Decentralized motion planning and scheduling of AGVs in FMS

Guillaume Demesure, Michael Defoort, Abdelghani Bekrar, Damien Trentesaux, Mohamed Djemai

Decentralized Motion Planning and Scheduling of AGVs in FMS

Guillaume Demesure, Michael Defoort, Abdelghani Bekrar, *Member, IEEE*, Damien Trentesaux, *Member, IEEE* and Mohamed Djemaï, *Senior member, IEEE*

***Abstract*—In this paper, decentralized motion planning and scheduling of automated guided vehicles (AGVs) in a flexible manufacturing system (FMS) is proposed. A motion planner is combined with a scheduler allowing each AGV to update its destination resource during navigation in order to complete the transported product. The proposed strategy is based on two steps. The first step consists in planning a presumed trajectory to avoid collision conflicts previously detected by a central supervisor, enabling more appropriate decentralized scheduling by AGVs. The presumed trajectories, which represent the intentions of AGVs, are then exchanged with neighboring AGVs. The second step uses the presumed trajectories of neighbors to compute a collision–free trajectory according to the priority policy. Numerical and experimental results are provided to show the pertinence and the feasibility of the proposed strategy.**

***Index Terms*—Motion planning, Scheduling, Automated Guided Vehicles, Flexible Manufacturing Systems.**

I. Introduction

Technological advances in mechatronics, computer science, and Information and Communication Technologies (ICT) facilitates the use of AGV-based fleets in manufacturing applications while limiting their costs [1]. AGV fleet-based factories can now cover increasingly stricter industrial requirements [2], looking for reactivity in the short term and market adaptability in the long term. In this paper, a "factory of the future"-oriented environment is assumed, where AGVs navigate freely in the FMS transporting products from resource to resource.

In this context, scheduling and product transportation are both well-known problems in the literature (Fig. 1). The scheduling problem for each AGV concerns finding the "best" resource on which the operation of the transported product will be performed [3], [4] while respecting its production specifications provided at a higher level dealing with Manufacturing Operation and Resource Management (M.O.R.M.). These specifications are composed of a due date and a set of possible resources, which translates the flexibility of the FMS.

This work has been supported by International Campus on Safety and Intermodality in Transportation, the European Community, the Regional Delegation for research and Technology, the Haut-de-France Region, the Ministry of Higher Education and Research and the National Center for Scientific Research under the ARCIR SUCRe, BI-CFINES, PHC NUSANTARA and ELSTAT/VUMOPE projects.

G. Demesure is with the Lorraine University, CRAN, CNRS UMR 7039, Nancy, France (e-mail: guillaume.demesure@univ-lorraine.fr), since September 2017. He was previously with the University of Valenciennes and Hainaut-Cambrésis, UVHC, LAMIH UMR CNRS 8201, France.

M. Defoort, A. Bekrar, D. Trentesaux and M. Djemaï are with the University of Valenciennes and Hainaut-Cambrésis, UVHC, LAMIH UMR CNRS 8201, France (e-mail: {michael.defoort, abdelghani.bekrar, damien.trentesaux, mohamed.djemai}@univ-valenciennes.fr).

In terms of product transportation, two similar problems are primarily considered in the literature (see Fig. 1). On the one hand, routing problems that consist in determining a collision-free predefined route to reach the desired destination [5] and on the other hand, motion planning that consists in generating a collision–free trajectory from the initial configuration to the desired destination [6]. From our perspective, the mentioned motion planning is not local (i.e. generation of motions based on the route and the determined behavior) [7]. Thus, the main difference between these transportation problems is the existence of predefined routes for the routing problem while, in the case of motion planning, the AGVs can navigate without following any imposed routes. Integrating the scheduling problem into transportation problems is a major challenge in AGV-based FMS. This integration allows reactive updating of schedules during navigation in order to tackle any possible conflicts as they occur.

In the literature, two types of integrated scheduling and routing can be identified depending on routing flexibility [8] which relates to the number of alternative routes an AGV can choose to reach its destination [9]. The first considers reduced routing flexibility with one or two alternative routes between workstations (resources) [10], [11]. Although it reduces the complexity of the routing problem, the full potential of the AGVs is not exploited since routing flexibility is reduced. The second type consists in using a mesh topology where a set of collision-free routes (or nodes) throughout the network must be computed to solve the routing problem [12], [13], [14]. Despite greater flexibility, the AGVs need to stop at nodes to choose their next route.

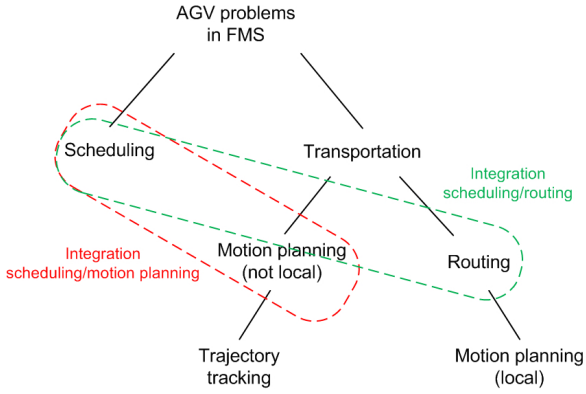


Fig. 1: Scheduling and transportation problems of AGVs in FMS.

Although the integration of scheduling and routing has been excessively studied over the past few decades, few studies propose the integration of scheduling and motion planning where AGVs do not follow an imposed route. Indeed, the scheduling is computed by a central system and the motion planning is decentralized [15]. The integration of scheduling and motion planning depends on the chosen planning methods, which have been studied excessively over the past few decades, especially in the robotic field. Different methods have been proposed such as roadmaps, cell decomposition, potential fields and optimization techniques, among others [6], [16]. Optimization-based motion planners [17], [18] seem to be interesting tools, especially when manufacturing performances in terms of completion time, for instance, are required. By considering integration with scheduling, [1] proposed an artificial potential fields approach. Besides its low computational time, this heuristic method has issues with local minima and equilibrium attraction. In [19], integration is achieved in a decentralized way. However, although decentralized architectures are more suitable to satisfy flexibility requirements, they lead to poorer global performance due to a lack of information [20]. In [21], the AGVs have to stop to synchronize their schedules.

This paper focuses on product transportation by AGVs in a FMS (see Section II) while satisfying AGV behavior and manufacturing constraints. In order to solve the limitations identified, the contribution is the following. A motion planner is combined with a scheduler to select the destination for the ongoing operation, in a decentralized way and without stopping to synchronize the schedule. Integrated scheduling and motion planning (instead of routing) is therefore considered. The global architecture, presented in Section III, includes a supervisor to solve the different conflicts between AGVs. The second objective of this supervisor is to detect possible collisions between AGVs to provide them with anticipation, leading to more appropriate scheduling and better global performances. Due to decentralization, the AGV trajectories are updated gradually over time in two steps (see Section IV). The first step generates a presumed trajectory representing the intention of the AGV. The second step involves exchanges of intentions with neighboring AGVs to plan the final trajectory which ensures decentralized collision avoidance. Numerical and experimental results are provided in Section V to prove the effectiveness of the proposed approach. Moreover, comparisons with [19] and obstacle avoidance routines are investigated.

II. Problem Setup

Let us consider an AGV-based FMS with N_c resources (workstation, machine...), N_l products, and N_i AGVs (denoted hereinafter by "agent" for convenience) (N_c, N_l, $N_i \in \mathbb{N}^*$). The notations (adapted from [4]) are given in Tab. I.

Hereinafter, the following assumptions are made. Each agent transports only one product from one resource to another until the product is completed without following any imposed routes. Each product, specified by the M.O.R.M. level, is only transported by one agent. All resources, represented here as machines performing operations on products, are motionless.

TABLE I: Notation for variables and parameters

$\mathcal{A}$:	Set of AGVs (agents) $\mathcal{A} = \{1, ...N_i\}$
$\mathcal{R}$:	Set of resources (machines) $\mathcal{R} = \{1, ..., N_c\}$
$\mathcal{J}$:	Set of products $\mathcal{J} = \{1, ..., N_l\}$
$\mathcal{O}_l$:	Set of sequential operations $\mathcal{O}_l = \{o_{nl}\}$ on product l
o_{nl}:	The n^{th} operation on product $l \in \mathcal{J}$
$\mathcal{R}_{nl}$:	Set of resources which can perform operation n
ct_{nlb}:	Completion time of operation n at ressource $b \in \mathcal{R}_{nl}$
opt_{nl}:	Processing time of operation n of product l
odd_{nl}:	Due date of operation n of product l
p_b:	Position $(x_b, y_b)^T$ of resource $b \in \mathcal{R}$
T_c:	Update period of the motion planner
τ_k:	Update time $\tau_k = \tau_0 + k \cdot T_c, k \in \mathbb{N}$, τ_0 is initial time
w_b:	Actual waiting time in the queue at resource b
wp_{ib}:	Additional waiting time due to agents moving towards resource b
sch_i:	Binary variable set to 1 if agent i may schedule at the update time τ_k, 0 otherwise
$q^*_{ib}(t, \tau_k)$:	Initial guess trajectory of agent i at update time τ_k towards resource $b \in \mathcal{R}_{nl}$
$\widetilde{q}_{ib}(t, \tau_k)$:	Presumed trajectory of agent i at update time τ_k towards resource b
$q_i(t, \tau_k)$:	Final planned trajectory of agent i at update time τ_k
p_i:	Actual position of agent i (i.e. at update time τ_k)
v_i:	Actual velocity of agent i
cr_{ik}:	Resource chosen by agent i at update time τ_k
$T_{i,init}$:	Initial time when agent i starts from a resource
TT^*_{ibk}:	Transportation time corresponding to $q^*_{ib}(t, \tau_k)$
$\widetilde{TT}_{ik}$:	Transportation time corresponding to $\widetilde{q}_i(t, \tau_k)$
TT_{ik}:	Transportation time corresponding to $q_i(t, \tau_k)$
$T_{ik,fin}$:	Final time when agent i plan to arrive at a resource
R_{com}:	Broadcasting range of all agents $i \in \mathcal{A}$
d_{safe}:	Safety distance between agents to avoid collision
$\mathcal{N}_i$:	Actual set of neighbours of agent i
IP_i:	Actual individual performance of agent i
$\mathcal{HP}_i$:	Actual priority set of agent i
$\mathcal{CF}_i$:	Actual conflict set of agent i, $\mathcal{CF}_i = \{\mathcal{C}_{ijb}\}$
$\mathcal{C}_{ijb}$:	Conflict of agent i with agent j when ressource b is chosen

Once an operation has been completed by a resource, the agents leave via the output area of this resource. Conversely, the agents arrive via the input area of a resource, which corresponds to its working storage (waiting queue). The input and output areas of resources are in different positions to prevent collisions between incoming and outgoing agents. The resource queuing capacities are assumed to be infinite and resource breakdowns are not studied in this paper. When an agent $i \in \mathcal{A}$ is available, the M.O.R.M. level assigns a product $l \in \mathcal{J}$ to this agent. The M.O.R.M. level provides the production specifications (for which the calculation is not addressed in this paper) for each ongoing operation $o_{nl} \in \mathcal{O}_l$ on assigned products. These specifications are not modified over the considered time interval and include a set of resources $\mathcal{R}_{nl}$ on which the ongoing operation n can be performed as well as a due date odd_{nl} by which this operation must be completed. The M.O.R.M level also provides the waiting time w_b of each resource $b \in \mathcal{R}$, which corresponds to the time period between when the agent arrives at the resource and when the resource starts processing the operation. All agents are identical in terms of physical behavior, device, and size. Figure 2 illustrates some variables and assumptions. The agent dynamics are described as $\forall i \in \mathcal{A}$

$$\dot{p}_i(t) = v_i(t), \quad q_i \in \mathbb{R}^2, v_i \in \mathbb{R}^2 \tag{1}$$

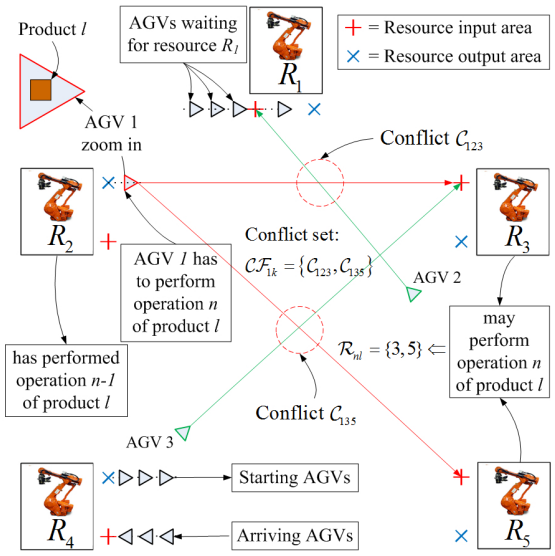


Fig. 2: Variables and assumptions considered for the proposed problem in an illustrative FMS.

The velocity of each agent i is bounded as follows:

$$\|v_i(t)\| \leq v_{max}, \quad v_{max} \in \mathbb{R} \tag{2}$$

Each agent i knows its initial position $p_i(T_{i,init}) = p_{i,init}$ and the final position of each possible resource p_b ($b \in \mathcal{R}_{nl}$). The position p_b depends on the operation o_{nl}. Similarly to [19], a coordinated motion planning problem is considered, where besides avoiding collisions, agents need to cooperate and make decisions in order to reach their chosen resources $b \in \mathcal{R}_{nl}$. For each agent i, the actual set of neighbors $\mathcal{N}_i$, at update time τ_k, is defined as:

$$\mathcal{N}_i = \{j \in \mathcal{A}, \|p_i - p_j\| < R_{com}\} \tag{3}$$

In this paper, only the completion of one operation by an AGV is considered. For each AGV, the objective is to select the best resource in terms of completion time and to compute the optimal trajectory taking into account agent behavior (1)-(2), collision avoidance and manufacturing constraints according to the M.O.R.M. level. For each trajectory $q_i^*(t, \tau_k)$, $\widetilde{q}_i(t, \tau_k)$ or $q_i(t, \tau_k)$, the first argument denotes time, and the second one specifies at which update time τ_k the trajectory is planned.

III. Proposed Architecture for an AGV-based Flexible Manufacturing System

To solve the introduced problem, an architecture (shown in Fig. 3) is proposed and combines a navigation scheme for each agent and a supervisor. The M.O.R.M. level provides manufacturing information at each update for both the supervisor and the agents.

Here, the supervisor algorithm and the navigation scheme are applied gradually over time with respect to the update period T_c, which corresponds to a fixed time interval between two updates. At each update time τ_k, the supervisor applies its algorithm to provide each AGV with the required information, including any conflicts detected. The navigation algorithm then starts in a decentralized way, and each AGV updates its trajectory iteratively and in parallel.

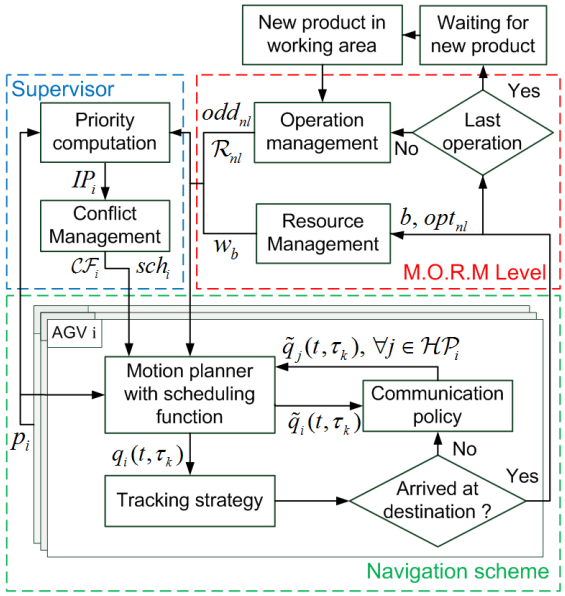


Fig. 3: Navigation scheme and its interactions with the M.O.R.M. level and the supervision layer.

A. Supervisor design

To apply its navigation scheme at update time τ_k, each AGV needs the information relating to conflict solving. This information comes from the supervisor and its algorithm is divided into two parts: priority computation and conflict management. The priority is computed according to the individual performance IP_i of each agent $i \in \mathcal{A}$, defined as follows:

$$IP_i = \begin{cases} Ind_i & \text{, if } Ind_i \in [0\ 1] \\ 1 & \text{, otherwise} \end{cases} \tag{4}$$

where $Ind_i = \dfrac{\tau_k - T_{i,init} + \frac{\|p_i - p_b\|}{v_{max}}}{odd_{nl} - T_{i,init} - opt_{nl} - w_b}$. If $Ind_i \in [0\ 1]$, IP_i characterizes the ratio between the estimated travel time (assuming a straight path) and the maximum transportation time permitted by the M.O.R.M specifications. If $Ind_i \notin [0\ 1]$, the specifications cannot be satisfied for resource b, chosen after the last update. In this case, the agent has the highest priority, i.e. $IP_i = 1$, allowing it to schedule its product as soon as possible. Using this index, one can define the priority policy, where for an agent i, the set of agents with higher priority is:

$$\mathcal{HP}_i = \{j \in \mathcal{N}_i, IP_i \leq IP_j\} \tag{5}$$

This priority policy changes dynamically, allowing coordination between agents according to the actual situation in the FMS. Indeed, an agent may have a higher priority according to IP_i. These values change according to the AGV fleet decisions made after the last updates (e.g. when an agent i reduces its velocity to avoid collisions or when the waiting

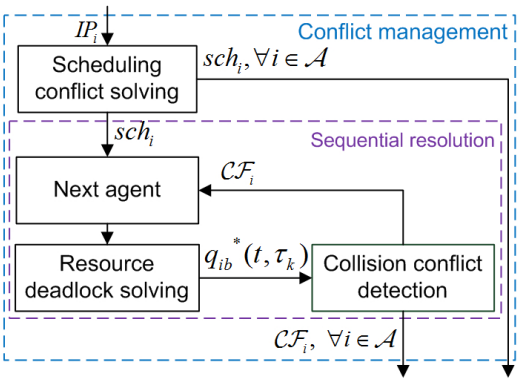


Fig. 4: Conflict management algorithm by the supervisor.

time for resource b increases). The priority policy is then used by the conflict management block. Its role is to solve scheduling conflicts and resource deadlocks as well as to detect the collision conflicts, as shown in Fig. 4. When the algorithm ends, the supervisor provides agents with the binary variable sch_i and the set of detected conflicts $\mathcal{CF}_i$.

Scheduling conflict: This conflict occurs when two or more neighboring agents schedule their product (select the destination) at the same update time. Indeed, if this occurs, the collision conflicts cannot be easily solved in a decentralized way due to the unknown resource chosen by these agents. In this case, the supervisor checks the number of updates since the last scheduling to decide which agent will select its resource at the actual update time. If several agents have the same number of updates, the priority set (5) is used to make a decision. This decision is given to each agent by means of the variable sch_i where $sch_i = 1$ means that the scheduling is allowed for agent i at update time τ_k, otherwise $sch_i = 0$. Therefore, each agent does not schedule at each update time τ_k and must wait a certain number of updates before the next scheduling.

Sequential resolution: In order to detect collision conflicts, the supervisor algorithm requires a sequential resolution depending on the priority mechanism. The agents are sorted according to the index IPi, from the highest to the lowest values, and so it is possible to start (resp. end) with the agent with the highest (resp. lowest) priority. The resolution thus allows each agent to only consider others with higher priority for which conflicts have been solved. When $sch_i = 1$, the resolution is achieved for each possible resource $b \in \mathcal{R}_{nl}$. A presumed resource $\widetilde{cr}_{ik}$ is chosen by computing a heuristic estimation of the completion time according to the conflicts detected along the straight trajectory to each resource. This choice of resource is used to solve resource deadlocks for subsequent agents with lower priority.

Resource deadlock solving: This conflict arises when some neighboring agents head towards the same resource and may, therefore, arrive at the same resource at the same time. This block allows the initial guess trajectories used for collision conflict detection to be generated. At the update time τ_k, the initial guess trajectory is the best straight trajectory, i.e. the one minimizing the transportation time TT^*_{ibk}. It satisfies the velocity constraint $\|\dot{q}^*_{ib}(t, \tau_k)\| \leq v_{max}$ and the following terminal constraints:

$$q^*_{ib}(\tau_k, \tau_k) = p_i \quad (6)$$
$$\dot{q}^*_{ib}(\tau_k, \tau_k) = v_i \quad (7)$$
$$q^*_{ib}(\tau_k + TT^*_{ibk}, \tau_k) = p_b \quad (8)$$
$$\dot{q}^*_{ib}(\tau_k + TT^*_{ibk}, \tau_k) = 0 \quad (9)$$

In the case of a resource deadlock, the following constraints, imposing an arrival order at the resource, are added to the transportation time of the initial guess:

$$TT^*_{ibk} > TT^*_{jbk}, \quad \forall j \in \{\mathcal{HP}_i : b = \widetilde{cr}_{jk}\} \quad (10)$$

Collision conflict detection: The initial guess trajectories are used to detect collision conflicts of agent i with all other agents with a higher priority, i.e. $\forall j \in \mathcal{A}, IP_i \leq IP_j$. Detection is carried out even when agents do not communicate with each other so they can pre-empt collisions. If the distance between the initial guess trajectories is lower than the safety distance d_{safe}, a new conflict $\mathcal{C}_{ijb} = (j, b, I_{ij}, cd_{ij})$ is created where j is the agent to avoid when resource b is chosen. I_{ij} is the conflict time interval and cd_{ij} is the degree of the conflict, which represents the number of agents with higher priority involved in the conflict created. When all conflicts are created, they are gathered in the conflict set $\mathcal{CF}_i$, given to agent i.

B. Navigation scheme

The objective of the navigation scheme, for each agent, is to compute the best resource $b \in \mathcal{R}_{nl}$ and to generate a collision-free trajectory to the best resource. The best resource b is chosen by minimizing the completion time ct_{nlb} comprising the transportation time TT_{ik} to the resource, the resource waiting time w_b and the processing time opt_{nl}. Thus, minimizing the completion time means that the objective is to complete the operation at the earliest date, which is a common production objective [4]. Other cost functions will be investigated in further studies (see for instance [5]). The navigation scheme of each agent combines several processes:

- The *motion planner with scheduling function* computes both the optimal collision-free trajectory at each update time and the best resource in terms of completion time. The planner strategy is divided into two steps. The first step uses global information, provided by the supervisor, to compute a presumed trajectory representing the agent's intentions. The second step uses these intentions to compute the final planned trajectory by taking local collision constraints into account.
- The *tracking strategy*, not detailed in this paper, allows short-term objectives to be fulfilled despite the presence of external disturbances and inherent discrepancies between the model and the real AGV [17].
- The *communication policy* allows the required presumed trajectories to be transmitted and received between agents. Each agent only receives the presumed trajectories of neighbors with a higher priority.

When an agent arrives at a resource, the waiting time for this resource is updated.

IV. Motion planner with scheduling function

The proposed motion planning problem is expressed as a constrained optimization one. A decentralized approach, allowing the decomposition of the overall motion planning problem into simpler optimization subproblems is proposed for parallel resolution. Each simpler problem is implemented in each agent i. At each update time τ_k, the agents sequentially compute an optimal planned trajectory satisfying agent behavior (1)–(2), manufacturing constraints described by the product specifications, and the collision avoidance constraint. Agents only handle partial information, limiting the computational complexity of their subproblems.

A. Global algorithm

As shown in Fig. 5, the proposed algorithm is divided into two steps where a trajectory is planned at each step.

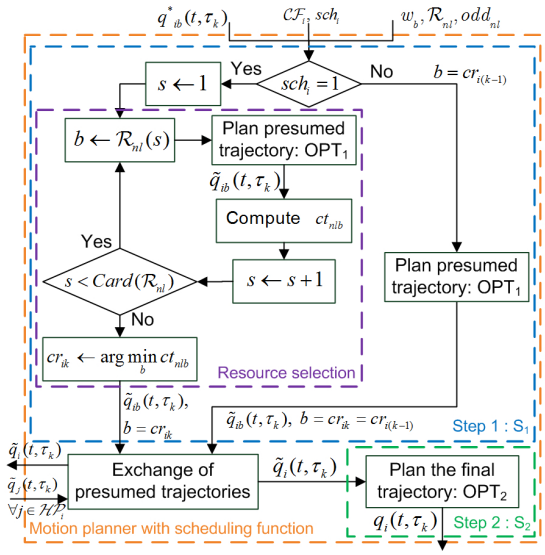


Fig. 5: Strategy of motion planner with resource selector.

Step 1: This step ensures the generation of a presumed trajectory by avoiding conflicts detected by the supervisor. These conflicts are represented by partial initial guess trajectories of other agents with higher priorities. In this step, scheduling may be applied by some agents i according to the variable sch_i provided by the supervisor. When scheduling is allowed for agent i, a presumed trajectory $\tilde{q}_{ib}(t,\tau_k)$ is computed for each resource $b \in \mathcal{R}_{nl}$. The presumed trajectories are computed using an optimization algorithm, denoted by OPT$_1$ (see Fig. 5). For each presumed trajectory, a completion time ct_{nlb} is computed as:

$$ct_{nlb} = \tau_k + \widetilde{TT}_{ik} + w_b + wp_{ib} + opt_{nl}, b \in \mathcal{R}_{nl} \qquad (11)$$

where $\widetilde{TT}_{ik}$ is the transportation time of the presumed trajectory, w_b is the waiting time in the queue at resource b, and opt_{nlb} is the operation processing time. Variable wp_{ib} is a fictive waiting time representing the additional time agent i must wait for other agents $j \in \mathcal{HP}_i$ moving towards the same resource and it is estimated as follows:

$$wp_{ib} = \sum_{j \in \mathcal{HP}_i : cr_{jk} = b} opt_{n_j l_j} \qquad (12)$$

where $opt_{n_j l_j}$ is the processing time of neighbor j transporting its product l_j for operation n_j. Neighbors $j \in \mathcal{HP}_i$ do not schedule product at the same update time τ_k since $sch_i = 1 \Rightarrow sch_j = 0 \Rightarrow cr_{jk} = cr_{j(k-1)}$. The resource is chosen by minimizing this completion time i.e. $cr_{ik} = \arg\min_b ct_{nlb}$. If scheduling is not allowed for agent i ($sch_i = 0$), only one presumed trajectory to the resource selected at the last update is computed. However, conflict avoidance only considers partial trajectories, so some collisions may still occur. Once step 1 is completed, all the agents exchange their presumed trajectories with their neighbors.

Step 2: This step allows the presumed trajectory to be adjusted to ensure collision avoidance by means of an optimization problem, denoted by OPT$_2$ (see Fig. 5). Using the presumed trajectories of neighbors $j \in \mathcal{HP}_i$, a planned collision-free trajectory for each agent is computed. Decentralized collision avoidance is guaranteed using an additional constraint ensuring a small deviation from the presumed trajectory.

B. Optimization problems

The two optimization problems introduced OPT$_1$ and OPT$_2$ are described here for an update time τ_k. These two optimization problems are continuous since they deal with continuous variables and parameters.

OPT$_1$: For each agent $i \in \mathcal{A}$, let us consider the following optimization problem which consists in determining the presumed trajectory $\tilde{q}_{ib}(t,\tau_k)$ at the update time τ_k to a resource b:

$$\min_{\tilde{q}_{ib}(t,\tau_k), \widetilde{TT}_{ik}} \int_{\tau_k}^{\tau_k + \widetilde{TT}_{ik}} \|\tilde{q}_{ib}(t,\tau_k) - q^*_{ib}(t,\tau_k)\| \, dt \qquad (13)$$

The following constraints must be satisfied, $\forall t \in [\tau_k, \tau_k + \widetilde{TT}_{ik}]$ and $\forall b \in \mathcal{R}_{nl}$:

$$\left\|\dot{\tilde{q}}_{ib}(t,\tau_k)\right\| \leq v_{max} \qquad (14)$$

$$\tilde{q}_{ib}(\tau_k,\tau_k) = p_i \qquad (15)$$

$$\dot{\tilde{q}}_{ib}(\tau_k,\tau_k) = v_i \qquad (16)$$

$$\tilde{q}_{ib}(\widetilde{T}_{i,fin},\tau_k) = p_b \qquad (17)$$

$$\dot{\tilde{q}}_{ib}(\widetilde{T}_{i,fin},\tau_k) = 0 \qquad (18)$$

$$ct_{nlb} \leq odd_{nl} \qquad (19)$$

$$\|\tilde{q}_{ib}(t,\tau_k) - q^*_{jc}(t,\tau_k)\| > cd_{ij} \cdot d_{safe} \quad \forall (j,b,I_{ij},cd_{ij}) \in \mathcal{CF}_i, \ \forall t \in I_{ij} \qquad (20)$$

The objective function (13) allows the presumed trajectory to be as close as possible to the initial guess trajectory. Constraint (14) satisfies the agent velocity bound. The terminal conditions are described by constraints (15)-(18). Constraint (19) allows the production specifications to be considered in terms of operation due date. Conflict avoidance is performed using constraint (20) where $c = \tilde{cr}_{jk}$ is the estimated chosen

resource of agent j. The conflict degree cd_{ij} allows the size of the conflict area to be tuned according to the number of agents involved. Thus, when two or more agents avoid a collision conflict in the same area, they avoid this area differently due to their respective degrees of conflict.

OPT$_2$: After solving **OPT**$_1$, for each agent $i \in \mathcal{A}$, a second optimization problem consisting in planning the final collision–free trajectory $q_i(t, \tau_k)$ at the update time τ_k, is defined:

$$\min_{q_i(t,\tau_k), T_{ik,fin}} TT_{ik} \tag{21}$$

Since the resource was chosen at step 1, the objective is only to minimize the transportation time TT_{ik} in order to arrive at the earliest date while respecting constraints similar to (14)–(19). Conflict constraint (20) is replaced by the two following coupled constraints, ensuring local collision avoidance, $\forall t \in [\tau_k, \tau_k + TT_{ik}]$ and $\forall j \in \mathcal{HP}_i$:

$$\|q_i(t,\tau_k) - \widetilde{q}_i(t,\tau_k)\| \leq \xi + f(t) \tag{22}$$

$$\|q_i(t,\tau_k) - \widetilde{q}_j(t,\tau_k)\| > d_{safe} + \xi \tag{23}$$

Constraint (22) ensures that each agent i stays within the vicinity ξ of their presumed trajectory. A function $f(t)$ is added to relax the constraint such as $f(t) = 0, \forall t \in [\tau_k, \tau_k + T_p]$ and $f(t) > 0, \forall t > \tau_k + T_P$. If this constraint is not relaxed, the optimization solver may fail to find a solution for OPT$_2$ due to collision avoidance. With constraint (23), each agent avoids neighbors $j \in \mathcal{HP}_i$ according to both the safety distance d_{safe} and the neighborhood ξ. The two coupled constraints (22) and (23) ensure that collisions are avoided over $t \in [\tau_k, \tau_k + T_p]$, since trajectories are updated by agents at each τ_k and $T_p > T_c = \tau_{k+1} - \tau_k$.

V. Numerical and Experimental Results

In this section, the numerical results for the proposed approach are presented with different scenarios. The first scenario is devoted to a simple application of the approach and shows the superiority of the scheme in terms of trajectory generation. The second scenario includes obstacle avoidance. The third scenario is more complex in order to get closer to manufacturing workshops and evaluates the global performance of the strategy. The last scenario provides proof of concept with robots. For each scenario, the update period was set to $T_c = 0.5s$ and the maximum allowed speed of agents to $v_{max} = 1.5m/s$. All agents tracked their planned trajectory perfectly, which was carried out using a robust sliding mode controller [22].

In order to solve the optimization problems OPT$_1$ and OPT$_2$, spline curves were adopted in this paper to specify the trajectories. This transforms the trajectory generation problem into a parameter optimization one. As motion planning problems are NP-hard [23], a particle swarm optimization (PSO) algorithm [18] was applied to compute the B-spline coefficients.

A. Navigation-oriented scenario

The results of the proposed strategy were compared with [19] using a scenario with 4 AGVs and 3 resources. The initial parameters of the AGVs and resources can be found in [19]. Figure 6 shows the trajectories of all the agents in each scheme. The trajectory of agent 3 is a straight line (initial guess) due to the priority policy. Agent 4, which starts moving towards resource 2, selects resource 3 at $\tau_k = 2s$ due to its lower priority. Even if agents 1 and 4 are moving towards the same resource R_3, agent 4 arrives at its destination before a collision occurs.

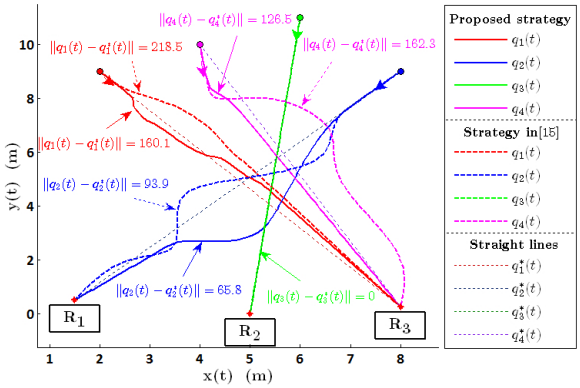


Fig. 6: Comparison of agent trajectories

From Fig. 6, one can see that the proposed strategy provides better agent trajectories than [19] since they are closer to the straight lines $q_i^*(t)$ (using the $\|\cdot\|$ norm). This is mainly due to supervisor conflict detection, which anticipates collisions so the agents are ready to avoid a collision before communicating with others. This example illustrates the capability of the supervisor to be proactive in agent motion planning and assist in selecting the appropriate resource.

B. Obstacle avoidance routine

In real FMS, the environment may include several obstacles. To avoid these obstacles, the following constraint was included in optimization problem **OPT**$_1$ for each obstacle o centered on the position p_o with a radius r_o:

$$\|\widetilde{q}_{ib}(t,\tau_k) - p_o\| > r_o + d_{safe}/2 + \xi \tag{24}$$

Obstacle avoidance is ensured since the agent must stay within the vicinity ξ for the second step with constraint (22). Figure 7 shows that obstacles are avoided with this additional constraint. The radius of each obstacle is augmented by the size of robots.

Although the agents avoid obstacles in a decentralized way, their completion performances may decline as an agent may choose a destination with lots of obstacles along the trajectory. These obstacles may be detected by the supervisor. However, coupling between the detection and the generation of initial guess trajectories is still an open problem due to the need for more than one straight line trajectory per resource.

C. Performance-oriented scenario

Here, the capability of the proposed strategy to deal with more complex situations, closer to real industrial systems, is evaluated. The FMS configuration (i.e. resource positions,

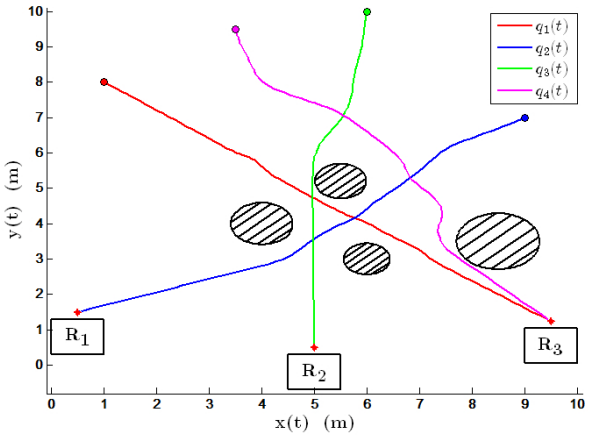


Fig. 7: Scenario with several obstacles

navigation area...) is the same as the one indicated in Fig. 2. Using this configuration, different cases were tested 10 times to assess the performance according to the number of agents in the working area. For each case, the initial specifications (e.g. processing time opt_{nl}, due date odd_{nl}, ...) were generated randomly.

TABLE II: Resource specifications of scenario 2

Resource	Agent starting position	Agent ending position
R_1	$[16 \quad 30]^T$	$[13 \quad 30]^T$
R_2	$[3 \quad 19]^T$	$[3 \quad 16]^T$
R_3	$[27 \quad 19]^T$	$[27 \quad 16]^T$
R_4	$[3 \quad 6]^T$	$[3 \quad 3]^T$
R_5	$[27 \quad 7]^T$	$[27 \quad 4]^T$

The global performances were evaluated with the variable $P\% = 1/N_i \cdot \sum_{i=1}^{N_i} P_i\%$ where N_i is the number of agents and $P_i\%$ is the local agent performance defined as:

$$P_i\% = 100 \cdot \frac{odd_{nl} - ct_{nlb}}{odd_{nl} - \overline{ct}_{nlb}} \tag{25}$$

$\overline{ct}_{nlb}$ refers to the initial completion time and is computed when the agent starts from the resource as $\overline{ct}_{nlb} = T_{i,init} + \min_b(\overline{TT}_{ib}) + opt_{nl}$. $\overline{TT}_{ib}$ corresponds to the minimum transportation time to resource b. The agent performance is the best (resp. worst) when $P_i\% = 100 \Leftrightarrow \overline{ct}_{nlb} = ct_{nlb}$ (resp. $P_i\% = 0 \Leftrightarrow ct_{nlb} = odd_{nl}$). Table III shows the performances with different numbers of agents. The maximum (resp. minimum) corresponds to the best (resp. worst) performance among the 10 tests.

TABLE III: Evaluation of global performances

Number of agents	Global performance $P\%$			Computational cost	
	Min	Mean	Max	Step 1	Step 2
5	96.5	97.9	99.3	1.49 s (3)	1.12 s (2)
10	86.5	91.3	95.4	1.82 s (3)	1.18 s (2)
15	76.8	81.6	85.3	2.56 s (5)	1.30 s (3)
20	71.9	79.6	81.9	2.64 s (6)	1.38 s (4)
25	58.6	64.2	69.8	2.66 s (6)	1.41 s (4)

The right part of Table III shows the maximum computational costs for the 10 tests. The numbers in parentheses are the number of conflicts (step 1) and the number of neighboring agents (step 2). These numbers tend to reach a maximum value (when the agent number is greater than 20) due to agent coordination. Indeed, at step 1 the agents deviate from straight line trajectories according to their degree of conflict thus reducing the number of conflicts for the next updates. At step 2, since the degree of conflict is different from step 1, the number of neighbors is limited. Step 1 is computationally more expensive than step 2 due to the scheduling function which requires a trajectory to be generated per possible resource. Moreover, the computational times are greater than the update period $T_c = 0.5$. Nevertheless, they result from Matlab programming and can be greatly reduced using C++ programming (see [24]).

With 5 agents, the global performance was close to the best one. When the number of agents increased, the performance worsened due to the greater number of conflicts and the longer resource waiting times. An example with 13 agents for this scenario is shown in the following video[1]. This was compared with [19] (where $P\% = 69.24$), described in this video, to show the superiority of the proposed strategy (where $P\% = 77.80$). This superiority comes mainly from the supervisor as it provides anticipation for the scheduling. Moreover, the agents do not stop during navigation, even when they update their schedule.

D. Proof of concept

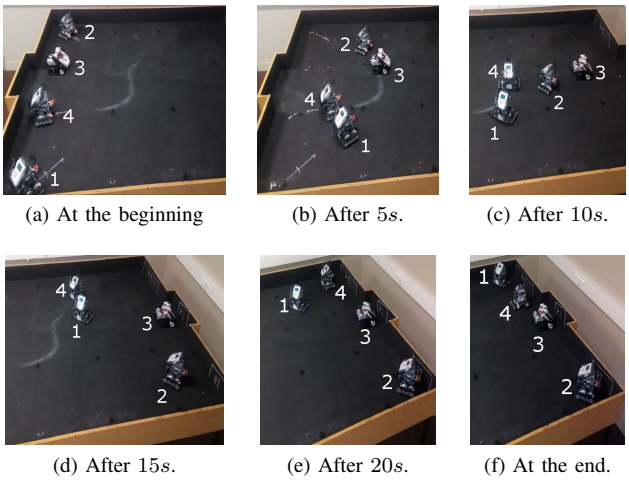
To highlight the feasibility of the collision–free trajectories generated by each agent, a team of LEGO Mindstorms robots with tracks instead of wheels (Fig 8) was used to demonstrate the proof of concept. An experimental scenario was designed with four robots, used to represent the agents (AGVs) and three virtual resources. The robot communication range and the safety distance were set to $R_{com} = 0.4m$ and $d_{safe} = 0.2m$, respectively. The maximum velocity of the robots allowed was $v_{max} = 0.15m/s$. The results for this scenario are given in Tab. IV where the differences between the simulation and the real cases are highlighted.

TABLE IV: Results of the proof of concept

	Final time (simulation)	Final time (real)	Completion time	Operation due date
AGV 1	$25.44s$	$25.92s$	$31.6s$	$76.1s$
AGV 2	$17.76s$	$18.07s$	$22.38s$	$30.8s$
AGV 3	$13.14s$	$13.33s$	$18.33s$	$23.8s$
AGV 4	$21.02s$	$21.68s$	$26.36s$	60.2

The planned trajectories, allowing AGVs to reach the optimal resources, were computed for each robot. Figure 8 shows six snapshots of our experiments. One can see in Fig. 8 (a) and (f) the initial and final configurations of the robots. For this proof of concept, the overall explanations are provided in the following video[2].

[1]Video available online: https://youtu.be/XL4yfJcy9KA
[2]Video available online: https://youtu.be/jlMfQso7ye4



(a) At the beginning (b) After $5s$. (c) After $10s$.

(d) After $15s$. (e) After $20s$. (f) At the end.

Fig. 8: Four Lego Mindstorms robots moving towards resources while avoiding collisions between each other.

This proof of concept illustrates the ability of the motion planner to plan feasible trajectories. Indeed, there is little difference between the simulation results and the real results and no collisions occurred during the navigation of the robots. Moreover, the reactivity of the proposed approach was proven since agent 4 updated its resource during navigation.

VI. Conclusion

A new navigation approach for AGVs in FMS has been proposed. The motion planner proposed includes a scheduling function minimizing the operation completion time. Each AGV finds its most appropriate resource to perform an ongoing operation while generating a collision free-trajectory in two steps. The first step generates a presumed trajectory according to the global conflict information while the second step uses these intentions locally to compute a collision-free trajectory. A supervisor was designed to assist agents with conflict solving and provide them with global conflict information. The numerical results and the proof of concept have highlighted the pertinence of the proposed strategy. The results have shown the reactivity induced by the scheduling aspect of the motion planner and the proactivity of the supervisor. In further studies, the focus will be on the completion of all operations where other aspects will be addressed such as obstacles, energy, and other objective criteria (e.g. earliness/tardiness, mean flow time).

References

[1] K. Ueda, "Emergent synthesis approaches to biological manufacturing systems," in Dig. Ets. Tech. Springer, 2007, pp. 25–34.
[2] O. Cardin, F. Ounnar, A. Thomas, and D. Trentesaux, "Future industrial systems: Best practices of the intelligent manufacturing services systems (IMS^2) french research group," IEEE Trans. on Indus. Inf., vol. 13, no. 2, pp. 704–713, 2017.
[3] W. Du, Y. Tang, S. Leung, L. Tong, A. V. Vasilakos, and F. Qian, "Robust order scheduling in the fashion industry: A multi-objective optimization approach," IEEE Trans. on Indus. Inf., p. accepted, 2017.
[4] D. Trentesaux, C. Pach, A. Bekrar, Y. Sallez, T. Berger, T. Bonte, P. Leitão, and J. Barbosa, "Benchmarking flexible job-shop scheduling and control systems," Cont. Eng. Pract., vol. 21, no. 9, pp. 1204–1225, 2013.
[5] H. Fazlollahtabar and M. Saidi-Mehrabad, "Methodologies to optimize automated guided vehicle scheduling and routing problems: a review study," Journal of Int. & Rob. Syst., vol. 77, no. 3-4, p. 525, 2015.
[6] E. Masehian and D. Sedighizadeh, "Classic and heuristic approaches in robot motion planning-a chronological review," World Academy of Sci., Eng. and Tech., vol. 23, pp. 101–106, 2007.
[7] S. Tapase, T. Shirshikar, and A. Pachghare, "A review on autonomous car," Int. Adv. Journal. in Sci., Eng. and Tech., vol. 6, no. 4, pp. 95–100, 2017.
[8] P. Sharma and A. Jain, "Effect of routing flexibility and sequencing rules on performance of stochastic flexible job shop manufacturing system with setup times: Simulation approach," Proc. of the Inst. of Mech. Eng., Part B: Journal of Eng. Man., vol. 231, no. 2, pp. 329–345, 2017.
[9] H. A. ElMaraghy, "Flexible and reconfigurable manufacturing systems paradigms," Int. journal of flex. man. syst., vol. 17, no. 4, pp. 261–276, 2005.
[10] S. C. Srivastava, A. K. Choudhary, S. Kumar, and M. Tiwari, "Development of an intelligent agent-based AGV controller for a flexible manufacturing system," Int. Journal of Adv. Man. Tech., vol. 36, no. 7-8, pp. 780–797, 2008.
[11] T. Nishi, Y. Tanaka, and M. Inuiguchi, "Petri net decomposition approach for the simultaneous optimization of task assignment and routing with automated guided vehicles," in IEEE Int. Conf. on Aut. Sci. and Eng. IEEE, 2008, pp. 175–180.
[12] U. A. Umar, M. Ariffin, N. Ismail, and S. Tang, "Hybrid multiobjective genetic algorithms for integrated dynamic scheduling and routing of jobs and automated-guided vehicle (agv) in flexible manufacturing systems (fms) environment," Int. Journal of Adv. Man. Tech., vol. 81, no. 9-12, pp. 2123–2141, 2015.
[13] M. Saidi-Mehrabad, S. Dehnavi-Arani, F. Evazabadian, and V. Mahmoodian, "An ant colony algorithm for solving the new integrated model of job shop scheduling and conflict-free routing of agvs," Comp. & Ind. Eng., vol. 86, pp. 2–13, 2015.
[14] H. Ghasemzadeh, E. Behrangi, and M. A. Azgomi, "Conflict-free scheduling and routing of AGVs in mesh topologies," Rob. and Aut. Syst., vol. 57, no. 6, pp. 738–748, 2009.
[15] H. Martínez-Barberá and D. Herrero-Pérez, "Autonomous navigation of an automated guided vehicle in industrial environments," Rob. and Comp.-Integ. Man., vol. 26, no. 4, pp. 296–311, 2010.
[16] I. A. Sucan, M. Moll, and L. E. Kavraki, "Open motion planning library," Rob. & Aut. Mag., vol. 19, no. 4, pp. 72–82, 2012.
[17] M. Defoort, A. Kokosy, T. Floquet, W. Perruquetti, and J. Palos, "Motion planning for cooperative unicycle-type mobile robots with limited sensing ranges: A receding horizon approach," Rob. and Aut. Syst., vol. 57, no. 11, pp. 1094–1106, 2009.
[18] E. Masehian and D. Sedighizadeh, "A multi-objective PSO-based algorithm for robot path planning," in IEEE Int. Conf. on Indus. Tech. (ICIT). IEEE, 2010, pp. 465–470.
[19] G. Demesure, M. Defoort, A. Bekrar, D. Trentesaux, and M. Djemaï, "Navigation scheme with priority-based scheduling of mobile agents: Application to AGV-based flexible manufacturing system," Journal of Int. & Rob. Syst., vol. 82, no. 3, pp. 495–512, 2016.
[20] S. R. González, I. Mondragón, G. Zambrano, W. Hernandez, and H. Montaña, "Manufacturing control architecture for fms with agv: A state-of-the-art," in Adv. in Aut. & Rob. Res. in Latin America. Springer, 2017, pp. 157–172.
[21] R. Alami, S. Fleury, M. Herrb, F. Ingrand, and F. Robert, "Multi-robot cooperation in the martha project," Rob. & Aut. Mag., vol. 5, no. 1, pp. 36–47, 1998.
[22] M. Defoort and M. Djemaï, "A lyapunov-based design of a modified super-twisting algorithm for the heisenberg system," IMA Journal of Math. Cont. and Inform., vol. 30, no. 2, pp. 185–204, 2012.
[23] J. Canny, "The complexity of robot motion planning," MIT press, 1988.
[24] S. B. Aruoba and J. Fernández-Villaverde, "A comparison of programming languages in economics," NBER Working Paper Series, vol. 3, pp. 1–19, 2014.

Guillaume Demesure is an associate professor at ENSTIB (High School of Wood Sciences and Timber Engineering) in the University of Lorraine, since September 2017. He received his PhD in automatic control from the University of Valenciennes in 2016. He is also member of the CRAN/CNRS–University of Lorraine, Nancy, France. His PhD researches focus on the navigation of mobile robots/AGVs in flexible manufacturing systems. His current research interests are in the area of production activity control of flexible manufacturing systems, with applications to wood industry with or without the presence of AGVs.

Michael Defoort is an Associate Professor at University of Valenciennes, since 2009. He is with Laboratory of Industrial and Human Automation control, Mechanical engineering and Computer Science (LAMIH), CNRS UMR 8201. He received the PhD degree in 2007 from Ecole Centrale de Lille. From 2007 to 2008, he was a Research Fellow with the Department of System Design Engineering, Keio University, Japan. His research interests include nonlinear control with applications to power systems, multi-agent systems and vehicles.

Abdelghani Bekrar received his engineering degree in computer science (1999) from INI (Algerian National high school), Master degree from ECN (French National high school), and his Ph.D. (2007) from the University of Technology of Troyes (UTT), France. He is currently an associate Professor at the University of Valenciennes. His research interests include Metaheuristic design and implementation, hard optimization for engineer applications and supply chain management. He has published more than 20 papers and conference presentations and he is involved in several research projects.

Damien Trentesaux is Professor at the University of Valenciennes and Hainaut-Cambresis, in France. His areas of interest' include intelligent scheduling and control of discrete event systems (manufacturing, transport, logistics, and services) using multi-agent, bio-inspired and holonic models. Prof. Trentesaux has supervised 12 PhD thesis and is author and co-author of more than 100 peer reviewed publications in journals, books, chapters of books and conference proceedings. He is a member of the IEEE IES Technical Committee on Industrial Agents and of the IFAC TC 5.1 on Manufacturing Plant Control.

Mohamed Djemaï is full professor at the University of Valenciennes, France since 2008. He is with LAMIH–Laboratory of Industrial and Human Automation, Mechanics and Computer Science, CNRS. UMR 8201. He is member of IFAC TC 2.1 control system and IFAC TC. 1.3 on Discrete Event and Hybrid Systems. Prof. Djemaï was visiting professor at Northumbria University (2010-2013). His research interests are mainly related to nonlinear control, observation, and fault detection theory including hybrid system control, sliding mode, and variable structure systems, with applications to power systems, robot and vehicles.