graph between the times t and $t + n$, denoted $G_{t,t+n}$, is such that any appearance or disappearance of nodes can only occur at time*

t ($n^{(t)} = n^{(t+i)}, \forall i \in [t, t+n]$). This graph contains all nodes available at time t and all edges (i, j) used for load balancing between the times t and $t+n$.



(a) Communication graph at time t.

(b) Communication graph at time $t-1$.

(c) Communication graph at time $t+1$.

(d) Superposed communication graph $G_{t,t+1}$ between t and $t+1$.

Figure 3. Three communications graphs and the superposed communication graph $G_{t,t+1}$ between times t and $t+1$. The superposed communication graph $G_{t-1,t+1}$ cannot be defined because the nodes number has changed between $t-1$ and t.

Theorem 1. *Algorithm 5 reduces the unbalance between two modifications of nodes number and converges toward the uniform load distribution if the time between two modifications tends to ∞, if and only if to any time t corresponds a time $t+n$ such that the superposed communication graph $G_{t,t+n}$ is a connected graph.*

Proof. It is sufficient to apply the convergence results on networks with dynamic edges given in [8], respecting the previous remarks about the goal of the algorithm. □

It should be noted that, if a node connects and disconnects itself infinitely to the network, the algorithm cannot reach the uniform load distribution.

Corollary 1. *If the conditions of Theorem 1 are not reached between two modifications of nodes number, Theorem 1 can be applied on each connected sub-graph of the superposed communication graph. Indeed, the unbalance of the global system can be reduced.*

Proof. If each connected sub-network (sub-graph) is studied separately, each of them can be considered as a static network between these modifications and the result given in [1] can be applied for each sub-network. □

3.3 Relaxed Diffusion Algorithm

The relaxed diffusion algorithm is the diffusion algorithm in which we introduce a relaxation parameter [8, 7]. So in the totally dynamic networks case, the diffusion matrix $M^{(t)}$ is defined by Equation 7 like the diffusion algorithm. The main problem of the relaxed diffusion algorithm concerns the relaxation parameter $\beta^{(t)}$. Let us recall that $\beta^{(t)}$ is defined by:

$$\beta^{(t)} = \min\left(R^{(t)}, \frac{2}{2-(s^{(t)}+l^{(t)})}\right), \qquad (8)$$

where $s^{(t)}$ and $l^{(t)}$ are respectively the smallest and the second largest eigenvalue of $M^{(t)}$, and $R^{(t)}$ is the relation such that $W^{(t)}$ stays positive if $\beta^{(t)} \leq R^{(t)}$. $R^{(t)}$ is defined by:

$$R^{(t)} = \min_i \frac{w_i^{(t)}}{(1-M_{ii}^{(t)})(w_i^{(t)} - w_{min}^{(t)})} \qquad \forall w_i^{(t)} \neq 0,$$

where $w_{min}^{(t)} = \min_i w_i^{(t)}$.

When the dynamism is limited to the edge, the relation $R^{(t)}$ is an isotone function, but in a totally dynamic case it is not isotone. In this case $\beta^{(t)}$ must be re-computed at each time step and $\beta^{(0)}$ cannot be used for each time. Moreover the calculation of $\beta^{(t)}$ needs a global information about the network and this is not convenient on distributed systems. However, an estimation of $\beta^{(t)}$ can be given: it is known that $\beta^{(t)} \in [1, 2]$, and the most pessimist estimation of $R^{(t)}$ is $\frac{1}{1-M_{ii}^{(t)}}$. Despite this main problem, Corollary 2 can be given.

Corollary 2. *For β chosen according to 8 and under the assumption of Theorem 1, the relaxed diffusion algorithm on totally dynamic networks reduces the unbalance between two modifications of nodes number and converges toward the uniform load distribution if the time between two modifications tends to ∞.*

Proof. It is sufficient to apply the results of [12]. □

3.4 Dimension Exchange Algorithm

As presented in Section 2.2 the GAE algorithm balances the load by peer of processors- a node balances its load with only one of its neighbors at each time step - according to a strategy to determine the peer of processors. The dynamism of the network must be taken into account by the strategy, a neighbor can be chosen only if it exists and only if the

edge with this neighbor is alive . Thus, Algorithm 4 that gives the exchange of workload between a processor i and its neighbor j does not change, it is recalled as follows:

$$\begin{aligned} w_i^{(t+1)} &= w_i^{(t)} + \lambda(w_j^{(t)} - w_i^{(t)}) && \text{if } i \text{ balances its load with its neighbor } j, \\ &= w_i^{(t)} && \text{if } i \text{ does not balance its load at this time.} \end{aligned}$$

Let us note $B_i^{(t)}$ the set that contains the neighbor $b_i^{(t)}$ of i with which it balances its load, if i does not balance its load with any neighbor at time t, $B_i^{(t)}$ is empty. $b_i^{(t)}$ is the neighbor of i chosen by a strategy, so $b_i^{(t)} \in V^{(t)}$ and $(i, b_i^{(t)}) \in E^{(t)}$. With this definition, the diffusion matrix $M^{(t)}$ becomes:

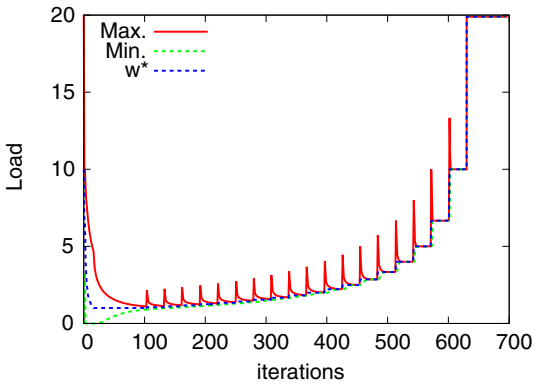
$$m_{ij}^{(t)} = \begin{cases} \lambda & \text{if } (i,j) \in E^{(t)} \wedge i \neq j \wedge j \in B_i^{(t)}, \\ 1-\lambda & \exists k | k \in B_i^{(t)} \wedge (i,k) \in E^{(t)} \wedge i = j \\ 1 & B_i^{(t)} = \emptyset \wedge i = j \\ 0 & otherwise. \end{cases} \tag{9}$$

Corollary 3. *GAE reduces the unbalance between two modifications of nodes number and converges toward the uniform load distribution if the time between two modifications tends to ∞ under the assumption of Theorem 1.*

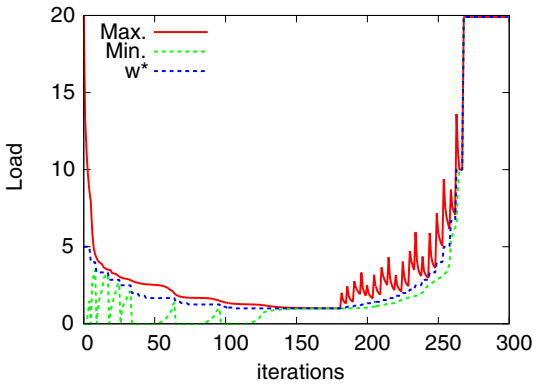
Proof. This is a particular case of Theorem 1 by using $M^{(t)}$ defined by Equation 9. Let us recall that $M^{(t)}$ represents the communication graph at time t (see [8]). □

4 Experimentation

This section presents two experiments of the FOS algorithm on totally dynamic networks. In the first one the nodes appear and disappear in the network and in the second experimentation a node moves and goes through the network. These experiments have been realized in Java with Jace [13] - Java Asynchronous Computation Environment - to manage the totally dynamic networks. The Jace console allows to dynamically manage the used network: some nodes can be added or retrieved during the computation of tasks. These experiments are real implementations on a cluster, only the managed load is virtual, it is represented by a real. It is not represented by an integer value to simplify the experiments. The first experimentation uses totally dynamic networks that start with only one node, some nodes appear during a first stage of the computation and disappear during a second stage to finish with only one node. Three classical topologies have been studied. The first topology is a ring composed of 1 to 20 nodes, when a node appears (at each second) it is connected between the first and the last node of the ring to give a ring with $n+1$ nodes. In the second stage,



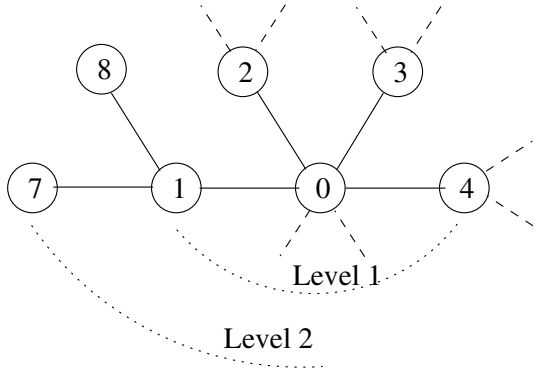
(a) Ring.



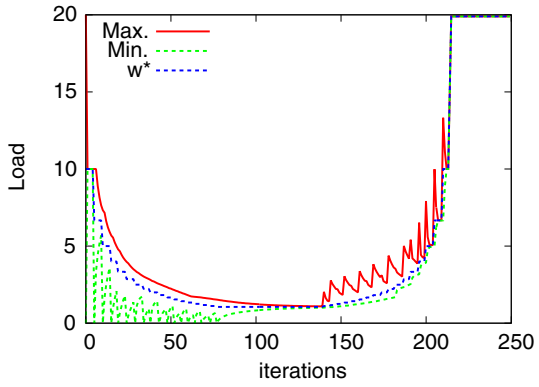
(b) Mesh.

Figure 4. Load evolution on classical dynamic networks

the node that has the smallest index disappears (each $30s$) to give a ring with $n-1$ nodes. The second topology is a 2 dimensional mesh from 1x4 to 5x4, the nodes appear and disappear (each $5s$) by lines of 4 nodes. The last topology is a star with two levels (see Figure 5(a)), the first level is composed of 6 nodes and the second one of 12 nodes, thus each node of the first level is connected to 2 nodes of the second level (a node appears/disappears each $5s$). The results of FOS on these networks are given by Figures 4 and 5. These figures show for each studied topology the load of the most loaded node, the load of the least loaded node and the value of $w^{*(t)}$ given by $w^{*(t)} = \sum_{i=0}^{n^{(0)}-1} w_i^{(0)}/n^{(t)}$ where $n^{(t)}$ is the number of nodes in the network at time t. Recall that $\sum_{i=0}^{n^{(0)}-1} w_i^{(0)} = \sum_{i=0}^{n^{(t)}-1} w_i^{(t)}$. We can observe on each figure the two stages - appearance and disappearance of nodes - and the impact of the interval of nodes number modification. In the ring topology the nodes appear too fast to reach a uniform load distribution between 2 appearances

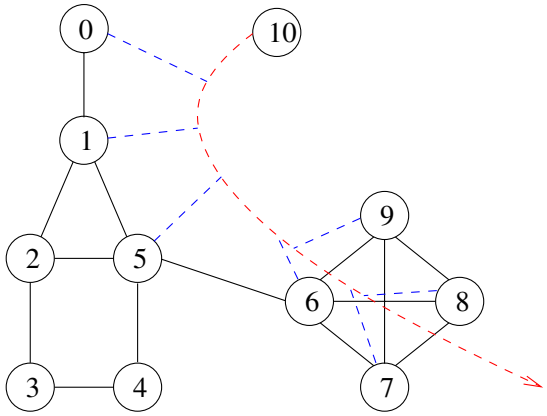


(a) Star topology.

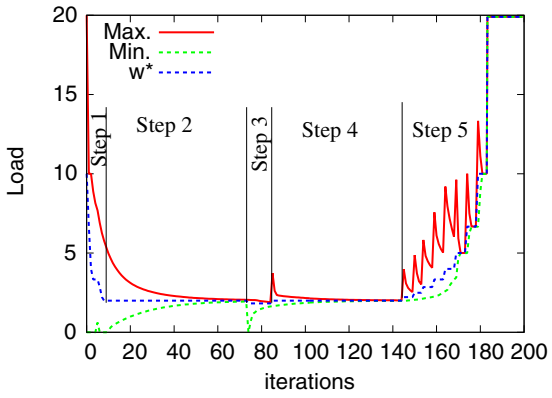


(b) Star.

Figure 5. A 2 levels star topology and its corresponding load evolution.



(a) Network with path of node 10.



(b) Unbalance evolution.

Figure 6. Load balancing on an arbitrary network with a node that goes through it.

(Figure 4(a)). In the other topologies (Figures 4(b), 5(b)), the interval of appearance/disappearance is long enough to illustrate the impact of the dynamism. In the figures each peak on the least loaded line corresponds to an appearance and each peak on the most loaded line corresponds to a disappearance. These peaks are induced by our context, when a node appears it has no load and when a node disappears it gives its load to one of its neighbors. This first experimentation only illustrates the appearance and disappearance of nodes. The next one shows the impact of a node that goes through an arbitrary network topology. Figure 6(a) represents the network with the path of node 10. Along its way, node 10 connects and disconnects to the nearest nodes of the path. Figure 6(b) illustrates the behavior of the most and the least loaded node. This figure can be split in 5 steps. The first one is the building of the network, it is short, the 9 nodes of the network appear in 9 iterations. The second step corresponds to a static network, no node appears or disappears and the LB algorithm converges towards the uniform load distribution. The third step is limited by the two peaks on the middle of the figure, one for the appearance and the other for the disappearance of node 10. The next step is equivalent to the second and the last step is the destruction of the network. The remarks about the first and last steps are the same as the first experiments. The second and fourth steps correspond to a static network. When the node goes through the network, it disturbs the balance twice (near iterations 75 and 85). The first time, the LB algorithm takes a little more time to correct the unbalance than the second time. This is due to the fact that node 10 moves the unbalance throughout the network.

5 Conclusion

This paper extends the diffusion load balancing models - First Order, Relaxed First Order and GAE - to totally dynamic networks. Totally dynamic networks are networks in which some edges or nodes can appear, disappear or move during the time. They are useful when the topology may change as it is the case in Ad-Hoc, P2P or sensor networks and are well-suited for large problems that need to share computations among distant processors, as it is the case in grid computing. In our previous work on dynamic networks, the dynamism was limited to the edges.

To the best of our knowledge, this work is the first one which takes into account volatility of nodes for the diffusion like algorithms. The main result of this paper is that we have given the necessary and sufficient conditions to reduce the unbalance in the totally dynamic network frameworks. We prove that the studied algorithms always decrease the unbalance and reach a uniform load distribution if the conditions of Theorem 1 are satisfied. Finally the paper is concluded by significant experiments, leading to interesting results. These experiments include node or link appearance or disappearance and a node move. This first work will be continued by implementing these algorithms on a real application and by deploying them on real Ad-Hoc, P2P or any totally dynamic network environment.

References

[1] G. Cybenko. Dynamic load balancing for distributed memory multiprocessors. *Journal of Parallel and Distributed Computing*, 7:279–301, 1989.

[2] J.E. Boillat. Load balancing and poisson equation in a graph. *Concurrency: Practice and Experience.*, 2(4):289–313, 1990.

[3] C.Z. Xu and F.C.M. Lau. Optimal parameters for load balancing with the diffusion method in mesh networks. *Parallel Processing Letters*, 4(1-2):139–147, 1994.

[4] S.H. Hosseini, B. Litow, M. Malkawi, J. McPherson, and K. Vairavan. Analysis of a graph coloring based distributed load balancing algorithm. *Journal of Parallel and Distributed Computing*, 10:160–166, 1990.

[5] C.Z. Xu and F.C.M. Lau. Analysis of the generalized dimension exchange method for dynamic load balancing. *Journal of Parallel and Distributed Computing*, 16(4):385–393, 1992.

[6] R. Elsässer, B. Monien, and R. Preis. Diffusion schemes for load balancing on heterogeneous networks. *Theory of Computing Systems*, 35:305–320, 2002.

[7] J.M. Bahi, R. Couturier, and F. Vernier. Accelerated diffusion algorithms on general dynamic networks. *5th International Conference, PPAM Czestochowa, Poland*, pages 77–82, 2003.

[8] J.M. Bahi, R. Couturier, and F. Vernier. Synchronous distributed load balancing on dynamic networks. *Journal of Parallel and Distributed Computing*, 65(11):1397–1405, 2005.

[9] R. Elsässer, B. Monien, and S. Schamberger. Load balancing in dynamic networks. In *7th International Symposium on Parallel Architectures, Algorithms and Networks*, 2004.

[10] F. Vernier. *Algorithmique itérative pour l'équilibrage de charge dans les réseaux dynamiques*. PhD thesis, Université de Franche-Comté (France), 2004.

[11] C.Z. Xu, B. Monien, R. Lüling, and F.C.M. Lau. An analytical comparison of nearest neighbor algorithms for load balancing in parallel computers. In *9th International Parallel Processing Symposium*, pages 472–479. IEEE Computer Society Press, 1995.

[12] A. Berman and R.J. Plemmons. *Nonnegative Matrices in the Mathematical Sciences*. Academic Press, SIAM, Philadelphia, third edition, 1979 edition, 1994.

[13] J. Bahi, S. Domas, and K. Mazouzi. Jace : a java environment for distributed asynchronous iterative computations. In *12-th Euromicro Conference on Parallel, Distributed and Network based Processing, PDP'04*, pages 350–357. IEEE computer society press, 2004.