Distributed Multi-Robot Coordination Combining Semantics and Real-Time Scheduling

Abstract — In this paper, we study a distributed intelligent multi-robot system (MRS) in assembly setting where robots have partially overlapping capabilities. We treat the problem of the system’s real-time self-(re)configurability and self-optimization. In this light, we propose an optimized distributed coordination system ORCAS that integrates the MRS configuration based on semantic descriptions and process scheduling. In more detail, initially, robots and the corresponding devices get matched semantically to respond to the assembly requests. Then, the best configurations are chosen by dynamically minimizing total assembly costs and off-line times. During the execution, the performance is controlled for contingencies in case of which the robots, if necessary, self-reconfigure or reschedule the tasks.

Keywords – industrial robots; ontologies; scheduling.

I. INTRODUCTION

In this paper, we study distributed and intelligent MRSs in assembly setting where robots have partially overlapping capabilities. We focus on systems’ self-configuration and self-optimization properties for seamless and distributed real-time optimization of robots’ performance in dynamically changing environments with varying resource availability and demand.

For the internal decision-making architecture of each robot, we propose a distributed and optimized configuration and scheduling system ORCAS. First, feasible semantic correspondences between product requests and the available assembly resources are found. The knowledge is presented in a distributed way such that it is possible for robots to exchange data, identify each other and if compatible, to collaborate. Furthermore, semantic correspondences are taken into account to compute the lowest cost schedule in a distributed manner among robots. Robot-task assignments and sequencing of tasks assigned to each robot are determined based on the minimization of the total assembly cost and time while respecting inventory constraints, task interrelations and the robot assembly capacities resulting from the available configurations. Finally, the actual real-time execution of the schedule is controlled and, if necessary, available resources are rescheduled and/or reconfigured.

The proposed solution with distributed semantic descriptions and scheduling allows for a modular and scalable MRS facilitating the integration of robots on a plug & play principle, online reconfiguration, and shop floor resizing without the need for stopping or reprogramming the shop-floor as is the case in most of the State-of-the-Art solutions.

This paper is organized as follows. Section II discusses the State-of-the-Art. In Section III, we introduce the integrated (re)configuration and scheduling problem. In Section IV we propose our solution model ORCAS. Section V contains conclusions and indications of our future work.

II. RELATED WORK

To increase a MRS’s autonomy and decrease the dependency on human operators, a reasonable way is to aid assembly process planning with knowledge representation and reasoning. The initial knowledge about products, equipment and production processes is needed where available software can convert the data from native formats to text with structure and to separate the structure from content [1]. Hedelind and Kock in [2] and Nägele et al. in [3] describe approaches for handling and modeling of knowledge for assembly processes in an automated production environment. In [4], the focus is on semantic knowledge use for robot task planning through semantic maps and the improvement of task planning through extending the capabilities of the planner by reasoning about semantic information and by improving the planning efficiency in large domains through semantic maps.

One of the recent achievements in robotics is a more intelligent execution of complex tasks by controllers with a high degree of autonomy through the network access to a knowledge repository and management system, e.g., [1][5]. In [5], Naumann et al. describe an integrated approach for the generation of executable programs for assembly processes using Knowledge Integration Framework (KIF). The goal of KIF is to represent, store, adapt and distribute knowledge across engineering platforms [6].

The scheduling problem is typically NP-hard and a robot cell scheduling is one of the most difficult scheduling problems since different assembly resources and tools are considered during the scheduling process. In the case of contingencies, the scheduling needs to be generated in real time. This kind of scheduling is also called dynamic or real-time scheduling. A review of the state-of-the-art of recent research on dynamic scheduling can be found in [7]. In [8] Phanden et al. present a state-of-the-art review on integration of process planning and scheduling to increase flexibility and profitability and creation of realistic process plans that can be executed readily on shop floor. Different distributed multi-agent approaches for the integration of process planning and scheduling have been proposed [9], [10]. Most agent-based manufacturing process planning and scheduling systems use negotiation protocols for resources and
tasks allocation. Contract-net protocol (CNP) with its modified versions is the most common approach. The drawbacks of traditional approaches, as CNP, are the problems with detecting or resolving conflict, high communication cost and low scalability [11].

In [12], the real-time distributed manufacturing scheduling framework at the shop floor level was presented where the framework is made of a distributed shop floor control structure, dynamic distributed scheduling algorithms, multi-agent system modeling of workcells, and service oriented integration of the shop floor. In [13], a distributed multi-agent system was developed for dynamic scheduling of mobile robots able to execute different production tasks.

To the best of our knowledge, the State-of-the-Art approaches that integrate knowledge reasoning with assembly scheduling use centralized knowledge bases, and scheduling is performed mostly over different centralized evolutionary optimization methods. In this paper, on the other hand, we propose a distributed architecture where each robot, based on its local individual skill-oriented knowledge and communication with other robots, manages its local knowledge base and distributively plans and schedules its activities in line with the system objectives. Robots collaboratively assign their tasks through a negotiation process similar to the one described in [13].

III. PROBLEM FORMULATION

We consider a multiple robot system made of \( n \) industrial robot manipulators with partially overlapping assembly capabilities producing a set \( P \) of \( q \) different types of products. Moreover, every product \( p \in P \) is semantically described by the user as a set of tasks to be executed on specific parts \( d \in D \), where \( D \) is a set of all parts. Other than industrial robots, other types of devices such as conveyor belts, computer and numerically controlled machines and other stand-alone systems such as inspection machines may also exist on the shop floor.

Let a robot configuration be a combination of a set of compatible resources, i.e., robot arm, gripper, camera, and auxiliary material. Furthermore, let us assume there is a set of \( m \) different tasks, denoted by \( J = \{J_1,...,J_m\} \). A task denotes a production process (e.g., glue, attach, transport) and is made of more atomic tasks (actions) (e.g., for glue task: open gripper, use camera, insert glue material, align and push parts together). Furthermore, each task requires a specific robot configuration for processing depending on the geometric, mechanical and physical parameters of the manipulated parts. This is why multiple resources within each robot configuration need to be combined semantically to perform a single task.

Let \( R_k \) denote a set of robot configurations eligible for the performance of task \( J_k \) \( (i = 1,..., m) \). Every robot configuration includes a robot, gripper, camera, and auxiliary material depending on the task. Each robot configuration \( R_k(i = 1,...,m) \) is capable of executing the tasks \( J_k \subseteq J \) possibly with varying efficiency in respect to time and cost. If \( J_i \in J_k(1 \leq j \leq n) \), \( J_i \) can be processed by robot configuration \( R_k \). For any two robot configurations \( R_k, R_l \in R \), nonempty \( J_k \cap J_l \) indicates overlapping capabilities.

We assume that the following information is given: assembly workflow, assembly ontology, robots’ and tools’ ontology (their possible tasks and atomic actions), and constraints on coupling of assembly resources in the robot configurations. Furthermore, we assume predefined user inputs for every assembly product \( p \in P \) and for every task \( J, J \in J \) holding, backlog, production, and other relevant costs.

Given a finite time horizon of \( T \) time periods and assuming that the demand for the \( q \) products in the \( T \) time periods is predefined, the discrete time assembly configuration and scheduling problem which we study in this paper consists of pre-calculating (offline) for each time period the configuration of robots and other necessary resources to be assigned to each product and the product assembly rate based on the minimization of a total assembly cost with included inventory constraints, task interrelations and the robot assembly capacities. Furthermore, in the case of unexpected events during the execution, robot schedule(s) and/or configuration(s) have to be updated online.

IV. ORCAS ARCHITECTURE

In order to provide a scalable modular architecture robust to failures, our proposed solution follows a distributed multi-agent approach where every physical device with processing capability as well as every product demand is considered as a collaborative agent. Fig. 1 shows the internal architecture of each robot agent made of the semantic, scheduling and execution layer.

Initially, the following semantic information is input into the system by human operators: factory setting (plant configuration), resource descriptions for available devices, tools and materials, products’ descriptions describing tasks on interrelated parts needed to make each product, and product request description with temporal demand rates of every product. Furthermore, optimization criteria and KPIs are given by the plant manager.

The Semantic Layer analyzes the product request and, using the local knowledge, generates compatible subsets of resources for the given tasks. Then, for each product, the assigned coordinator robot triggers a distributed scheduling algorithm at the Scheduling Layer and an optimal schedule is obtained. The Execution Layer is in charge of monitoring the correct execution of the schedule and, in case of unexpected events causing assembly delays or arrests, carry out the actions (individually, locally or globally) to minimize their effects.

For simplicity and w.l.o.g., we illustrate different aspects of our proposal on a simple example in Fig. 2. There are altogether 3 robots with 6 degrees of freedom (DOF) working on a conveyor belt. \( FR1 \) and \( FR2 \) can only rotate around the platform axis, while \( MR1 \) is mobile and can both move linearly and rotate. There are altogether 3 grippers, 3 cameras, and two glue materials positioned in the depots. Robots can take parts from the Assembly parts buffer if they are within their range of reach, and when they assemble a product, they leave it in the Assembled products depot. In the following, we explain in more detail each layer.
A. Semantic Layer

The Semantic Layer stores relevant factory settings, available resources and product specifications information, and obtains feasible configurations. In order to deal with the large amount of information that can be available in a complex plant, robots only store information about local and compatible resources. This way they use less memory and computational resources to carry out inference tasks, thus improving system’s scalability. Initially, every robot receives all the information and extracts and stores the pieces of the information relevant in respect to its skills and configuration description (e.g. compatible tools). Afterwards, robots broadcast the identity of their compatible resources, which is used by the others to store, for each resource, the set of compatible robots. This information is used to keep updated the local information of resources by every robot (e.g. when a robot takes or releases a tool it informs of that fact (only) the affected robots).

In the case there is an addition of new devices or tools to the system or the breakage of the present ones, local ontologies can be updated individually by every robot. For example, when a new tool is plugged into the system, it is physically located somewhere (e.g. in a depot). This information along with the one provided by the manufacturer (e.g. dimensions or technical specifications) is forwarded to all robots in the plant. The robots receive the semantic description, analyze and store it in their local knowledge base (ontology) in case it is relevant (e.g. if a gripper is not compatible with a robot, it discards that information). In addition, the factory manager can introduce a new product request (in terms of the number of parts to be produced in each time period) and optimization criteria (e.g. time and cost). Based on the semantic matching between its skills and the necessary tasks for each product, each robot can thus propose its possible configuration, process execution time and cost for every task.

The user sends product requests based on the assembly workflow described for each part to be produced. At this layer, the resources, processes and products are semantically specified and introduced into the ontology along with the particular factory description. The factory description includes constraints on individual interoperability of particular resources in a specific scenario (e.g. physical location of robots and other devices). Note that the factory is specific for a particular industrial setting while resource, process and product descriptions are developed and introduced by human experts and can be reused in different scenarios (resource description should be provided by a manufacturer). Furthermore, the semantic layer is also in charge of semantic matching of all the above descriptions with the product request by the user.

RDF is the chosen model for knowledge representation. RDF models are directed graphs (semantic networks), where nodes represent concepts or instances and arcs represents properties (or predicates). Semantic descriptions are given in standard ontology languages (e.g. RDF Schema, OWL), which extend RDF and are based on description logics. This technique models knowledge in terms of TBox (terminological box) and ABox (assertional box). In general, the TBox contains sentences describing general concepts (e.g. Robot Base, Tool, Gripper, etc.) and relations between them (e.g. Gripper is a Tool, Robot Base is compatible with Arm). The ABox contains sentences about concrete individuals and their relations (e.g. robot base FRI is compatible with Arm1, GL is a Gripper, etc.).
Fig. 3 shows a partial view of the ontology for our scenario (IRIs, used in RDF to represent resources, have been omitted in the examples). It comprises different types of descriptions as mentioned above. Process descriptions detail product manufacturing aspects, e.g. product Assembly1 has a fixed part Part1, a mobile part Part2 and a task. GlueTask (Assembly1 is produced by gluing Part1 and Part2, Part1 is fixed on the table while Part2 has to be manipulated by a robot).

Process descriptions include information about the tasks and the resources they need. For instance, gluing two pieces (GlueTask) has three tasks requiring other resources: OpenGripper requires a Gripper, UseCamera requires a Camera, and UseGlue requires GlueMaterial (note the difference to glue as a task). Factory setting descriptions describe features of the working environment, e.g. which resources are located in the same cell (and therefore they are reachable). All resources described in this example are assumed to be in the same environment. Finally, product request specifies what kind of products and quantity have to be produced (e.g. 3 products of type Assembly1).

Resources are robot bases, arms, grippers, cameras and glue materials. Each gripper-robot arm pair has a unitary production capacity. Each robot base is compatible with some specific arms, which can be combined with particular grippers.

A feasible connection between the available resources, processes, products and factory settings on the one hand, and requested products on the other is found by using a DL inference engine and SPARQL query language (obtaining compatible subsets of resources for feasible schedules, Fig. 1).

Semantic representations provide the means to easily obtain inferred knowledge. For instance, compatible resources with any of the available robot bases can be automatically obtained by a reasoner if the compatibleTool property is defined as an OWL transitive property (if robot base R1 is compatible with arm Arm1, and Arm1 is compatible with gripper G1, then R1 is compatible with G1). For more complex reasoning tasks we use rules on top of our OWL ontologies, which typically include new inferred knowledge into the A-Box. We include rules to infer e.g., compatible grippers with a product part, compatibility between robot bases and arms, arms and grippers, reachable object/resource by a robot, reachability between robots (so they can cooperate), and potentially reachable resource by a robot.

From all that descriptions and using DL reasoning and rules as explained before, the semantic layer produces all possible combinations of compatible resources for that operation (e.g. Arm1 is compatible with G1, G2 and G3).

We use SPARQL queries to obtain resource combinations for given tasks. For instance, to glue two parts a robot base, arm, gripper, camera and glue material are needed. A query finds (i) a gripper compatible with the mobile part, (ii) an arm compatible with that gripper, (iii) a glue material, (iv) a camera, and (v) a robot base compatible with the arm and able to reach (range of reach) all those resources. Note that most knowledge needed to answer such a query has been included by the inference engine.

For gluing the mobile part with dimension 2, and weight 10, the semantic layer produces all possible compatible combinations for that operation. Some of the results (out of 29 combinations) for our example are shown in Table I. It also includes another task consisting in attaching two parts.

<table>
<thead>
<tr>
<th>Task</th>
<th>Part</th>
<th>Robot</th>
<th>Arm</th>
<th>Gripper</th>
<th>Camera</th>
<th>Material</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attach1</td>
<td>Part3</td>
<td>FR1</td>
<td>Arm1</td>
<td>G1</td>
<td>Cam1</td>
<td></td>
</tr>
<tr>
<td>Attach2</td>
<td>Part3</td>
<td>FR1</td>
<td>Arm1</td>
<td>G1</td>
<td>Cam2</td>
<td></td>
</tr>
<tr>
<td>Glue</td>
<td>Part2</td>
<td>FR2</td>
<td>Arm2</td>
<td>G1</td>
<td>Cam2</td>
<td>Glue1</td>
</tr>
</tbody>
</table>

If there is at least one available production resource combination (machine + tool) for every requested product, the system proceeds to scheduling.

B. Scheduling Layer

At this layer, the best combination of compatible subsets of robots with necessary production resources (tools, grippers, etc.) for producing product requests is computed for each time period through a modified version of distributed scheduling optimization approach described in [13] based on the minimization of total assembly cost considering resource combinations obtained from the Semantic layer. Since the before mentioned paper deals with identical production resources, we adapt it to the heterogeneous ones. The modification is based on the categorization of resources’ compatibilities where each product has different combinations of compatible sets of resources adequate for its production.

Each robot agent (decision maker) finds its feasible local configuration considering its local objective and constraints, communicates relevant information and negotiates with other robot agents to reach a globally satisfactory solution in respect to the limited amount of available production resources.

We assume that the scheduling optimization process is supervised by product demands’ coordinator agents, which can
be arbitrary preassigned robot agents that for each product assign
the robots to the necessary tasks for each time period on the basis
of the product requests, guaranteeing the fulfillment of the
constraint on the limited resources’ amount. In general, the
demand for the resources is higher than their available quantity
and, hence, a negotiation process must be implemented among
the product demand agents to come up with a set of local
solutions that together satisfy also constraints on global
limitedness of the resources. The factory manager distributes
product assembly scheduling coordinator roles among available
robots. The assignment of scheduling coordinator roles to
robots can be performed in a semi-automatic way through heuristic
methods, but in this paper we do not focus on this aspect.

Since resources are limited, products compete for resource
sets obtained from the Semantic layer over the iterative auction
sending bids for all time periods to the resource owner. The
objective of resource owner is to diminish the resource conflicts
in every time period. Products, on the other hand, give their bids
based on the local cost minimization with updated prices.
Resource owner updates resource prices and performs auction
iterations until balance between demand and offer is obtained
within some threshold. The negotiation among the products is
modeled by an iterative auction process controlled by the
coordinator, which is in charge of coordination of the
negotiation. At each iteration:

Step 1 The coordinator (auctioneer) communicates to all the
product agents (bidders) current prices of the available robot
combinations in the T time periods;

Step 2 Each product agent, based on the local utility function of
its local objective, the constraints, and the current robot
combination prices, determines for each of its tasks the robot
requests (a bid) for the T time periods maximizing its utility
function and communicates its robot combinations requests to
the coordinator;

Step 3 Based on the bids received from the product agents, the
coordinator allocates the robot configurations to the products for
each time period, maximizing the robots’ total utility function;

Step 4 In order to reduce possible conflicts among products
generated by their bids (i.e., the violation of robot capacities), or
to stimulate usage of the robots, the coordinator updates the
robot combinations’ prices.

The iteration process repeats until a certain (halting) condition
is reached (e.g., a maximum number of iterations have
been performed, or the best robot allocation found is sufficiently
good from the products’ agents point of view). At the end, the
best robot allocation is retrieved.

Let \( B_i = \{ n_i(k) = \sum_n n_{ik} : k = 1, ..., T \} \) be the bid of product agent
\( i \), where \( n_{ik} \) is the number of robot configurations to carry out
task \( k \) which is necessary to make product \( i \). Furthermore, let \( A \)
\( = \{ \Lambda a,k : k = 1, ..., T \} \) be the (current) set of the resource prices
fixed by the coordinator (each one for each time period). The
utility function \( U(\Lambda a,k) \) of agent \( i \) is the opposite of the total
production cost during the planning time horizon:

\[
U(\Lambda a,k) = -z(\Lambda a,k) - p(\Lambda a,k).
\]

In (1), \( z(\Lambda a,k) \) is the minimum production cost for task \( k \) with
fixed numbers of assigned resources \( n_i(k) \), with \( k = 1, ..., T \),
according to bid \( B_i \) and \( p(\Lambda a,k) = \sum_\Lambda n_{ik} \lambda(k) n_i(k) \) is the
(additional) robot cost. Note that \( z(\Lambda a,k) \) is the optimal solution of
the local optimization problem \( P \) with fixed values of \( n_i(k) \).

In Step 2, agent \( i \) finds the best bid \( B_i^* = \{ n_i^*(k) : k = 1, ..., T \} \),
i.e., the bid that maximizes its utility function \( U_i(\Lambda a,k) \). Let
\( U_i(\Lambda a,k) \) be its maximum value for a given set \( \Lambda a \) of resource
prices. Finding \( B_i^* \), and determining \( U_i(\Lambda a,k) \), corresponds to
solving problem \( P \) with \( \lambda(k) \) added to the resource hiring cost
\( \rho(\Lambda a,k) \) in the objective function \( P \).

In Step 3, the coordinator receives bid \( B_i^* \) from each product
agent \( i \), and assigns resources to tasks maximizing resources’
utility function taking into account the received bids. Denoting
with \( v(k) \) the number of resources assigned to task \( k \) in time
period \( k \), the resources utility function \( R(\Lambda a,k) \) is the total profit
obtained from the resource assignment, and, hence,

\[
R(\Lambda a,k) = \sum_\Lambda n_i(k) v(k) \quad \text{for } k = 1, ..., T.
\]

Since \( v(k) \) cannot be greater than resource request (bid)
\( n_i(k) \) of product agent \( i \) and \( \sum_\Lambda v(k) \) cannot be greater than \( N \),
for each \( k = 1, ..., T \), the maximization of \( R(\Lambda a,k) \) can be easily
obtained by (heuristically) assigning resources to tasks on the
basis of the values of \( n_i(k) \), for example according to non-
increasing resource requests \( n_i(k) \), and, following this task
order, assign \( v(k) = \min\{n_i(k), N(k)\} \) robots to task \( k \). The
resource assignment quality is evaluated through measuring the
social welfare \( w(\Lambda a,k) \) of the tasks related to a given resource
assignment \( \Lambda a,k \) as the (minimum)

\[
\sum_\Lambda v(k) \quad \text{for } k = 1, ..., T.
\]

In Step 4, the coordinator updates the resource prices \( \lambda(k) \),
with \( k = 1, ..., T \). This is done, considering the deviation \( \Delta \lambda(k) \)
\( \equiv \sum_\Lambda n_i(k) - N \) of the total number of requested resources
from the number \( N \) of available resources, and by increasing the
current value of \( \lambda(k) \) if \( \Delta \lambda(k) \) is positive (i.e., \( \Delta \lambda(k) < 0 \)) or
by decreasing it (at most to \( 0 \)) if \( \Delta \lambda(k) \) is negative (in this latter case \( \Delta \lambda(k) \) is the deficit of
robot requirements). The value of the increase (decrease) of \( \lambda(k) \)
should be a non-decreasing function of the excess (deficit).

C. Execution Layer

The main objective of this layer is to enable the MRS to
seamlessly perform tasks and adapt to unexpected events (e.g.,
break-down of an already present or installation of a new robot,
change in ordered products, assembly time delays, etc.) without
the interruption in the operation. The real-time performance is
controlled through a set of Key Performance Indicators (KPIs)
and the disturbances are identified.

The KPIs we choose for the evaluation of the system are
utility \( U \) and stability \( S \) e.g., [14]. We use utility function as the
difference between the value of the social welfare \( w(\Lambda a,k) \) of
the tasks related to a given robot assignment of the new schedule
after taking into account the real time events and the social welfare \( w(\nu) \) of the initial schedule.

\[
U = w(\nu)^{\text{new}} - w(\nu).
\]  

(4)

The stability measure is a function of the start and completion times before and after rescheduling.

\[
S = \Sigma^m \min \{ a((t_i - t_i') + |C_i - C_i'|), D_i,\}
\]

(5)

where \( D_i \) represents the maximum allowed disturbance which might occur due to movement of the job \( J_i \), \( t_i \) is the initial and \( t_i' \) new start time. Initial completion times are \( C_i \) and the new ones \( C_i' \). Moreover, \( a \) represents the cost of the destabilizing effect of the moved job per time unit [14].

In the case of disturbances, two different cases are considered: schedule repair and rescheduling. In the case of rescheduling, feedback to the Scheduling layer is used for the disturbances which do not relate to the availability of the resources (breakdowns or new resources added to the system), while Semantic layer is contacted when these events occur. This feedback provokes an update of ontologies so they can provide updated system information at all times. Note that the Semantic layer only considers functioning resources when calculating feasible schedules.

We use rescheduling strategies of schedule repair and complete rescheduling, e.g., [15]. In the case of minor changes in the schedule, we use the schedule repair. In then case the current schedule is locally adjusted thus saving CPU time and preserving the stability of the system. In the case of significant changes in the schedule, we use global system rescheduling. The latter is better in maintaining optimal solutions but it requires prohibitive computational time; using it too frequently can lead to increased production costs. However, in the case that a robot individually and locally does not have sufficient production resources to finish the assigned task(s) in the time required, it sends a call for help to the robot(s) with necessary skills (which can be inferred on the Semantic layer by reasoning with the ontologies). Robots can accept or decline the call for help with the explanation for the choice: list of available resources with their temporal and physical availability. This process can be carried out in chain, i.e. \( R_1 \) asks \( R_2 \) for help, which in turn asks for help \( R_3 \), and so on. In the worst case, all the system is affected.

The process continues until there is at least one robot configuration available for the actual demand. In the contrary, the execution layer informs the semantic layer of the lack of resources. The manager should consider recombining the equipment by relocating or adding new resources such that they respond better to the overall product demand.

V. CONCLUSIONS

In this paper, we presented a multi-agent based distributed and optimized robots configuration and scheduling system ORCAS with a use-case example. Integrated distributed semantic descriptions and scheduling endow robotic systems with higher autonomy and less down times. Using the knowledge representation through standardized ontology languages facilitates an easy reuse of resource, process, and product descriptions in other factory settings. This makes it easier for SMEs to incorporate additional functionalities in their factories at lower deployment and maintenance costs. Furthermore, distributed knowledge representation allows for scalable and efficient knowledge management.

In the future work, we plan to further develop the execution layer concentrating on the real-time heuristic methods to respond to on-line errors, production efficiency improvement, and other KPIs. Furthermore we plan to experiment with real-time delays due to unexpected events and evaluate the local vs. global system’s real-time performance.

REFERENCES