Pollux: a dynamic hybrid control architecture for flexible job shop systems

José-Fernando Jimenez Gordillo, Abdelghani Bekrar, Gabriel Zambrano Rey, Damien Trentesaux, Paulo Leitão

International Journal of Production Research 55/15 (2017) 4229-4247

Pollux: a dynamic hybrid control architecture for flexible job shop systems

Jose-Fernando Jimenez[a,b]* , Abdelghani Bekrar[a], Gabriel Zambrano-Rey[b], Damien Trentesaux[a] and Paulo Leitão[c,d]

[a]*LAMIH, UMR CNRS 8201, University of Valenciennes and Hainaut-Cambrésis, UVHC, Valenciennes, France;* [b]*Industrial Engineering Department Pontificia, Universidad Javeriana, Bogotá, Colombia;* [c]*Department of Electrical Engineering, Polytechnic Institute of Bragança, Campus Santa Apolónia, Bragança, Portugal;* [d]*LIACC – Artificial Intelligence and Computer Science Laboratory, Porto, Portugal*

**Corresponding author. Email: j-jimenez@javeriana.edu.co*

Abstact

Nowadays, manufacturing control systems can respond more effectively to exigent market requirements and real-time demands. Indeed, they take advantage of changing their structural and behavioural arrangements to tailor the control solution from a diverse set of feasible configurations. However, the challenge of this approach is to determine efficient mechanisms that dynamically optimise the configuration between different architectures. This paper presents a dynamic hybrid control architecture that integrates a switching mechanism to control changes at both structural and behavioural level. The switching mechanism is based on a genetic algorithm and strives to find the most suitable operating mode of the architecture with regard to optimality and reactivity. The proposed approach was tested in a real flexible job shop to demonstrate the applicability and efficiency of including an optimisation algorithm in the switching process of a dynamic hybrid control architecture.

Keywords : flexible job shop; hybrid production system; dynamic scheduling; switching mechanism

1. Introduction

Industries expect to deploy manufacturing control systems that provide optimal and reactive solutions to meet exigent market demands (Gunasekaran and Ngai 2012). For this, it is fundamental to define a good control system architecture that responds to the optimality and reactivity required. In this paper, *control system architectures* refer to the characterisation of a control system that defines the constituent elements, structural composition and operational behaviour and manage the functioning of a control system. In practice, in manufacturing systems, this corresponds to the arrangement of manufacturing execution systems (MES) that, interacting with an enterprise resource system (ERP), governs manufacturing operations such as planning, manufacturing management, scheduling, storage and transportation. Originally and currently in some cases, control system architectures are implemented over hierarchical architectures. This approach divides the control of manufacturing operations into organised master/slave decisional entities according to dependent sub-problems. The advantage is that manufacturing operations can be globally optimised. Normally, it has a centralised control that enables the deployment of optimal decision-making to cascade through the entire hierarchical dependencies. The disadvantages are mostly related to the computational limitations of the lower dependencies, delayed communications due to the increased number of middle dependencies, lack of the adapting the control strategy, and lack of reactivity capabilities (Dilts, Boyd, and Whorms 1991; Saharidis, Dallery, and Karaesmen 2006). However, more recently, control system architectures have migrated to heterarchical structures, where the control problem is solved through the emergent behaviour of cooperative decisional entities. The advantage is that they respond promptly to perturbations and mitigate the risk of system breakdowns. Nevertheless, difficulties in predicting global performance, the inability to provide global optimality and the need for robust communication protocols are the main disadvantages of this approach (Lee, Lv, and Hong 2013; Borangiu et al. 2015). In short, both approaches satisfy part of industries requirements. Still, neither hierarchical nor heterarchical architectures simultaneously fulfil optimality and reactivity requirements.

To resolve this issue, researchers have introduced a new type of arrangements for controls systems architectures, named *Hybrid Control Architectures* (HCA). These coupled arrangements aim to benefit from the advantages of hierarchical and heterarchical architectures, avoiding the associated limitations. The main feature is that they contain simultaneously hierarchical and heterarchical relationships (Trentesaux 2009). In this sense, decisional entities can still react autonomously to unexpected events while they are coordinated for optimal performance. However, between hierarchical

DOI : 10.1080/00207543.2016.1218087

and heterarchical relationships, it is difficult to synchronise the level of autonomy for each entity (Bongaerts et al. 2000; Cardin et al. 2016). Nonetheless, although some contributions have been made to solve this challenge (Pach et al. 2014; Zambrano Rey 2014) further exploration in this direction is required.

According to the literature, HCA can be classified into two categories: static-HCA (S-HCA) and dynamic-HCA (D-HCA). For S-HCA, the control system architecture starts with an initial operating mode that maintains the same hierarchical/heterarchical coupling throughout production execution. An *operating mode* is a specific parameterisation (definitions of all parameters) that defines a specific setting of the control system architecture (Jimenez et al. 2015). For D-HCA, the control system architecture also starts with an initial operating mode, but it contains a mechanism that switches from one operating mode to another (i.e. periodical and/or event-based) during execution. *Switching* occurs, changing the operating mode of an HCA, either to respond to an unexpected event or to improve system performance.

In this paper, we explore the potential benefits of D-HCA with regard to the two needs introduced: optimality and reactivity. These benefits are interesting and bring a level of novelty to manufacturing control architectures. For this reason, Pollux as a D-HCA that contains a switching mechanism to steer the operating modes of a control system architecture and to ensure an adequate configuration according to real-time needs is proposed. The application in this paper concerns production scheduling in a real flexible job shop system (type of manufacturing system). Flexible job shop systems, where an operation of a job (also named products) can be processed by a subset of machines from a manufacturing system, feature several types of flexibility such as machine, process, routing, material handling and operation flexibility. These systems are interesting as they permit the reconfiguration of the scheduling settings to respond to unexpected events (Zambrano Rey et al. 2014). This problem is also known as dynamic scheduling in the flexible job shop problem (FJSP) in other domains such as the operational research, manufacturing engineering and operational management community.

The paper is organised as follows: Section 2 provides a literature review of D-HCA used for dynamic scheduling of flexible job shop problems. Section 3 details the structure, behaviour and dynamic characteristics of the proposed control system. Section 4 introduces the case study in a flexible job shop and describes the implementation of the proposed D-HCA. In Section 5, the experiments and results obtained in a real environment are presented. Finally, Section 6 summarises the conclusions and highlights future prospects.

2. Literature review

This section provides a review of the literature with regard to D-HCA that features a switching mechanism responsible for reconfiguring the control system architecture in response to an unexpected event, specifically for the flexible job shop systems. It focuses on understanding the contribution of each approach in relation to the dynamic characteristics and the degree of optimisation achieved by the switching mechanism.

2.1 *Definitions*

This subsection details some of the main definitions of a D-HCA relating to the inherent characteristics and switching process in the control system architecture. These definitions are organised into three categories, related to their structural characteristics, behavioural characteristics and the dynamic characteristics of the studied control system architecture.

2.1.1 *Structural characteristics*

Constituent elements of the D-HCA refer to the virtual components used in the architecture that define the characteristics and attributes of each decisional entity. These entities, generally agents or holons, are included in the control system to solve an assigned problem in production operations. In practice, these are created to mirror, in a virtual environment, the physical components of the flexible job shop system (e.g. tools, machines, conveyors, automated guided vehicle), as well as to encapsulate information about the component (perceptions, actions, monitoring, etc.).

The interaction refers to the communication protocol and agreements between the constituent elements. These interactions determine the type of control system architecture as they characterise the relationship between the elements. For example, if the interaction is a master–slave relationship, it is said that both elements hold a hierarchical relationship. On the contrary, if two elements interact to organise a coordinated behaviour, it holds a heterarchical structure. In a nutshell, it is assumed that the entire set of interactions between all decisional entities leads to a specific architecture (i.e. hierarchical, heterarchical or hybrid architecture).

2.1.2 *Behavioural characteristics*

Predictive schedule generation, which is performed before operation execution (offline), is the process that controls the global actions of a particular problem. Specifically, it refers to any predictive decision-making technique used by entities to create the production schedule. Predictive techniques can be based on operational research methods that manage the actions to optimise global performance (e.g. MILP, Genetic algorithms, Grasp).

Reactive control policy is the functioning rules that the constituent elements follow throughout the execution of operations (online). Specifically, it refers to the actions or behaviour of constituent elements, which is guided by a reactive decision-making technique during normal or disruptive events. Reactive techniques can be based on heuristics methods to guide the elements' behaviour (e.g. priority rules, dispatching rules, distributed arrival time control).

2.1.3 *Dynamic characteristics*

Switching mechanism in D-HCA is defined as a mechanism that evaluates the system performance, activating a change in the system configuration, balancing the operation functioning and finding a custom-built operating mode for the control system architecture.

Switching points are the moments in time when the control system makes the switches between operating modes in a D-HCA.

Switching type refers to the kind of changes that occur in the architecture at a switching point. These changes can be structural (i.e. hierarchical, hybrid or heterarchical arrangements), behavioural (i.e. predictive or reactive decision-making methods) or both structural and behavioural.

Switching purpose is the goal or objective pursued by the D-HCA when it triggers the switching process in the control system architecture.

Operating mode is a specific parameterisation (definitions of all parameters), which defines a specific setting of control system architecture. It defines the structural composition and operational behaviour level of the control system architecture.

Degree of optimality is the intended level of optimisation in the switching mechanism process. Based on the degree of optimality defined by Baker (1998) for scheduling algorithms, we defined the degree of optimality of a switching mechanism can be: *optimal* (e.g. mathematical programming, combinatory optimisation), *near-optimal* (e.g. approximation algorithms, probabilistic algorithms), *towards-optimal* (e.g. neural networks, evolutionary algorithms, swarm algorithms) and *heuristic* (e.g. artificial intelligence, priority rules, iterative simulation). An additional degree of optimality, named *reaction transition*, is included in order to classify the approaches with no specific optimality process (lack of optimality).

2.2 *Switching mechanism in D-HCA*

In this subsection, the papers reviewed are examined and positioned according to their general D-HCA characteristics (structure, behaviour and dynamism). They focus on solving the dynamic schedule of a flexible job shop system and present a well-defined hybrid control architecture for managing the execution of manufacturing operations. Still, these approaches differ in many aspects, such as the type of switching, the switching purpose, the functioning of the switching mechanism and the degree of optimality expected with the switching mechanism (refer to definitions in subsection 2.1). Table 1 presents the reviewed papers and characterises each D-HCA according to the structural, behavioural and dynamic characteristics.

From the literature review, three main aspects were identified concerning the switching of the D-HCA studied. Regarding the first aspect, switching is either structural or behavioural characteristics. At structural characteristics, these approaches switch the structure formation between decisional entities (i.e. hierarchical, heterarchical) to modify the control scope in the interconnected graph. A typical example of such architectures is ADACOR (Leitão and Restivo 2006). This architecture changes the structure by propagating pheromones that activate or deactivate the connection and interaction of a supervisor holon. At behavioural characteristics, these approaches switch functions, objectives or decision-making processes of decisional entities (i.e. priority rules or first available machine method) to react and/or to evolve to new, better configurations. As an example of behavioural-based switching, Holvoet, Weyns, and Valckenaers (2009) propose D-MAS where, according to an ant exploration technique that anticipates the future production conditions, it adjusts holon behaviour to reach the expected result. In spite of these contributions, few approaches have addressed switching at structural and behavioural characteristic at the same time (Barbosa et al. 2015). The switching mechanism in this approach, named ADACOR2, performs a micro (behavioural) and/or a macro (structural) self-configuration according to the level of degradation caused by the perturbation. While a structural switch is triggered

Table 1. Dynamic hybrid control architectures with a switching mechanism.

	Structural characteristics		Behavioural characteristics		Dynamic characteristics				
Reference	Constituent element	Interaction of elements	Predictive schedule generation	Reactive control policy	Switching type	Switching purpose	Operating mode	Switching mechanism	Degree of optimality
Barbosa et al. (2015)	Holon driven (ADACOR)	Cooperation (agent UML protocol)	Optimised schedules (not specified)	Client–server intentions	Structure Behaviour	Evolution via Bio-inspired dissemination	Multiple modes (evolvable characteristics)	Reasoning modules define how to evolve according to pheromone technique and learning	Towards optimal
Borangui et al. (2015)	Holon driven (PROSA)	Negotiation (FIPA-ACL)	Production rules and constraint programming	Heuristic allocation	Structure	Reaction to improve strategy	Three different modes	The predictive commands are deleted and allocation is performed reactively (negotiated or not)	Reaction transition (no optimality)
Holvoet, Weyns, and Valckenaers (2009)	Agent driven (PROSA)	Coordination (direct communication protocol)	Not considered	Virtual ant agents (explore intention feasibility)	Behaviour	Evolution via improving ant objective	Multiple modes (active or inactive state per local entity)	Own behaviour change module (forward and backwards)	Towards optimal
Jimenez et al. (2015)	Decisional entities	Cooperation (direct communication)	Genetic algorithm	Product guided by potential fields	Structure Behaviour	Evolution to improve strategy	Multiple modes (evolvable characteristics)	Evaluation of expected operating modes for improvement through an iterative process	Towards optimal
Leitao and Restivo (2006)	Holon driven (ADACOR)	Cooperation (Agent UML)	Optimised schedules (not specified)	Guided by decision-making comportment	Structure	Reaction via Bio-inspired pheromone dissemination	Two different modes: Stationary and transitory state	Holons increase the autonomy factor and reject proposed allocation	Towards optimal
Novas et al. (2013)	Agent driven (PROSA)	Collaboration (XML communication protocol)	Constraint programming model	Virtual ant agents (explorative and intention)	Behaviour	Evolution via bio-inspired exploration ants	Multiple modes (two objectives per local entity)	Each agent adapts its behaviour comparing schedule and exploratory solutions	Towards optimal
Pach et al. (2014)	Entity driven	Iterative bidding (potential fields)	Integer linear programming	Product guided by potential fields	Structure	Reaction to accomplish entity objective	Multiple modes (two behaviours per local entity)	Intelligent products change to reactive potential fields causing behaviour and posterior structure switch	Reaction transition (no optimality)
Raileanu et al. (2012)	Holon driven (PROSA)	Negotiation (FIPA-ACL)	Offline planner (not specified)	Active holon entity schedules product trajectory	Structure	Reaction to improve strategy	Three different modes	The predictive commands are deleted and the allocation is performed reactively (negotiated or not)	Reaction transition (no optimality)

Valckenaers et al. (2007)	Agent driven (PROSA)	Coordination (direct communication)	Offline schedule (not specified)	Virtual ant agents (explorative and intention)	Behaviour	Evolution via bio-inspired exploration ants	Multiple modes (evolvable characteristics)	Holons constantly change the intention according to the exploration ants	Towards optimal
Zambrano Rey et al. (2014)	Part-driven entities	Cooperation (direct communication)	Genetic algorithm	Local distributed arrival time schedules	Behaviour	Reaction to improve with post-optimal heuristic	Multiple modes (local schedules per entity)	Parts recalculate local schedules to improve behaviour	Heuristic

when the predictive schedule deviates, a behavioural switch is triggered when a decisional entity cannot accomplish its own objective or it discovers an opportunity to evolve.

Regarding the second aspect, the review shows two reasons for switching: to react and to evolve. On one side, in the react approach, the switching mechanism is triggered by an event-based method to adapt the execution to an unpredicted event. Generally, it seeks to provide continuity in the execution, and to mitigate possible breakdowns. It is therefore necessary to constantly monitor the system to detect perturbations. Once a perturbation is detected, the system evaluates the level of degradation to activate the switch (most suitable operating mode). As an example of a react approach, Borangiu et al. (2015) propose a control system that detects a perturbation which, depending on the type and frequency of the perturbation, switches from a hierarchised to a negotiated or non-negotiated heterarchical architecture. On the other side, in the evolvable approach, the switching mechanism is activated by a periodic (or mixed periodic/event-driven) technique responsible for reacting and constantly improving the architecture. The switching mechanism constantly monitors the system, but also performs a predictive/prescriptive analysis of the system dynamics towards an efficiency improvement. For this purpose, it might contain a technique that anticipates system behaviour or a performance evaluator that identifies forthcoming out-of-control situations. As an example of an evolvable approach, Jimenez et al. (2015) examine the inclusion of a control performance indicator to evaluate system efficiency over time, and consequently have a criterion for featuring continuous evolvable switching.

Finally, a switching technique with an associated degree of optimality was identified in the literature review. While some studies discuss the use of bio-inspired mechanisms to perform switching, others present proactive mechanisms to improve the current operating mode. For the bio-inspired mechanism, two examples are ADACOR and ADACOR 2 architectures (Leitão and Restivo 2006; Barbosa et al. 2015). The authors use pheromone dissemination of stigmergy technology to guide an adequate elements formation after switching. Other examples of bio-inspired mechanisms, namely PROSA and D-MAS, use virtual ants to explore the intentions of future behaviour and guide decisional entities through the decision-making process (Valckenaers et al. 2007; Holvoet, Weyns, and Valckenaers 2009; Novas et al. 2013). The degree of optimality of bio-inspired approaches is towards-optimal as, while they do not guarantee an optimal solution, they attempt to obtain the best result in a specific metric. A different approach is the proactive mechanism, where the architectures execute normally until an event occurs (perturbation or evolution opportunity). At this moment, the architecture switches either to a predefined configuration according to the type of event (Raileanu et al. 2012; Pach et al. 2014; Borangiu et al. 2015), or it executes a post-event process to adjust structure and/behaviour of the decisional entities (Böhnlein, Schweiger, and Tuma 2011; Herrera, Thomas, and Parada 2014; Zambrano Rey et al. 2014). With predefined configurations, the system has a pre-determined architecture with production strategies and control objectives tailored for using according to a specific event. However, these approaches lack a degree of optimality because they focus on reacting to ensure execution continuity according to a pre-determined catalogue. Conversely, some approaches include post-event processes to respond to perturbations or opportunities for evolution. From the literature review, the degree of optimality of these approaches reaches heuristic as it employs a method for immediate goals. Still, it is notable that the degree of optimality is bounded by the reduced time available to respond to the event.

From our point of view, the approaches reviewed have certain characteristics that contribute to an efficient switching mechanism. However, the papers reviewed do not fully exploit the benefits of reconfiguring a switching mechanism. In fact, three main limitations for developing an efficient switching mechanism in D-HCA have been identified in the literature. Firstly, in relation to the interaction between the switching mechanism and the control system architecture, the switching mechanism of these approaches is integrated into the control system architecture. This condition makes it difficult to compare different switching approaches as they are applied to different control system architectures. For this reason, a reference architecture with dynamic possibilities that supports structural/behavioural flexibility, and serves as a framework tool to implement and compare different switching approaches, is required. A second major limitation is that the possible operating modes evaluated by the switching mechanisms are bound by changes to only a few alternative configurations (pre-determined, pre-evaluated or adjustable). In this sense, a switching mechanism is unable to explore diverse configurations with different outcomes to reach a satisfactory control solution. To improve this, a switching mechanism might offer a broader scope of control solutions (from diverse operating modes) resulting in better manufacturing control systems. Finally, in relation to the optimisation attained by the switching mechanism, the contributions have not explored different switching mechanism techniques. Efforts have been concentrated on stigmergy or proactive-configuration mechanisms. However, these techniques have not explored the entire scope of optimisation of the switching mechanism. In fact, concerning the degree of optimality and the search for optimal switching, only towards-optimal, heuristics and reaction transition techniques have been addressed. It suggests that the lack of use of better degrees of optimality resides in the high computational cost associated with dynamic schedules. However, other techniques with different degrees of optimality need to be explored for solve efficient and reactivity switching requirements.

In these circumstances, a general architecture of the proposed approach is introduced in the next section. The architecture of *Pollux* features different operating modes and includes a switching mechanism that efficiently manages the changing of operating modes according to manufacturing needs. This approach is an extension of the work proposed in Jimenez et al. (2015, 2016). Although this architecture is developed to be used in any control system (i.e. logistics, robotics, health care), the architecture of *Pollux* will be applied in this paper to schedule dynamically a flexible job shop system.

3. Pollux architecture

3.1 *Structural and behavioural characteristics*

The proposed architecture is an assemblage of *decisional entities* that, like atomic components, participate in a collective process to accomplish their own and joint objectives (single or multiple). As in other architectures, a decisional entity in *Pollux* is a virtual unit that features autonomous, sociable, cooperative, reactive and proactive behaviour with the capability of achieving the stated objective(s). From the literature review, the closest studies related to this paper are the ORCA (Pach et al. 2014) and Borangiu et al. (2015) architectures. This section presents the proposed approach.

3.1.1 *Constituent elements: decisional entity*

The constituent element of the proposed approach is a decisional entity. A decisional entity is composed of an entity objective (single or multiple), a decision-making technique, parameters, governance parameters, a communication component, a data-storage component and an execution component (see Figure 1(a)). In this approach, the main characteristic of the decisional entities is that the decisional process is ruled by the governance parameters. These are explicit sets of parameters (defined at the beginning but with the possibility of being changed during execution) that define the attributes and rules of conduct that dictate the decisional entity action. For example, they can define the objective(s), the interrelation with other entities, the decision-making technique, the roles of the entities in the shared environment, the priority of objectives in a multi-objective environment, the conveyor speed or machine capacity, amongst others. The communication component acts as a data transmitter for the other entities. The data-storage component consolidates relevant information during system operation. The execution component performs the action of the decisional entity.

The decisional process starts by sensing the current job shop state through the communication component. Then, with the aim of executing the objective (minimise the makespan or minimise the route for the next operation), the decision-making technique is activated subject to the current control configuration (parameters and governance parameters).

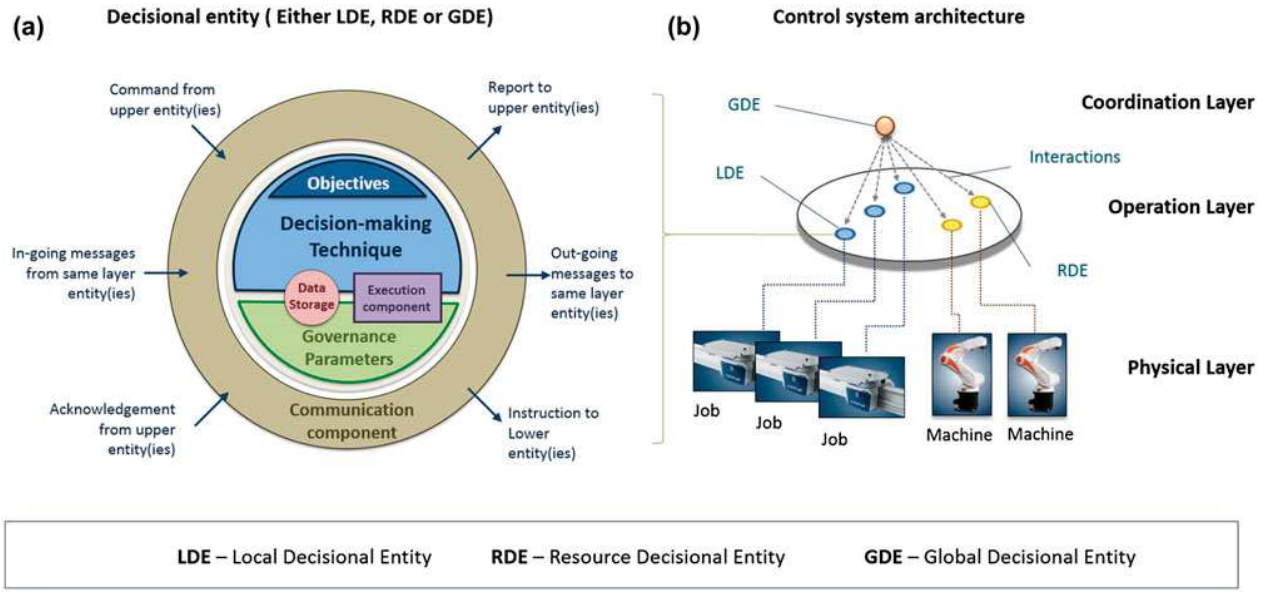


Figure 1. (a) Internal composition of decisional entities and (b) structure of a general control system architecture.

This technique evaluates action alternatives, chooses an action solution based on its own objective, and commands the actions through the execution component. The configuration of the decisional entity is a key driver in the flexibility and capability achieved in the proposed approach. For example, depending on the operations strategy, the objective or objectives can meet criteria such as minimising the earliness/tardiness of delivery date, minimising cost, minimising energy consumption, among many others. Furthermore, the operating modes, or possible executing alternatives, are derived from the defined governance parameters and decision-making techniques. For instance, the job or resource can be configured with governance parameters that define alternative process routines or the possibility of changing the machines. Concisely, the configuration of decisional entities set the capabilities for inferring the manufacturing system.

3.1.2 *Arrangements of the control system architecture*

The control system architecture of this approach is organised in three layers: coordination layer, operation layer and physical layer (see Figure 1(b)). While the coordination level hosts the decisional entities responsible for global production optimisation, the operation level hosts the decisional entities responsible for the functioning and reactivity of the jobs. The resources and jobs of the flexible job shop are located in the physical layer. This architecture is composed of three main types of decisional entities: local decisional entities (LDE), resource decisional entities (RDE) and global decisional entities (GDE). Each one is a virtual decisional entity capable of sensing, processing, storing and acting in the production environment. This specific definition of the control system architecture and decisional entities used the concept proposed by Zambrano Rey (2014)

The LDEs, located in the operation layer, are responsible for coordinating the online scheduling and guiding the jobs located in the physical layer (raw materials, work-in-progress or finished products). The LDE has all the information related to the manufacturing of the jobs such as the bill of materials, the production sequences and the constituent operations. The main objective is to support the reactivity requirement in the case of unexpected events. For this, it runs a decision-making process based on a reactive decision-making technique to guide its behaviour on the flexible job shop and to support its reactivity.

The RDEs, located in the operation layer, are responsible for controlling the service-oriented resources located in the physical layer (conveyors, robots, storage systems, AGV, etc.), and fulfilling the objective assigned to the flexible job shop (i.e. product processing, energy management, machine productive/idle management, maintenance management). These entities have all the information related to the resource such as processing times, layout information, storage capacity and operation capacity. The main objective of the RDEs is to maintain resource-related goals whilst ensuring jobs processing. For this, it processes the jobs and hosts a decision-making technique that optimises resources behaviour/utilisation.

The GDEs, located in the coordination layer, are responsible for the offline scheduling and fulfilling the global objectives (i.e. completion of production order, energy management). For this, they host a predictive decision-making technique to guide the achievement of goals and support the optimality requirement of the control system.

3.1.3 *Interactions between decisional entities*

Pollux includes a novel contribution for characterising the control system architecture. This approach dynamically changes the interaction of decisional entities to modify the arrangement of the control system architecture. The pairwise relation between the governance parameters of two entities defines the interaction between these two decisional entities and their level of interaction can range from a fully master/slave hierarchical interaction to a fully cooperative heterarchical relationship. In the end, the control architecture is characterised by the emergence of the entire set of interactions within the existing decisional entities. As an example, a GDE might have a governance parameter to define the role in the machine sequence decision. Specifically, the role refers to the function or part played in the negotiation regarding machine sequence. The possible values of this role might be coercive, limitary or permissive. While coercive and limitary roles are based on the interaction concept proposed by (Zambrano Rey 2014), the permissive role is created as a complementary role from the two already defined. A *coercive role* corresponds to the direct command of actions (impose an objective, behaviour or action) to be performed by the LDE or RDE. A *limitary role* concerns the case when the GDE proposes a set of solutions for the LDE or RDE in a modified master/slave relation. This relation gives some but not complete autonomy to slave entities. A *permissive role* is when the GDE delegates full decisional autonomy to LDE and RDE entities. Figure 2(a) illustrates this concept as it shows a GDE with four governance parameters as an interaction between each LDE in the control system architecture.

This approach also uses the governance parameters of the entire set of decisional entities (GDE, LDE and RDE) to define the operating mode of the *Pollux architecture*. The operating mode gathers the governance parameters of the

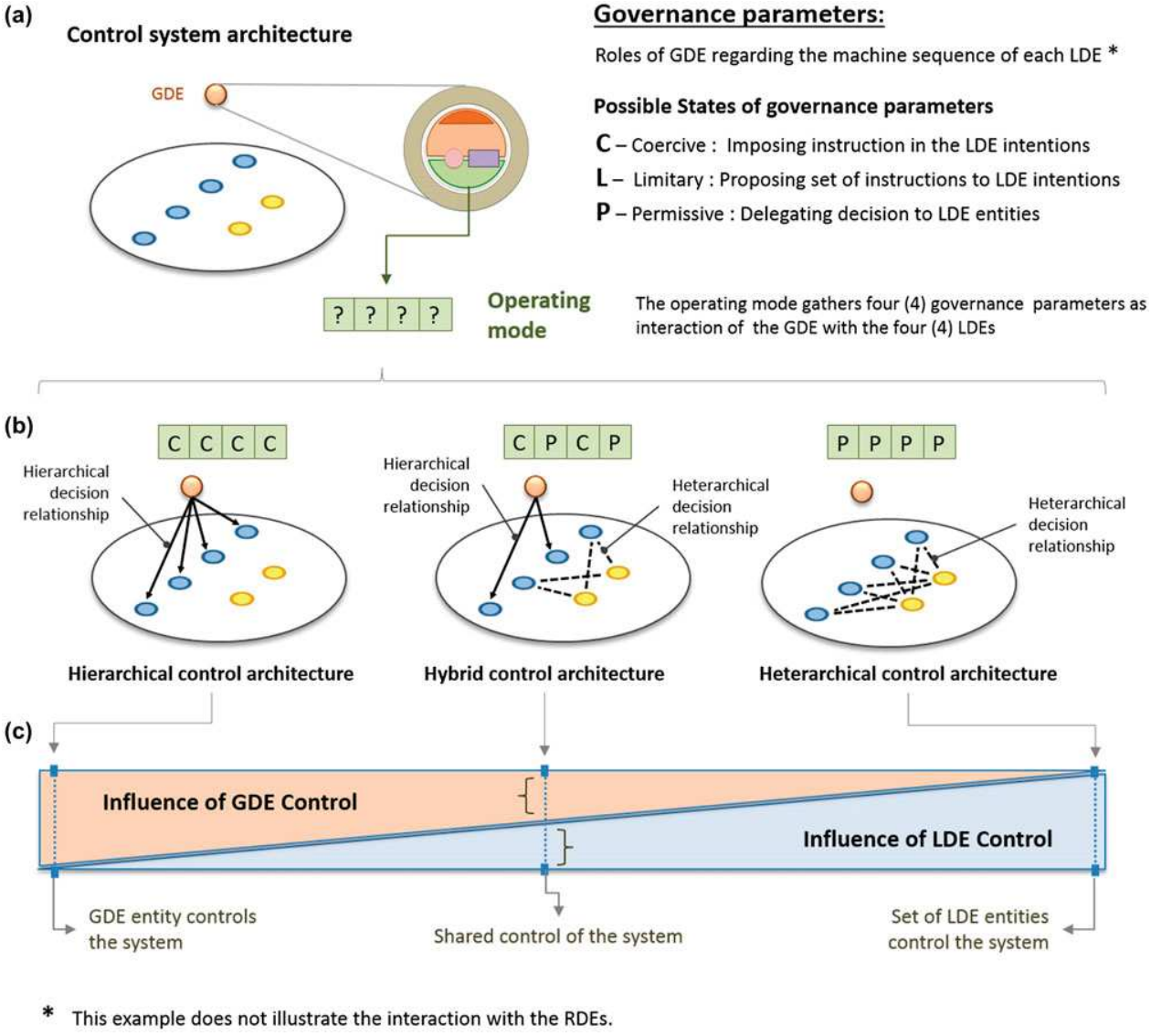


Figure 2. (a) Relation between GDE's governance parameters and an operating mode, (b) operating modes defining the structure of the control system architecture and (c) illustration of control of dynamic hybrid control architectures.

entire set of decisional entities in a vector, where it becomes an identification characteristic. The advantages of using this representation are that this unique vector characterises the control system architecture, helps evaluate the benefits of the architecture in advance, distinguishes a unique architecture capability, provides some insight into an expected result and serves as a comparator between different operating modes. Figure 2(b) illustrates how the architecture is defined by the operating modes, using an example with a single GDE, four LDEs and two RDEs. Although the operating mode defines the interaction with all the LDEs and RDEs, in this example only the relation with the LDEs is illustrated.

Another characteristic of this approach is that the operating mode defines the structure and the influence (decisional weight) of the entities over the control system architecture. For instance, in a control system architecture with only one GDE and several LDEs and RDEs, if the GDE has a coercive role over the LDE and RDE in the machine sequence, a fully hierarchical architecture emerges. This implies that the GDE has complete control over the flexible job shop system. On the contrary, if the role of the GDE is defined as permissive, a fully heterarchical architecture emerges. As a result, the LDE has complete control over the flexible job shop. Figure 2(c) illustrates this characteristic.

3.2 *Dynamic characteristics of the control system architecture*

A switching mechanism is introduced in this approach to manage the dynamism of the control system architecture. Indeed, it adjusts the operating mode to respond to flexible job shop requirements (typically, opportunities and perturbation reactiveness). The goal of the switching is to pursue an optimal operating mode according to these events. In this sense, a switching mechanism in Pollux is defined as an external or internal instrument of the D-HCA responsible for changing the operating mode in the control system architecture. In general, the switching mechanism process has two responsibilities. Initially, it is in charge of tuning the initial operating mode $\boldsymbol{U_1}$ according to the complexity of the jobs to be processed and system availability. This is considered as the offline process. When execution begins, it is in charge of monitoring the flexible job shop, triggering the switching, executing a switching technique that searches for the new operating mode $\boldsymbol{U_j}$ and executing the changes to reconfigure the control system architecture from $\boldsymbol{U_1}$ to $\boldsymbol{U_j}$ whenever necessary. Figure 3 illustrates and details the switching mechanism process after the occurrence of an unexpected event.

3.4 *Implementing Pollux*

In order to facilitate the implementation of *Pollux*, the following list presents different possibilities for setting up this D-HCA.

- **Structural characteristics**
  (a) *Constituent elements:* holons, agents, part-units or objects.
  (b) *Interaction of elements:* UML (unified modelling language), agent-UML, contract nets, FIPA-ACL (agent communication language for the FIPA standards organisation), XML (extensible mark-up language) or sockets communication.

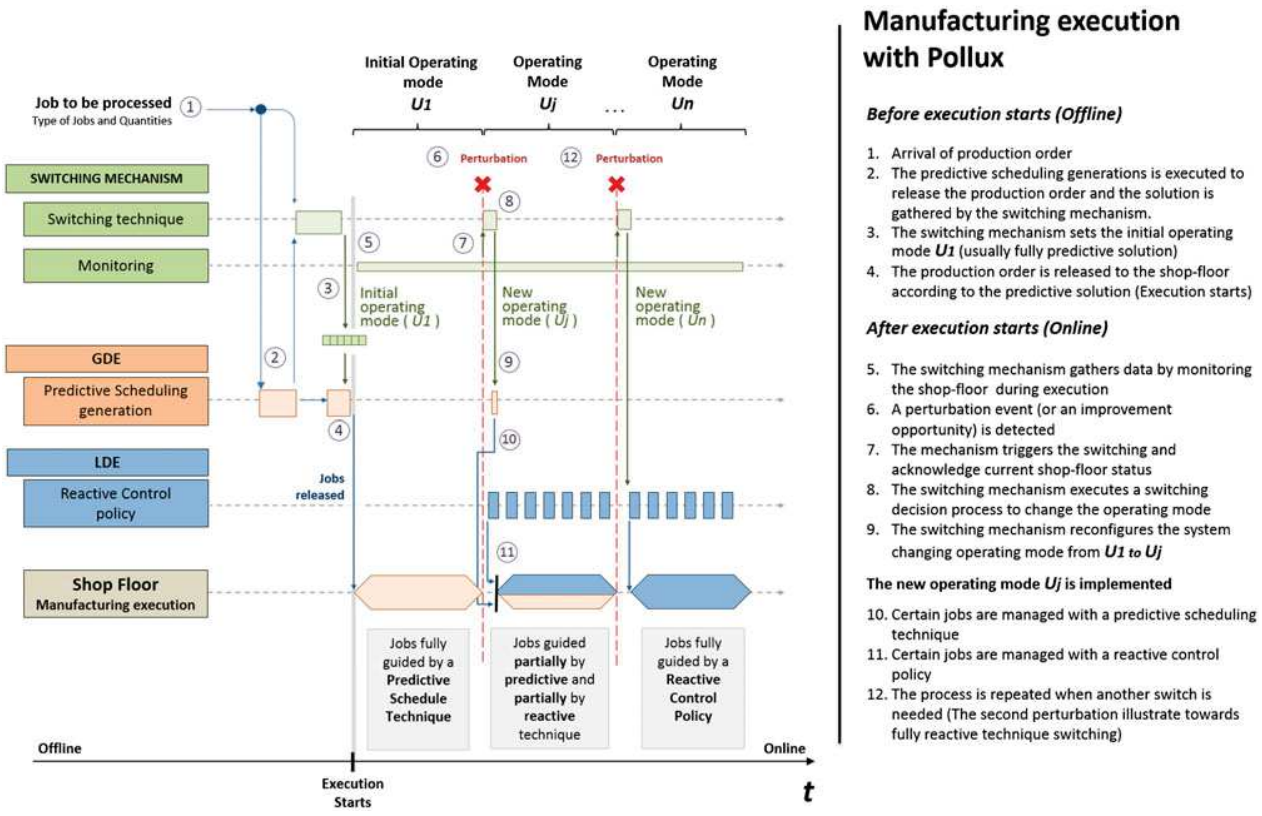


Figure 3. General process of Pollux's switching mechanism.

- **Behavioural characteristics**

(a) *Predictive schedule generation:* mathematical programming (linear programming, integer programming, mixed integer and linear programming), non-linear programming or metaheuristics (genetic algorithms, grasp, tabu search, bio-inspired).
(b) *Reactive control policy:* heuristics, dispatching rules, bidding processes, first available machine potential fields, neural networks or genetic learning.

- **Dynamic characteristics of control system architecture**
  (a) *Switching type:* structural and behavioural.
  (b) *Switching purpose:* reacting to perturbations, evolving to better conditions or both.
  (c) *Operating mode:* operating modes used in *Pollux.*
  (d) *Switching mechanism:* mathematical programming, approximation algorithms, evolutionary algorithms, swarm algorithms, artificial intelligence, heuristics or simulation-based optimisation.
  (e) *Degree of optimality:* optimal, near-optimal, towards-optimal, heuristics or reaction transition.

4. Application of Pollux: case study

This section presents the application of Pollux on a real full-size academic flexible job shop system. In this study, Pollux dynamically controls scheduling by determining job dispatch and machine order for each job to be processed, named job dispatching and machine sequence, respectively, in this paper. Production is considered in a dynamic environment and predefined perturbations are examined. This section is organised as follows. Firstly, it is described the flexible job shop system used in this case study. Then, the structural and behavioural characteristics are presented. Afterwards, the switching mechanism is presented. Finally, the technical implementation of the case study in the manufacturing system is described.

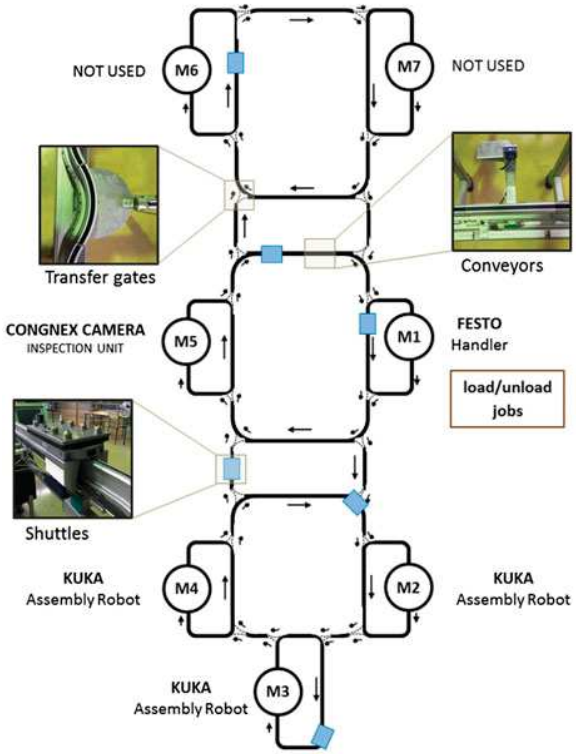
4.1 *System description*

The case study presented in this paper is for the production in the AIP-PRIMECA flexible job shop located at the University of Valenciennes. The flexible job shop system is composed of seven machines connected by a MONTRACTEC monorail transport system with self-propelled shuttles. The transport system contains 22 transfer gates (controlled by 18 PLC-Wago controllers 750-841) that move according to the route requested by the shuttles. The machines are three KUKA assembly robots (M_2, M_3 and M_4), an automated inspection unit composed of a COGNEX camera (M_5), a manual-inspection unit (M_6), a redundant assembly robot (M_7) and a FESTO handler for loading/unloading jobs (M_1). Machines M_6 and M_7 are not used in this paper. Seven types of jobs (B, E, L, T, A, I and P) can be produced. Each job has a predetermined sequence of operations to be processed. The *jobs to be processed* are defined in a data-set which includes the number of jobs to be produced and the number of shuttles permitted in the flexible job shop at the same time. During execution, each job is dispatched for processing when the load/unload machine (M_1) positions a plate on a shuttle. The shuttles are self-propelled devices that prepare and guide the jobs through the transport system.

For more information about the flexible job shop system at Valenciennes, the interested reader can consult Trentesaux et al. (2013). Figure 4 illustrates the layout, the sequence of operations, the processing times, and the operation processed in each machine from the AIP-PRIMECA facility (Valenciennes).

4.2 *Structural and behavioural characteristics*

The proposed approach is built over three layers: *the operation, the coordination and the physical layer*. The operation layer contains *n* LDEs as jobs to be produced, and five RDEs as available machines in the flexible job shop system. A GDE is created to hold the predictive schedule generation technique (offline) of the jobs to be processed. For this, it solves the dispatching and machine sequence of each job according to the MILP model formulated in Trentesaux et al. (2013). For this, it was used the IBM ILog Cplex optimisation studio (IBM ILOG CPLEX 2016) using concert technology (C++). The LDE reactive control policy is guided by the potential fields approach. The potential fields approach is a reactive control policy technique that guides the routeing of the jobs depending on the emission of the potential fields of the RDEs (e.g. Machines). This field, which can be attracting or repelling fields, is dynamically calculated by the



Code	Letter	Operations Sequence
901	B	O7 - O1 - O1 - O1 - O2 - O2 - O5 - O3 - O6 - O8
902	E	O7 - O1 - O1 - O1 - O2 - O2 - O4 - O6 - O8
903	L	O7 - O1 - O1 - O1 - O5 - O5 - O3 - O3 - O6 - O8
904	T	O7 - O1 - O1 - O2 - O4 - O6 - O8
905	A	O7 - O1 - O1 - O1 - O2 - O4 - O5 - O3 - O6 - O8
906	I	O7 - O1 - O1 - O5 - O3 - O6 - O8
907	P	O7 - O1 - O1 - O2 - O4 - O6 - O8

	Processing times per machine (Seconds)				
Operation	M1	M2	M3	M4	M5
Operation 1 (O1)		20	20		
Operation 2 (O2)		20	20		
Operation 3 (O3)				20	
Operation 4 (O4)		20		20	
Operation 5 (O5)			20	20	
Operation 6 (O6)					5
Operation 7 (O7)	10				
Operation 8 (O8)	10				

Figure 4. Flexible job shop at the University of Valenciennes.

availability of the machines (RDEs) that process the requested operation and distance between machine and job (LDEs). Further reading about potential fields in manufacturing, applications can be found in Pach et al. (2012).

The instantiation of the decisional entities (GDEs, LDEs and RDEs) is illustrated in Table 2. The GDE contains two governance parameters for each LDE for defining the pairwise interaction. Whilst the first governance parameter defines the role of the GDE regarding the dispatching of the job, the second governance parameter defines the machine sequence of the LDE. This definition in the GDE governance parameters is extrapolated for each LDE in the control system architecture. For instance, if there four LDEs, the GDE will have eight governance parameters. In this case study, the values of the governance parameters are coercive (C), to impose GDE intentions on the LDEs or permissive (P), to provide LDEs with local autonomy.

4.3 *Dynamic characteristics and switching mechanism*

The switching mechanism in the proposed D-HCA is based on the concept of evolutionary algorithms, specifically genetic algorithms. A genetic algorithm (GA) is a population-based optimisation technique that simulates the evolution process in order to reach an optimal or near-optimal solution according to a fitness function (Holland 1992). It has powerful optimisation ability, fast calculation, simple principles and operations, robust generality, and global search space ability (Pan et al. 2011). In this technique, an appropriate encoding, called chromosome, is defined to represent a feasible solution to the problem. It is used because GA chromosomes adjust perfectly to the operating modes and the representation of the control system architecture. The switching mechanism aims to maintain a set of feasible operating modes that evolve in parallel with execution and serves as a contingency plan in case of a disruptive event.

The process starts when the switching mechanism retrieves the data from the set of jobs to be processed. The switching mechanism uses the GA technique in two separate instances, named *GA-tuning* and *GA-contingency techniques*. Firstly, the GA-tuning technique is used to form a population of different operating modes of the control system architecture. It aims to obtain alternative operating modes in terms of population efficiency and diversity. Then, when execution starts, the GA-contingency technique periodically improves the same population based on its fitness function. This technique runs repeatedly and in parallel with execution every Δt according to the corresponding manufacturing

Table 2. Composition of each element.

	GDE	LDE	RDE
Number of entities	1	*n*	5 (M_1, M_2 M_3 M_4 M_5)
Responsibility	Predictive scheduling of job to be processed	Guide job execution and coordinate reactive schedules if necessary	Process the jobs and inform status of completion
Decisional components			
Objective	Minimising total jobs makespan	Minimise completion time of next operation	Null (not used in case study)
Decision-making technique	Mixed integer linear programming (MILP) programmed in IBM Ilog Cplex	Rational decision: evaluation of available alternatives and proximity to machine for next operation with a heterarchical relation (or following imposed decision with a hierarchical relation)	Null (not used in case study)
Governance parameters	Two for LDE with role in dispatching and machine sequence decisions, Coercive or Permissive (14 in total)	No flexibility in behaviour	Null (not used in case study)
Communication components			
GDE with …	Not necessary (unique entity)	Coercive imposes MILP intentions. Permissive allows LDE autonomy	Not used
LDEs with …	Information regarding current flexible job shop status	Not used	Request the potential field when it reaches a decisional point
RDEs with …	Report availability	Broadcasts machine availability using potential fields	Not used

conditions. When a perturbation occurs, the switching decision process changes the operating mode according to the most suitable chromosome from the most recent GA-contingency solution. Finally, the system adopts the alignments of the new operating mode to react to the new environmental conditions. The parameters of the genetic algorithms are presented in the next subsection. In addition, as an instance of the switching process in Section 3.2, Figure 5 illustrates the corresponding process of the switching mechanism described above.

4.4 *Design of GA-tuning and GA-contingency techniques*

This subsection presents the settings of the GA-tuning and the GA-contingency techniques. Figure 6 illustrates the use of the genetic algorithm in the *Pollux* switching mechanism process.

- *Chromosomes:* the chromosomes for the GA represent a control strategy in the control system. For this, both GA techniques uses the operating modes of the control system architecture. The operating mode is described in Section 3.2.
- *Coding:* the coding is these techniques is used to represent the governance parameters in each decisional entity. Both techniques uses a binary value at each gene according to the role of the GDE in the production environment (1 = Coercive role/0 = Permissive role). For example, if the gene of the chromosome is 1 for the first position, the interaction of the GDE with the correspondent LDE (previously allocated) is coercive and the LDE will follow the commands given by the GDE. Likewise, if the gene is 0, the correspondent LDE will act by its own decision-making technique and will ignore the GDE commands.
- *Population:* subset of feasible operating modes in the control system architecture.
- *Initial population:* for the GA-tuning technique, 10 chromosomes were selected for the initial population. Since the GA is only used for setting the control system architecture (and not for job scheduling, for example), a previous empirical study for this case study showed that this number of chromosomes was sufficient for our objective. For the GA-contingency technique, the initial population was obtained from the last iteration population of the GA-tuning technique.

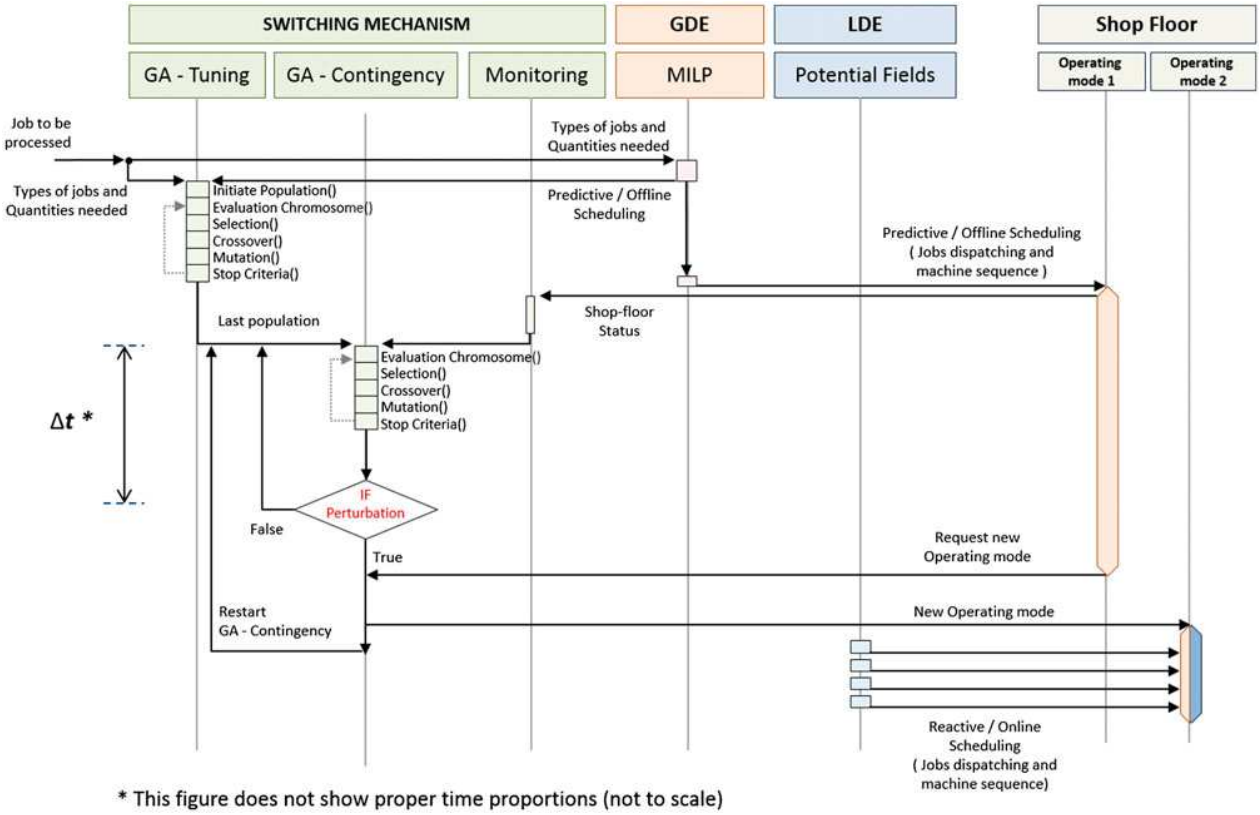


Figure 5. Sequence diagram of the switching mechanism for the case study.

- *Evaluation chromosome:* a simulation model of the same flexible job shop programmed in NetLogo agent-programming software was evaluated (Wilensky 1999).
- *Offspring selection:* the GA-tuning technique used the restricted tournament method. The GA-contingency technique used the tournament method.
- *Offspring crossover:* both techniques have a uniform crossover with 50%/50% from parent genes.
- *Offspring mutation:* both techniques use the bit inversion mutation with a probability $p_m = 0.3$.
- *Stop criteria:* the stopping criteria of the GA-tuning technique were three repetitions of the population average after achieving an optimality gap of 10%. The GA-contingency technique had a limited execution time of Δt.

4.5 *Technical implementation of the case study*

Consistently with the architecture of Pollux, the application was implemented in three layers. The hardware from the system was included in level 0 as it holds the physical layer of the flexible job shop (robots, inspection units, shuttles, transfer gates, positioning units). Level 1 holds the operation layer with a network of laptop computers allocating the LDEs (Asus Eee PC Intel (R) atom (TM) 1015PEM CPU@ 1.50 GHz with 1.00 GB of RAM memory) as jobs to be processed. Level 2 holds the coordination layer and the GDE in a PC (Intel(R) Core(TM) i5-3317U CPU@ 1.70 GHz with 4.00 GB of RAM memory). While each laptop computer runs a Java program for the LDEs behaviour, the PC runs a Java scheduling program for the GDE. The agent-based software Netlogo is used in this paper to simulate beforehand the comportment of the case study flexible job shop, specifically to evaluate the fitness of the GA chromosome. The IBM Cplex Optimisation software is used to program the MILP model (Trentesaux et al. 2013) of the flexible job shop, where it schedules offline the jobs. The three levels are connected via an Ethernet network and they communicate using the TCP/IP protocol. A WLAN connects the laptop computers and is connected to the Ethernet network through a Router. In the next section, the experiments conducted for the case study and the results to verify the contribution of our approach are presented.

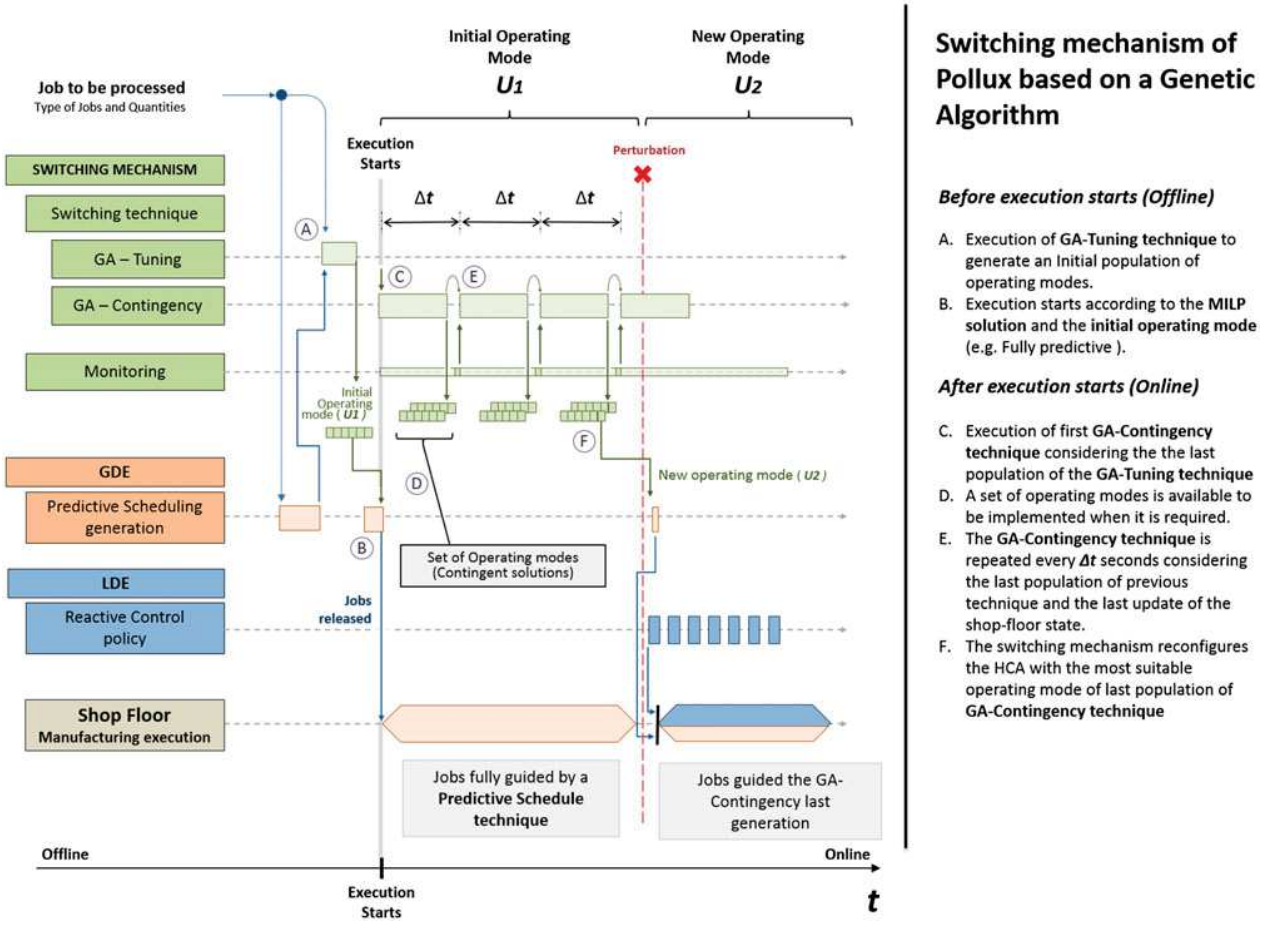


Figure 6. Switching mechanism process applied to the case study.

5. Experiments and results

This section presents the experiments conducted in the introduced flexible job shop to demonstrate the contribution of *Pollux* in manufacturing control systems. The benchmark data-set C0 of Trentesaux et al. (2013) was tested in the AIP-PRIMECA. The jobs to be processed are a data-set that includes one job of each type (B, E, L, T, A, I and P), limited to four shuttles at a time. The operating mode for this data-set is the 14 genes as the interaction of the GDE with the LDEs. The perturbation considered is a breaking down of the machine (M_3) at 100 s after the execution starts. For this, the machine is disconnected from the system. To outline the advantages of *Pollux* in the experiments, four scenarios were created. In scenarios A and B, *Pollux* was tested with different **Δ*t*** parameters for the GA-contingency technique (switching mechanism). Scenario C was considered as a reference scenario, presented in the ORCA approach (Pach et al. 2014), to compare the performance. This scenario starts the production execution with a predictive scheduling technique. It starts with an MILP model and changes to reactive technique (also potential fields) at the switching point. Still, whilst this change is performed just for the affected jobs, the not affected jobs stay with the predictive solution. Finally, scenario O is a fully hierarchical architecture with no perturbations.

In the first part of the experiment, considering that the only difference between scenarios A and B is the parameter of the GA-contingency technique, it is conducted the GA-tuning technique for these scenarios. Figure 7 presents the evolution of the GA-tuning technique, which plots the total makespan of the jobs processed and the time taken for each algorithm iteration.

In Figure 7, it can be seen the evolution of the chromosomes (operating modes) considered in the GA-tuning technique. This technique tunes the initial population of the GA-contingency technique in terms of optimality (average makespan of 423.6) and diversity (standard deviation of 12.51 due to restricted tournament selection). The average time to execute each chromosome in Netlogo was 0.286 s. The convergence of the GA-tuning technique in these experiments

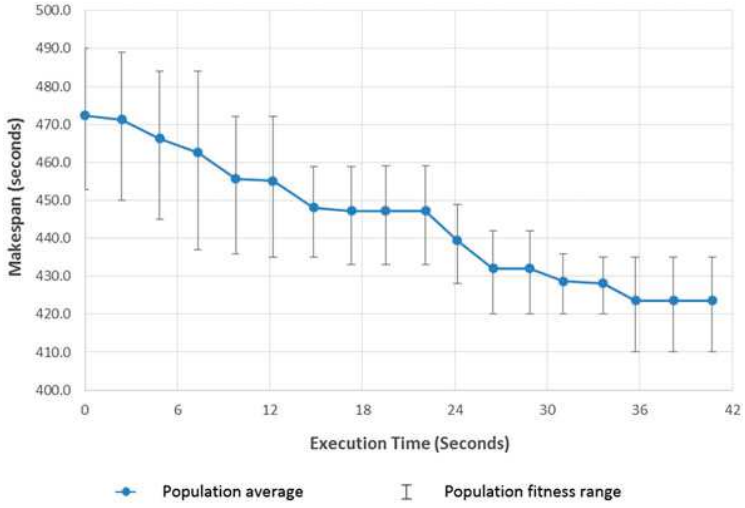


Figure 7. Process evolution in the GA-tuning technique.

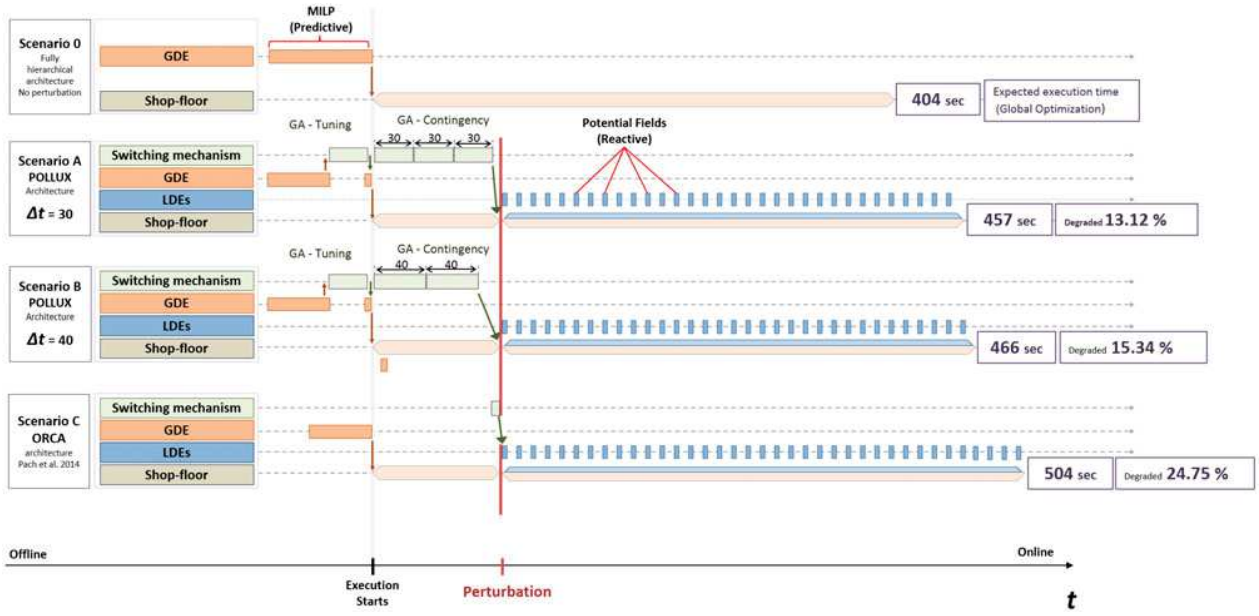


Figure 8. Makespan comparison for three different execution strategies.

was achieved in 35.72 s (15 iterations). The parameter Δt for the GA-contingency technique was defined as 30 and 40 s for two cases before and after the algorithm convergence. Still, this definition assumes that GA-contingency technique has a similar comportment to GA-tuning technique as it has the same population and its fitness function is evaluated using the same simulation tool (NetLogo).

In the second part of the experiments, four scenarios were tested in the real flexible job shop. In scenario O, the jobs were dispatched resulting a makespan of 404 s. In scenario A, the production of the jobs was conducted implementing *Pollux* with a GA-contingency technique every 30 s. For this, the switching mechanism retrieved the data from the flexible job shop every 30 s to readjust the initial state of the Netlogo Model. After the perturbation, the makespan of scenario A was 457 s (13.12% degradation). In scenario B, where the GA-contingency technique was executed every 40 s, the makespan was 466 s (15.34% degradation). In scenario C, the makespan was 504 s (24.75% degradation). Figure 8 illustrates the results of the experiments and compares the results with an existing D-HCA approach (ORCA).

The results outline that the proposed approach has a better performance than a reference approach (ORCA), but complementary statistical studies must be led to confirm possible generalisation of these results. In scenario C, even though it is observed that jobs unrelated to Machine 3 (perturbed resource) are not directly affected, the affected jobs have an indirect impact as they employ the previously assigned and available resources. Consequently, no affected jobs start looping through the flexible job shop searching for the predictive intention. On the contrary, in scenarios A and B, it was observed that the new operating modes adjusted better to the flexible job shop conditions. Specifically, the new operating mode pre-evaluated in a simulated tool (Netlogo), considered switching the behaviour of unaffected jobs if needed.

From the experiments led in the real manufacturing system, three different remarks can be deduced. Firstly, an improvement in reactivity requirement was demonstrated in situations where the proposed D-HCA adjusted the control solution to a better operating mode. In this sense, it is pertinent to continue research on the inclusion of a switching mechanism to tailor production control. Secondly, the inclusion of a switching mechanism with an evolutionary technique represents a promising area of research for D-HCA. In our experiments, this mechanism executed in parallel provides a set of alternatives to apply in response to unexpected events. However, we are aware that this technique requires further research, as many problems can occur such as the possibility of not finding a suitable operating mode from the set of alternatives due to the nature of a perturbation. Nonetheless, a mechanism that searches a sufficiently diverse population motivates us to pursue our research in this direction. Finally, this research helps prove that the D-HCA features dynamic autonomy shared between global and local control. In this sense, the inclusion of a switching mechanism is appropriate, as it is needed to adjust this coupled autonomy throughout production execution.

As a synthesis, the main contributions of this paper are twofold. Firstly, this paper proposes a reference architecture named Pollux, which contains a switching mechanism for dynamic hybrid control architectures. Pollux features a definition of operating modes that provides a configuration flexibility to the control system architecture. Secondly, this paper proposes a methodology to include an optimisation method in the switching mechanism. This enables the most suitable operating mode of the control system architecture to be sought during execution and fulfils optimality and reactivity requirements. Pollux then proposes a custom-built configuration of the control system architecture according to online production necessities.

6. Conclusions and future work

In this paper, Pollux, a D-HCA for the dynamic scheduling of a flexible job shop problem is presented. *Pollux* is presented as a reference control system that supports the switching process of control system architectures. A general process of the switching mechanism and the implications to be considered to achieve optimal switching have been described. An instantiation of Pollux is discussed and tested in a real flexible job shop. The results of the experiments conducted in this research illustrate that our model helps support the optimality and reactivity requirements demanded in production execution. Still, this approach needs complementary statistical studies in order to be generalised.

Several lines of research can be derived from this work. In the short term, considering the genetic algorithm approach for switching in Pollux, it is necessary to study the threshold between the calculation of a contingency time and the perturbation. In the medium term, it is necessary to develop different types of switching mechanisms with different degrees of optimality to compare performance in dynamic scheduling. This research requires a balance between the optimality required and the time of execution of the switching mechanism to provide an efficient dynamic solution. In the long term, the applicability of this approach to other manufacturing control problems and the feasibility of using the same approach in other domains (i.e. logistics or health care) may be studied.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported by the Colombian scholarship programme of department of science – COLCIENCIAS under grant ‘Convocatoria 568 – Doctorados en el exterior’ and the Pontificia Universidad Javeriana under grant ‘Programa de Formacion de posgrados del Profesor Javeriano’.

ORCID

Jose-Fernando Jimenez http://orcid.org/0000-0001-8336-6240

References

Baker, A. D. 1998. "A survey of factory control algorithms that can be implemented in a multi-agent heterarchy: dispatching, scheduling, and pull." *Journal of Manufacturing Systems* 17 (4): 297–320.

Barbosa, J., P. Leitão, E. Adam, and D. Trentesaux. 2015. "Dynamic Self-organization in Holonic Multi-agent Manufacturing Systems: The ADACOR Evolution." *Computers in Industry* 66: 99–111. doi:10.1016/j.compind.2014.10.011.

Böhnlein, D., K. Schweiger, and A. Tuma. 2011. "Multi-agent-based Transport Planning in the Newspaper Industry." *International Journal of Production Economics* 131 (1): 146–157. doi:10.1016/j.ijpe.2010.04.006.

Bongaerts, L., L. Monostori, D. McFarlane, and B. Kádár. 2000. "Hierarchy in Distributed Shop Floor Control." *Computers in Industry* 43 (2): 123–137. doi:10.1016/S0166-3615(00)00062-2.

Borangiu, T., S. Răileanu, T. Berger, and D. Trentesaux. 2015. "Switching Mode Control Strategy in Manufacturing Execution Systems." *International Journal of Production Research* 53 (7): 1950–1963. doi:10.1080/00207543.2014.935825.

Cardin, O., D. Trentesaux, A. Thomas, P. Castagna, T. Berger, and H. B. El-Haouzi. 2016. "Coupling Predictive Scheduling and Reactive Control in Manufacturing Hybrid Control Architectures: State of the Art and Future Challenges." *Journal of Intelligent Manufacturing*: 1–15. doi: 10.1007/s10845-015-1139-0

Dilts, D. M., N. P. Boyd, and H. H. Whorms. 1991. "The Evolution of Control Architectures for Automated Manufacturing Systems." *Journal of Manufacturing Systems* 10 (1): 79–93. doi:10.1016/0278-6125(91)90049-8.

Gunasekaran, A., and E. W. Ngai. 2012. "The Future of Operations Management: An Outlook and Analysis." *International Journal of Production Economics* 135 (2): 687–701. doi:10.1016/j.ijpe.2011.11.002.

Herrera, C., A. Thomas, and V. Parada. 2014. "A Product-driven System Approach for Multilevel Decisions in Manufacturing Planning and Control." *Production & Manufacturing Research* 2 (1): 756–766. doi:10.1080/21693277.2014.949895.

Holland, J. H. 1992. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. Arbor, MI: University of Michigan Press.

Holvoet, T., Weyns, D., and Valckenaers, P. 2009. "Patterns of Delegate Mas". *Third IEEE International Conference on Self-adaptive and Self-organizing Systems*, 2009. SASO'09. IEEE. September 1–9. doi:10.1109/SASO.2009.31.

IBM ILOG CPLEX Optimize studio. *High-Performance Mathematical Optimization Engines*. http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/

Jimenez, J. F., A. Bekrar, D. Trentesaux, G. Z. Rey, and P. Leitao. 2015. "Governance Mechanism in Control Architectures for Flexible Manufacturing Systems." *IFAC-PapersOnLine* 48 (3): 1093–1098. doi:10.1016/j.ifacol.2015.06.229.

Jimenez, J. F., A. Bekrar, D. Trentesaux, G. Z. Rey, and P. Leitao. 2016. "A Switching Mechanism for Optimal Coupling of Predictive Scheduling and Reactive Control in Manufacturing Hybrid Control Architectures." *International Journal of Production Research*: 1–16. doi:10.1080/00207543.2016.1177237.

Lee, C. K. M., Y. Lv, and Z. Hong. 2013. "Risk Modelling and Assessment for Distributed Manufacturing System." *International Journal of Production Research* 51 (9): 2652–2666. doi:10.1080/00207543.2012.738943.

Leitão, P, and F. Restivo. 2006. "ADACOR: A Holonic Architecture for Agile and Adaptive Manufacturing Control." *Computers in Industry* 57 (2) (February): 121–130. doi:10.1016/j.compind.2005.05.005.

Novas, J. M., J. Van Belle, B. Saint Germain, and P. Valckenaers. 2013. "A Collaborative Framework between a Scheduling System and a Holonic Manufacturing Execution System." In *Service Orientation in Holonic and Multi Agent Manufacturing and Robotics*, 3–17. Berlin Heidelberg: Springer. doi: 10.1007/978-3-642-35852-4_1.

Pach, C., A. Bekrar, N. Zbib, Y. Sallez, and D. Trentesaux. 2012. "An Effective Potential Field Approach to FMS Holonic Heterarchical Control." *Control Engineering Practice* 20 (12) (December): 1293–1309. doi:10.1016/j.conengprac.2012.07.005.

Pach, C., T. Berger, T. Bonte, and D. Trentesaux. 2014. "ORCA-FMS: A Dynamic Architecture for the Optimized and Reactive Control of Flexible Manufacturing Scheduling." *Computers in Industry* 65 (4) (May): 706–720

Pan, Y., W. X. Zhang, T. Y. Gao, Q. Y. Ma, and D. J. Xue. 2011. "An Adaptive Genetic Algorithm for the Flexible Job-Shop Scheduling Problem." *2011 IEEE International Conference on Computer Science and Automation Engineering* 4: 405–409. IEEE. doi: 10.1109/CSAE.2011.5952878.

Raileanu, S., M. Parlea, T. Borangiu, and O. Stocklosa. 2012. "A JADE Environment for Product Driven Automation of Holonic Manufacturing." In *Service Orientation in Holonic and Multi-agent Manufacturing Control*, 265–277. Berlin Heidelberg: Springer. doi: 10.1007/978-3-642-27449-7_20.

Saharidis, G. K., Y. Dallery, and F. Karaesmen. 2006. "Centralized vERSUS Decentralized Production Planning." *RAIRO – Operations Research* 40 (2): 113–128. doi:10.1051/ro:2006017.

Trentesaux, D. 2009. "Distributed Control of Production Systems." *Engineering Applications of Artificial Intelligence* 22 (7): 971–978. doi:10.1016/j.engappai.2009.05.001.

Trentesaux, D., C. Pach, A. Bekrar, Y. Sallez, P. Leitao, and J. Barbosa. 2013. "Benchmarking Flexible Job-shop Scheduling and Control Systems." *Control Engineering Practice* 21 (9) (September): 1204–1225. doi:10.1016/j.conengprac.2013.05.004.

Valckenaers, P., H. Van Brussel, P. Verstraete, and B. Saint Germain. 2007. "Schedule Execution in Autonomic Manufacturing Execution Systems." *Journal of Manufacturing Systems* 26 (2): 75–84. doi:10.1016/j.jmsy.2007.12.003.

Wilensky, U. 1999. *NetLogo*. Evanston, IL: Center for Connected Learning and Computer-Based Modeling, Northwestern University. http://ccl.northwestern.edu/netlogo/.

Zambrano Rey, G. 2014. *Reducing Myopic Behavior in FMS Control: A Semi-heterarchical Simulation Optimization Approach*. PhD diss., University of Valenciennes and Hainaut-Cambrésis (France).

Zambrano Rey, G., A. Bekrar, V. Prabhu, and D. Trentesaux. 2014. "Coupling a Genetic Algorithm with the Distributed Arrival-time