A stigmergic approach for dynamic routing of active products in FMS

Yves Sallez*, Thierry Berger, Damien Trentesaux

LAMIH, CNRS, University of Valenciennes and Hainaut-Cambrésis, Le mont Houy, F-59313 Valenciennes cedex 09, France

ARTICLE INFO

Article history:
Available online 22 January 2009

Keywords:
Stigmergy
Dynamic routing
Active product
Manufacturing control
Flexible manufacturing system

ABSTRACT

This paper illustrates the capacity of a stigmergic routing control model to automatically find efficient routing paths for active products in flexible manufacturing systems (FMSs) undergoing perturbations. The proposed model is based upon a functional architecture with two levels: a virtual level in which virtual active products (VAPs) evolve stochastically in accelerated time, and a physical level in which physical active products (PAPs) evolve deterministically in real-time. The physical active products follow the best paths that have been detected on the virtual level, with a virtual level exploration being triggered when a perturbation is diagnosed in the transportation system. The data used for the simulation on the virtual level is then updated to reflect the real state of the transportation system. The model’s adaptive capabilities are illustrated with several simulation scenarios using NetLogo software, and an on-going real implementation is presented.

© 2008 Elsevier B.V. All rights reserved.

1. Introduction

This paper is relevant to the context of distributed dynamic control of production processes in flexible manufacturing systems (FMSs). In this context, a stigmergic routing control model, capable of automatically finding efficient routing paths for active products in flexible manufacturing systems undergoing perturbations, is proposed.

Most of the research in the FMS domain focuses on the distributed control of Dynamic Allocation Processes (DAP) (e.g., dynamic scheduling). Little attention has been paid to the distributed control of Dynamic Routing Processes for products (DRP), (e.g., transportation times are assumed constant, paths unique and conveying capacity unlimited) [1–10]. In addition, products are usually considered to be passive entities: they never communicate, decide or act during the routing process. However, recent technological advances (e.g., RFID, smart cards, embedded systems) have led to research on “active products”, in which products are able to act based on the real state of the system (e.g., resources, production and transportation systems). These “active” capabilities may be embedded in the product itself or may operate from a distance [11].

Given these realities, we have begun to work on the distributed control of DRP using the notions of “active product” and stigmergy to provide efficient, real-time, adaptive product routing. These notions make our model more realistic, able to take such aspects as limited system capacity and system reliability into account.

This paper is structured as follows: Section 2 gives a short presentation of the state-of-the-art in the domain of stigmergy applied to manufacturing control. Section 3 presents the assumptions on which our model is based, as well as the notations and variables that are used in subsequent sections. Section 4 describes the model, and Sections 5 and 6, respectively, introduce the developed simulator and report the results obtained. Section 7 provides a short synthesis and Section 8 describes the ongoing implementation. Section 9 offers our conclusions and perspectives for future research.

2. Stigmergy: concepts and the state-of-the-art in manufacturing control

FMS routing is a difficult problem because its nature is stochastic and time-variable. Our objective in this study was to build an efficient routing system that is capable of finding the best routing solutions in real-time and of adapting to new traffic situations and changes in the conveying network’s connectivity (e.g., jamming, failures or slowdowns on the network arc, topological modifications). Some insect societies that use stigmergy – for example, ant colonies – exhibit these desirable properties. However, the customary optimization algorithms (i.e., path-finding) can hardly support such dynamics. For a more detailed discussion of the advantages of stigmergy compared to other optimization approaches (e.g., simulated annealing, tabu search, iterated local search, evolutionary computation), see reference [14]. Once we had built our system, we sought to integrate it into a more general distributed control system. This integration was facilitated by the naturally distributed nature of stigmergic control.
2.1. Concept of stigmergy

The term “stigmergy” describes the mechanism used during ant foraging activities [12]. Ants find food and carry it back to the nest, simultaneously laying down a pheromone trail. Other ants, detecting the pheromones, follow the trail back to the food source. As more ants bring food to the nest, they each reinforce the chemical trail on the path they follow. Since pheromones tend to evaporate over time, the more attractive trails accumulate more pheromones and thus have an advantage over the other trails. Over time, due to the natural reinforcement of the ants, only the shortest trail remains.

2.2. Use of stigmergy in manufacturing control systems

The first experiments related to the industrial use of stigmergy were conducted in the early 1980s by Deneubourg et al. [13], who simulated “ant-like robots”. Dorigo’s work [14, 15] gave rise to a new field of research known as Ant Colony Optimization (ACO), which has been used successfully to solve optimization problems (e.g., Travelling Salesman Problems, Network Routing for telecommunications and the Internet). Several researchers – Parunak [16], Brückner [17], Peeters [18] and Hadeli [19] – have also applied the stigmergy concept to specific situations in manufacturing control systems.

2.3. Use of stigmergy for dynamic routing

Several successful routing algorithms inspired by ant colony behavior have been proposed [14]. In a time-variable context, the most appropriate is AntNet [15], which is an ACO approach applied to adaptive learning for routing tables in communications networks. In AntNet, routing information is gathered through a stigmergic learning process using routing packets. These packets are lightweight agents that are generated concurrently, but independently, by the network nodes, and given the task to sample the paths to an assigned destination. An ant going from a source node ns to a destination node nd collects information about the quality of the path that it follows (e.g., end-to-end delay). Its path is then backtracked from nd to ns to update the routing information at the intermediate nodes.

2.4. Parallel between stigmergic approach and reinforcement learning methods

The routing problem can be seen as a stochastic, distributed and time-variable reinforcement learning problem [20]. Dorigo and Stützle have explained [14] that the main difference between AntNet and the TD(λ) methods is that AntNet requires no information backchaining from one state (i.e., the triplet–current node, destination node and next hop node) to its predecessors. Each state is rewarded only on the basis of the ant’s trip time information strictly relevant to it.

2.5. Specificity of our approach

Our approach to the routing problem was inspired by AntNet. However, certain adaptations were needed to adapt AntNet for use in a FMS context. In AntNet, mobile agents (routing packets) are used to update the routing tables in real-time and to distribute information about the network’s traffic load. In an FMS context, this would be unrealistic since using real shuttles to explore the network would be highly prejudicial in terms of time efficiency.

In response to this problem, we propose a functional two-level – physical and virtual – architecture for the distributed control of DRP. The following section describes the assumptions on which our model is based and presents our modelling approach.

3. Assumptions and the proposed modelling approach

3.1. Assumptions

Our proposed architecture is based upon four assumptions concerning four different system aspects:

1. The topology of the transportation system is assumed to be associated with a strongly connected, directed graph, in which nodes can be both resources and disjunction points, and arcs are the parts of the system that require no decisions during the routing process since the product can only move in one direction towards the next node. Routing times are assumed to be non-negligible compared to production times.

2. The global systemic architecture of the production control system, in this paper, pertains to the distributed DRP and DAP control systems. This architecture is assumed hierarchic (Fig. 1), meaning that allocation is optimized prior to routing. Inputs to the distributed DRP control system are assumed to be the set of pairs (ns, nd), where ns is the resource source node

![Fig. 1. Global systemic architecture of the production control system.](image-url)
and $nd$ the resource destination node, related to one or more products at a given time $t$. Outputs are the optimized real transportation times for routed products.

(3) Memory refers to the presumed ability of both nodes and products to support data processing and memorization, and 

(4) Communication management refers to the presumed ability of nodes and products to communicate node-to-node, node-to-product, and product-to-node.

The last two assumptions are facilitated by new technological innovations that make embedding data processing on products (e.g., smart cards, embedded systems) easier. Since, in this paper, products are assumed to be capable of making routing decisions and of providing information about the real fluidity in the transportation system, all of the above assumptions give products an important role in the production control process, thus justifying the term, “active product”. In this paper, an active product is a decisional entity able to act in order to reach a desired goal. It is composed of a mobile physical body embedded with informational, communicational and decisional capabilities. By analogy, the “active products” used in other fields of application might be:

- the routing packets without communicational and decisional capabilities used in telecommunications networks (e.g., those in AntNet), and
- the intelligent vehicles (i.e., vehicles equipped with GPS and vehicle-infrastructure communications systems) used in transportation networks that are able to detect perturbations in an urban network and to make appropriate routing decisions.

### 3.2. Modelling approach

Fig. 2a illustrates the proposed architecture with its two levels and the intermediary Data Space (DS) that memorizes the information for both levels:

- The virtual level (VL), where virtual active products (VAPs) move, is an informational domain in which everything is simulated in accelerated time as quickly as possible;
- The physical level (PL) represents the real world, in which physical active products (PAPs) move in real-time.

A node thus has three components: a virtual image (in the VL), a data memorization and processing structure (in the DS), and a physical infrastructure (in the PL).

Since VL works in accelerated time, adaptation time becomes physically short. On the VL level, lots of VAPs can be used to discover new paths, which helps to keep the number of PAPs circulating on the PL level low. These virtual entities (VAPs) make decisions based on stigmergic principles, which include stochastic decision-making and allow adaptive behavior to deal with unexpected perturbations. VAP travel history is used to update the pheromones (i.e., routing data). From a logical point of view, VAP move within a model of the existing routing network in the PL, which contains PAP. At the PL level, PAP routing decisions are made deterministically, based upon the optimized results of the VL level, called “best efforts”.

The interaction between the two levels is roughly described below (cf. Fig. 2b): after an initialization phase, equivalent to initial learning in VL, PAPs move from source node $ns$ to destination node $nd$.
4. Proposed model

4.1. Notations and variables

Let \( N = \{n_i\} \) be the set of considered nodes \( n_i \) (production resources or transportation system disjunction points in both PL and VL). The variable \( \psi_{uij}^m \) denotes the wth neighbor of \( n_i \) and \( V_{ni} \), the set of the neighbors of \( n_i \): \( V_{ni} = \{\psi_{uij}^m / \exists \text{arc } n_i \rightarrow \psi_{uij}^m\} \). \( w \in \{1...\text{card}(V_{ni})\} \). \( N \) and \( V_{ni} \) thus describe the topology of the FMS, including the transportation system. A possible path between node \( n_i \) and node \( n_j \) is written \( u_{ij} = [n_i \ldots n_j] \) and corresponds to the ranked list of successive nodes to be visited to travel from \( n_i \) to \( n_j \). The set of all possible paths is expressed as

\[
U = \left\{ u_{ij} = [n_i \ldots n_j] \in N^2 \right\}
\]

In our model, seven main variables are used (Table 1). Each variable is described by its localization, production/use characteristics and role.

4.1.1. Variable relevant to the PAP

The first variable, called real measured time \( t_m(u_{ij}) \), is the travel time between a node \( n_i \) and its wth neighbor, as measured by a PAP.

4.1.2. Variables relevant to the DS

The second variable is \( \pi_m(n_j) \), the perturbation flag vector. \( \pi_m(n_j) \) indicates the status of the path \( [n_i \ldots n_j] \) from any node \( n_i \) to any other \( n_j \), where \( \pi_m(n_j) = 1 \) if at least one perturbed arc has been identified on path \( [n_i \ldots n_j] \), otherwise 0. The following notations will also be used later.

- Let \( N_p = \{(n_i, n_j) \in N^2 / \pi_m(n_j) = 1\} \) be the set of perturbed pairs of nodes source-destination.
- Let \( N_{np} = \{(n_i, n_j) \in N^2 / \pi_m(n_j) = 0\} \) be the set of unperturbed pairs of nodes source-destination.

The third variable is \( P_m \), the pheromone matrix. Columns of \( P_m \) correspond to all possible destination nodes \( n_j \) from \( n_i \) and rows contain all existing neighbor nodes \( \psi_{uj}^m \). In analogy to biological systems, this matrix characterizes the pheromone rate on the different arcs \( n_i \rightarrow \psi_{uj}^m \). The value \( P_m(u_{ij}) \) is associated to the preferred path from the current node \( n_i \) to destination node \( n_j \) when choosing the possible intermediate neighbor node \( \psi_{uj}^m \). Consequently, the following expression is true:

\[
\sum_{w} P_m(u_{ij}) = 1, \forall (n_i, n_j) \in N^2
\]

The goal is to find the “best path” from a current node \( n_i \) to a destination node \( n_j \), where the best path is the path on which PAPs spend the least time. Thus, the following fundamental property is applied:

\[
P_m(u_{ij}) > P_m(u_{ij}^2, n_j) \Leftrightarrow u_{ij}^2 \text{ is preferred to } u_{ij} \text{ when moving over the shortest path (shortest time) to destination node } n_j, \text{ where } \{u_{ij}^2, u_{ij}\} \in V_{ni}^2; a \neq b, (a, b) \in \{1...\text{card}(V_{ni})\}^2.
\]

The fourth variable \( r_m(n_j) \) is the time reference vector. \( r_m(n_j) \) is the reference time for a VAP to travel from node \( n_i \) to each \( \psi_{uj}^m \) neighbor.

Fig. 3 gives an example of the use of the above variables. Let a PAP be set on current node \( n_i = n_{h_k} \), coming from the source node \( n_1 \) with the destination node \( n_j = n_{h_0} \).

In this example, the best neighbor is \( n_k \) since:

\[
P_m(n_1, n_k) = .95 > P_m(n_1, n_3, n_8) = .05.
\]

Since \( \pi_m(n_k) = 0 \), the move is authorized. PAP will then travel from \( n_3 \) to \( n_k \), keeping its destination node \( n_8 \) in mind. When arriving at \( n_k \), the time spent \( t_m(n_k) \) is reported to node \( n_3 \) via node \( n_4 \) and is used by the node \( n_3 \) to determine whether or not a perturbation (e.g., bottleneck, slow down, break down) has occurred between \( n_3 \) and \( n_4 \) by comparing \( t_m(n_k) \) with \( t_m(n_4) \).

The fifth variable is \( t_m(n_j) \), the minimum times vector. \( t_m(n_j) \) thus denotes the minimum measured time to move from node \( n_i \) to any possible destination node \( n_j \). This variable is updated every time a lower time is clocked, according to a mobile time window (not described in this paper).

The sixth variable is \( r_m(n_j) \), the simulated time vector. \( r_m(n_j) \) is the time needed for a VAP to go from \( n_i \) to \( n_j \), expressed as \( r_m(n_j) = d_{nj} - d_{ni} \), where \( d_i \) is the arrival date on node \( n_i \). During simulation, this variable will be compared to \( r_m(n_j) \) to increase or decrease the pheromone values.

The seventh variable is \( P_n \), the roadmap vector. Among all possible paths \( u_{ij} = [n_i \ldots n_j] \), \( P_n(n_j) \) is the “best” known path from \( n_i \) to \( n_j \). The term “best” must be understood as a satisfactory path, since optimality can not be proved. Later on in the application, we will show that “best efforts” are often optimized path. \( P_n(n_j) \) is defined as \( P_n(n_j) = \arg \min_{u_{ij} \in U_{ij}} f(u) \),

where \( U_{ij} = \{u_{ij} \in U / u_{ij} = [n_i \ldots n_j]\} \) and \( u_{ij} \) has been explored at least once by a VAP and \( f(u) \) is a metric function that generates the total travel time for each visited path \( u \):

\[
f : U_{ij} \rightarrow \mathbb{R}
\]

\[u \rightarrow f(u)\]

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Notation</th>
<th>Localization</th>
<th>Modified by</th>
<th>Used by</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real measured time</td>
<td>( t_m(u_{ij}) )</td>
<td>PAP</td>
<td>PAP</td>
<td>VL</td>
<td>Travel time measurement</td>
</tr>
<tr>
<td>Perturbation flag</td>
<td>( \pi_m )</td>
<td>DS</td>
<td>VL</td>
<td>PAP</td>
<td>Routing authorization</td>
</tr>
<tr>
<td>Pheromone</td>
<td>( P_m )</td>
<td>DS</td>
<td>VL</td>
<td>PAP, VAP</td>
<td>Neighbor choice</td>
</tr>
<tr>
<td>Time reference</td>
<td>( t_m(n_j) )</td>
<td>DS</td>
<td>VL</td>
<td>VAP</td>
<td>Perturbation detection</td>
</tr>
<tr>
<td>Minimum time</td>
<td>( t_m(n_j) )</td>
<td>DS</td>
<td>VL</td>
<td>VAP</td>
<td>Virtual routing</td>
</tr>
<tr>
<td>Roadmap</td>
<td>( P_n )</td>
<td>DS</td>
<td>VL</td>
<td>VL</td>
<td>Pheromone updating</td>
</tr>
<tr>
<td>History table</td>
<td>( h_{vap}(n, n_j) )</td>
<td>VAP</td>
<td>VAP</td>
<td>VL</td>
<td>Pheromone propagation</td>
</tr>
</tbody>
</table>

Fig. 3 gives an example of the use of the above variables. Let a PAP be set on current node \( n_i = n_{h_k} \), coming from the source node \( n_1 \) with the destination node \( n_j = n_{h_0} \).

In this example, the best neighbor is \( n_k \) since:

\[
P_m(n_1, n_k) = .95 > P_m(n_1, n_3, n_8) = .05.
\]

Since \( \pi_m(n_k) = 0 \), the move is authorized. PAP will then travel from \( n_3 \) to \( n_k \), keeping its destination node \( n_8 \) in mind. When arriving at \( n_k \), the time spent \( t_m(n_k) \) is reported to node \( n_3 \) via node \( n_4 \) and is used by the node \( n_3 \) to determine whether or not a perturbation (e.g., bottleneck, slow down, break down) has occurred between \( n_3 \) and \( n_4 \) by comparing \( t_m(n_k) \) with \( t_m(n_4) \).

The fifth variable is \( t_m(n_j) \), the minimum times vector. \( t_m(n_j) \) thus denotes the minimum measured time to move from node \( n_i \) to any possible destination node \( n_j \). This variable is updated every time a lower time is clocked, according to a mobile time window (not described in this paper).

The sixth variable is \( r_m(n_j) \), the simulated time vector. \( r_m(n_j) \) is the time needed for a VAP to go from \( n_i \) to \( n_j \), expressed as \( r_m(n_j) = d_{nj} - d_{ni} \), where \( d_i \) is the arrival date on node \( n_i \). During simulation, this variable will be compared to \( r_m(n_j) \) to increase or decrease the pheromone values.

The seventh variable is \( P_n \), the roadmap vector. Among all possible paths \( u_{ij} = [n_i \ldots n_j] \), \( P_n(n_j) \) is the “best” known path from \( n_i \) to \( n_j \). The term “best” must be understood as a satisfactory path, called the “best effort”, since optimality can not be proved. Later on in the application, we will show that “best efforts” are often optimized path. \( P_n(n_j) \) is defined as \( P_n(n_j) = \arg \min_{u_{ij} \in U_{ij}} f(u) \),

where \( U_{ij} = \{u_{ij} \in U / u_{ij} = [n_i \ldots n_j]\} \) and \( u_{ij} \) has been explored at least once by a VAP and \( f(u) \) is a metric function that generates the total travel time for each visited path \( u \):

\[
f : U_{ij} \rightarrow \mathbb{R}
\]

\[u \rightarrow f(u)\]
4.1.3. Variable relevant to the VAP

The eighth and last variable is $h_{vap}(n_i, n_j)$, the history table. Each VAP travelling between nodes $n_i$ and $n_j$, updates its history table $h_{vap}(n_i, n_j)$, which is used to store the nodes reached and the corresponding arrival dates:

$$h_{vap}(n_i, n_j) = \left\{ \begin{array}{l} n_1 \quad n_2 \quad n_4 \quad n_5 \quad n_6 \quad n_7 \quad n_8 \\ P_{n_1}(n_1, n_i) \quad P_{n_2}(n_2, n_i) \quad P_{n_4}(n_4, n_i) \quad P_{n_5}(n_5, n_i) \quad P_{n_6}(n_6, n_i) \quad P_{n_7}(n_7, n_i) \quad P_{n_8}(n_8, n_i) \end{array} \right\} = 0.95$$

$$h_{vap}(n_i, n_j) = \left\{ \begin{array}{l} n_1 \quad n_2 \quad n_4 \quad n_5 \quad n_6 \quad n_7 \quad n_8 \\ P_{n_1}(n_1, n_j) \quad P_{n_2}(n_2, n_j) \quad P_{n_4}(n_4, n_j) \quad P_{n_5}(n_5, n_j) \quad P_{n_6}(n_6, n_j) \quad P_{n_7}(n_7, n_j) \quad P_{n_8}(n_8, n_j) \end{array} \right\} = 0.05$$

where $d_i$ is the arrival date at node $n_i$.

Fig. 4 shows an example of a history table for one VAP, which has just arrived at a destination node.

4.2. Model of the routing process

The colored Petri net in Fig. 5 shows all the interactions between PL and VL, which is called the routing process in our model. In this Petri net, five functions can be identified, which are described below.

4.2.1. Function F1/PAP movement in the network

At a given time, a PAP begins to travel from its source node $n_s$ to its destination node $n_d$. The pair $(n_s, n_d)$ is provided by the DAP control system. The current node is designated $n_c$, thus $n_c = n_s$ when the PAP begins to move. The PAP must test whether or not the path $[n_c \ldots n_d]$ is perturbed (transition between places 1 and 2). If the perturbation flag is set to 0 (i.e., $\pi_{nc}(n_d) = 0$), then the path is not perturbed and the PAP continues its trajectory. Otherwise, the path is perturbed and the PAP must wait until an alternative path has been found.

The values of $P_{nc}(v^w_{nc}, n_d)$ are used to choose the next neighbor node $n_n$. This choice is deterministic since the neighbor with the greater $P$ value is always chosen:

$$n_n = \tilde{v}_{nc} \subset V_{nc} / P(v^w_{nc}, n_d) = \max_{w \in \{1 \ldots |V_{nc}|\}} P_{nc}(v^w_{nc}, n_d).$$

During the real move on the arc $n_c \rightarrow n_n$, the real elapsed time $t_{nc}(n_n)$ is measured (place 3). When the PAP arrives at node $n_n$, it moves backwards $t_{nc}(n_n)$ to node $n_c$ (place 4). This time measurement is used later to detect a possible perturbation (e.g., jamming or slowdowns) on the arc $n_c \rightarrow n_n$. A node $m$ becomes the new current node $n_c$, and the PAP continues to travel to the destination node $n_d$, using the above procedure iteratively (place 7). The move is considered finished when the PAP arrives at $n_d$.

4.2.2. Function F2/local perturbation detection

When $t_{nc}(n_n)$ is reported by the PAP (place 8), a perturbation detection analysis is performed on node $n_c$. A local perturbation is assumed to be detected on the arc $n_c \rightarrow n_n$ if $|t_{nc}(n_n) - t_{nc}(n_m)| > \epsilon$, where $\epsilon$ is a fixed threshold parameter. The time reference $t_{nc}(n_m)$ is then updated with the new time $t_{nc}(n_n)$ and the perturbation flag $\pi_{nc}(n_m)$ is set to 1.
4.2.3. Function $F_3$/perturbation impact evaluation

A perturbation on a local arc $n_k \rightarrow n_l$ is propagated using each $\rho_{n_i}$ roadmap vector: a path $\rho_{n_i}(n_j)$ is perturbed if $\exists n_k \rightarrow n_l \in \rho_{n_i}(n_j) / \pi_{n_i}(n_l) = 1$. For each perturbed $\rho_{n_i}(n_j)$, the pair $(n_i, n_j)$ is added to the $N_p$ (set of perturbed pairs of nodes source-destination). For each pair $(n_i, n_j) \in N_p$ only, a new alternative path must be found. This is done via a VAP exploration, which will update the appropriate $P_{n_i}$ matrices.

4.2.4. Function $F_4$/VAP exploration

At each time interval $T$, a VAP is generated on node $n_i$ and assigned the $n_i$ destination node (place 14). On the current node $n_c$, the VAP must choose the next neighbor node $n_n$ (place 15). Since both luck and diversity are important adaptation mechanisms in natural biological systems, the choice of $n_n$ is not deterministic, like PAP choices are, but rather is stochastic. The different $P_{n_c}(n_n)$ are then associated to the probability of reaching the

---

**Fig. 5.** Colored Petri net of the routing process.
final destination \( n_j \) from \( n_c \), via the neighbor \( n^p_{nc} \). However, if \( P_{nc}(v^p_{nc}, n_j) \) is negligible, \( n^p_{nc} \) will be rarely chosen. To ensure diversity, a minimum value \( \alpha \) is introduced. Consequently, the probability of the node \( n^p_{nc} \) being chosen as the next neighbor (denoted \( n_m \)) can be expressed as:

\[
\text{Prob}(n_m = v^p_{nc}) = \max \left( \frac{P_{nc}(v^p_{nc}, n_j)}{\sum_{n_i} P_{nc}(v^p_{nc}, n_i)}, \alpha \right) = \max(P_{nc}(v^p_{nc}, n_j), \alpha) \text{ since } \sum_{n_i} P_{nc}(v^p_{nc}, n_i) = 1
\]

After reaching node \( n_m \), the passage date \( d_{nm} \) is then stored in the history table \( h_{wmp} \) (place 17). Because the move is virtual, this date is calculated by adding the time reference \( t_{x_{nc}(n_m)} \) to the last date in history table. Then the node \( n_m \) becomes the new current node \( n_c \), and the VAP continues to travel towards the destination node \( n_{t} \), iteratively following the same procedure (place 18). Finally, when the VAP reaches the destination node \( n_{t} \), the routing is finished and the history table is reported to node \( n_{t} \) (place 19).

4.2.5. Function FS/“best” alternative path finding

Every time a VAP reaches its destination node \( n_c \), the information contained in its history table \( h_{wmp} \) is forwarded to each visited node to update pheromone matrices. In our approach, this coefficient updating process is based upon a simplified reinforcement law. When the forwarded information reaches node \( n_c \), \( t_{x_{nc}(n_i)} \) and the minimal time \( t_{min}(n_i) \) are compared. If \( t_{x_{nc}(n_i)} \) is less than or equal to the minimal time \( t_{min}(n_i) \) and if \( n_m \) was the neighbor node used to travel from \( n_c \) to \( n_i \), then \( P_{nc}(n_m, n_i) \) is reinforced:

\[
P_{nc}(n_m, n_i) = P_{nc}(n_m, n_i) + r \times \left( 1 - P_{nc}(n_m, n_i) \right), n_m \in V_{nc}
\]

The coefficients \( P_{nc}(n_0, n_j) \) for the destination \( n_i \) of the other neighbors \( n_0 \) are negatively reinforced through a process of normalization:

\[
P_{nc}(n_0, n_j) = P_{nc}(n_0, n_j) - r \times P_{nc}(n_0, n_j), n_0 \in V_{nc} \quad \text{with} \quad n_0 \neq n_m
\]

Although good results have been obtained with this simplified law, we later plan to implement a more advanced law that includes a statistical model of the traffic situation (means and variances computed with trip times), as in AntNet.

The “best” path emerges when all the \( P_{nc} \) coefficients at the attained nodes \( n_c \) are over a fixed threshold \( \Omega \) (0.95 in our study). This state is reached after a transient period (Zone A in Fig. 6). The path is declared “best” after a “check” period (Zone B in Fig. 6). This verification period is indispensable in order to insure that the path found is good enough. This period is over when a fixed number of VAPs, denoted \( v_\gamma \), have reached the destination node \( n_{t} \).

When the pheromone updating has been completed, the roadmap tables \( \rho_{hc}(n_i) \) and the perturbation flags \( \pi_{hp}(n_i) \) are updated. The pair \((n_i, n_j)\) is re-integrated into the \( N_p \) set (place 22).

5. Simulation

5.1. Short description of the simulation tool

The proposed control model is naturally distributed, meaning there is no central memorization and processing system. This property influenced our choice of an agent-based parallel modelling and simulation environment. With Netlogo [22], each modelled entity can be described as an independent agent interacting with its environment. All agents operate in parallel on a grid of patches (i.e., a cellular world), and each agent can read and modify some of the attributes linked to the patches in its proximity. The behavioral rules defined for the agents make it possible to describe agent-environment interaction, which is very important when simulating the stigmergic process.

In the following case study, both the physical and virtual levels are simulated. Fig. 7 shows our simulator’s interface.

5.2. Case study

The proposed approach is applied to an FMS built around a conveyer network based on a one-way conveyer system with divergent transfer gates that allow PAPs to be routed toward the different workstations. This FMS, shown in Fig. 8, can be characterized as follows:

- 15 workstations (nodes in medium grey with a black square inside),
- 24 divergent transfer gates (nodes in medium grey), where a routing choice between two adjacent arc must be made,
- 24 convergent transfer gates (nodes in grey with cross bars).

The FMS is divided into two parts – upper and lower – connected by two arcs \( n_{16} \rightarrow n_{36} \) and \( n_{61} \rightarrow n_{22} \).

For this case study, an online web-based simulation is available at the url: http://www.univ-valenciennes.fr/sp/routing/

6. Scenarios and results

Two scenarios are presented, highlighting our model’s self-adaptive capacities with respect to perturbations. The second scenario is also used to analyze the optimality of the pheromone-based paths. Using an INTEL Xeon with 3 Gigabytes of Ram, the time needed for the different simulations ranged from 1 to 5 s.

6.1. Scenario 1

The first scenario is used to study the impact of a perturbation on an arc that can be by-passed. Nodes \( n_2 \) and \( n_{13} \) are the source and destination nodes, respectively.

6.1.1. Qualitative results

Fig. 9a shows the beginning of the initialization phase. Arrows (near divergent transfer gates) represent the pheromone rate corresponding to the adjacent arcs. Their size is proportional to the pheromone rate (initial values set to 0.5). Fig. 9b shows the result of the initialization phase. The VAPs travel randomly along the different arcs, and the path \( [n_2, n_{12}, n_{13}, n_9, n_8, n_7, n_5, n_3, n_10, n_{11}, n_{12}, n_{13}] \) emerges through \( n_2, n_{12}, n_{23} \) reinforcement. At each divergent transfer gate, the bigger arrows indicate the better arcs. The best path is then traced (in black). Fig. 9c indicates the presence of a perturbation on the arc \( n_9 \rightarrow n_{11} \), affecting the fluidity on this arc (The flow reduction is shown in dark grey). When a PAP (black disc) takes a longer time on this trajectory than expected, \( n_9 \) detects the perturbation and a new VAP exploration is triggered for each pair \((n_9, n_{11})\), where arc \( n_9 \rightarrow n_{11} \) belongs to the roadmap map.
Fig. 7. NetLogo interface.

Fig. 8. FMS description.
The VAPs travelling on the arc $n_9 \rightarrow n_{11}$ perform poorly, thus the appeal of this path decreases. The decreased appeal of this perturbed path consequently increases the appeal of an alternative path $[n_2, n_3, n_4, n_5, n_6, n_7, n_8, n_9, n_{10}, n_{11}, n_{12}, n_{13}]$ (Fig. 9d), even though this path was originally not the best. This dynamic response is a classic display of the natural route reconfiguration capacity of the stigmergic approach.

6.1.2. Quantitative results

Fig. 10 shows the evolution of some of the $P_{ni}(t)$ coefficients. Please note the change that occurs after the perturbation at date $d = 200$ (in simulated time). Fig. 11 shows more precisely the evolution of the coefficient $P_{n_9(n_{12}, n_{13})}$ for a perturbation occurring at $d = 902$ and for its resolution at $d = 1472$. After the perturbation has been resolved, the coefficient $P_{n_9(n_{12}, n_{13})}$ again converges on 1.

6.2. Scenario 2

This second scenario is designed to compare our “best effort” results with the optimal paths. Nodes $n_2$ and $n_{25}$ are the source and destination nodes, respectively. In the event of a perturbation on the arc $n_{20} \rightarrow n_{21}$, the optimal path becomes:

$\{n_2, n_3, n_4, n_5, n_6, n_7, n_8, n_9, n_{11}, n_{12}, n_{14}, n_{15}, n_{16}, n_{36}, n_{37}, n_{38}, n_{40}, n_{41}, n_{43}, n_{44}, n_{46}, n_{47}, n_{48}, n_{49}, n_{51}, n_{52}, n_{53}, n_{54}, n_{56}, n_{57}, n_{59}, n_{60}, n_{61}, n_{62}, n_{23}, n_{24}, n_{25}\}$.

Fig. 12 outlines the relationship between the success percentages for finding this optimal path, according to the $\gamma$ parameter (see Section 4.2.5). If $1 < \gamma \leq 140$, an increase in $\gamma$ induces a very significant increase in the success percentage; however, if $140 < \gamma$, a slow convergence towards the optimal path is observed. More importantly, the best efforts converge toward this optimal path in proportion with $\gamma$. Other simulations not described in this paper also highlighted this fact.

7. Synthesis

The results described above are consistent with those found by Di Caro using the AntNet algorithm. Di Caro and Dorigo [15] showed that AntNet out-performed most routing algorithms (e.g., Bellman-Ford, OSPF) for heavy perturbations. In our FMS context, the PAPs are able to determine the best path from the departure node to the destination node without any centralized control. In addition, they are able to overcome perturbations by seeking out new paths that bypass the perturbation but still lead to the desired destination. These capacities indicate that our approach offers a good level of adaptability and robustness in the face of environmental fluctuations.

Five parameters ($r, \alpha, \Omega, \kappa$ and $\gamma$) were introduced for fine-tuning the model. The $\Omega$ and $\gamma$ parameters represent an exploration effort and are dependant on the size of the network. As is true traditionally in ACO applications, and more generally in reinforcement learning applications, the parameter choices directly influence two main algorithm characteristics: optimality and scalability.

Consequently, we verified our approach’s performance in terms of these two characteristics:

- Optimality—we conducted experiments to verify our approach’s ability to find the optimal routing solution. For the network
shown in Fig. 8, we compared our results with the optimal paths obtained with the classic Dijkstra algorithm in a static environment without perturbations. Our approach found the optimal path for 99.75% of the all unperturbed source-destination pairs.

- Scalability—as the number of nodes increases in a network, the number of VAPs needed to find the shortest paths increases, and consequently, the convergence time also increases. One way to solve this problem would be to use topological information to divide a network into zones and then to use intra- and inter-zone routing.

More details on these two characteristics (e.g., respect of Bellman’s optimality criterion, parameter fine-tuning) can be found in references [14,15,20,21].

8. Elements for a real implementation

Given the promising results of the simulation, we decided to test our model on a real implementation. The experimental support for this implementation is the flexible assembly cell of the AIP-PRIMECA Center at the University of Valenciennes (Fig. 13). Its topology corresponds to the upper area of the FMS presented in Fig. 8.

Three types of equipment were used for the implementation:

- Conveyer system, which is a Montrac monorail transport system (Montech, 2008 [23]) using self-propelled electrical shuttles (Fig. 14) to transport products/materials on tracks;
- Instrumented Node (Figs. 15 and 18):
  - a GC, or gate controller, which works to oversee the transfer gate and to help avoid collisions (only used on the routing node);