Constraint-based Scheduling for Closed-loop Production Control in RMSs

E. Carpanzano & A. Orlandini & A. Valente
ITIA-CNR
Italian National Research Council
Milan, Italy

A. Cesta & F. Marinò & R. Rasconi
ISTC-CNR
Italian National Research Council
Rome, Italy

Abstract

Reconfigurable manufacturing systems (RMS) are conceived to operate in dynamic production contexts often characterized by fluctuations in demand, discovery or invention of new technologies, changes in part geometry, variances in raw material requirements. With specific focus on the RMS production aspects, the scheduling problem implies the capability of developing plans that can be easily and efficiently adjusted and regenerated once a production or system change occurs. The authors present a constraint-based online scheduling controller for RMS whose main advantage is its capability of dynamically interpreting and adapting to production anomalies or system misbehavior by regenerating on-line a new schedule. The performance of the controller has been tested by running a set of closed-loop experiments based on a real-world industrial case study. Results demonstrate that automatically synthesizing plans and recovery actions positively contribute to ensure a higher production rate.

Introduction

Highly automated production systems are devised to efficiently operate in dynamic production environments, as they implement at various levels the capability to adapt or anticipate uncertainty in production requirements (Smith and Waterman 1981; Wiendahl et al. 2007). Generally, Reconfigurable Manufacturing Systems (RMS) are endowed with a set of reconfigurability enablers related either to the single system component (e.g., mechatronic device, spindle axes), or related to the entire production cell and the system layout; as a consequence, possible fluctuations of the production demand can be counteracted by implementing the required enablers. Differently from RMSs, in Focused Flexibility Manufacturing Systems (FFMS) the responsiveness towards the changes relies on the production evolution forecasting. On the basis of the predicted events, the production system is preliminarily endowed with the necessary degree of flexibility which is exploited at the moment the change occurs (Terkaj, Tolio, and Valente 2009; 2010).

A particularly interesting case concerns the integration of production and automation RMS layers, as failing to provide an efficient integration between the previous two modules may severely affect the system global performance (Valente and Carpanzano 2011; Carpanzano et al. 2011). A production schedule module designed for highly automated systems must be able to manage both exogenous (e.g. change of volumes or machining features) and endogenous events (e.g. machine failures or anomalous behavior). At the same time, it must close the loop with the automation dispatching module, which is responsible for mapping production tasks into the related automation tasks that are assigned to the devices, coherently to the scheduled production jobs sequences. Closing the loop between the two modules entails that the dispatching module continuously feeds back the current status to the production schedule module, which may decide to possibly modify the plan (Fig. 1).

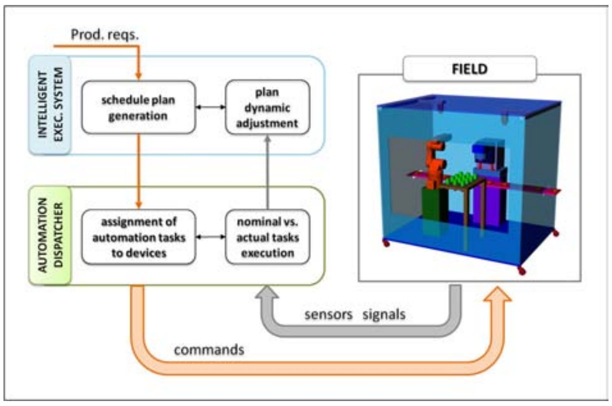

Figure 1: Production scheduler and automation dispatcher closed-loop.

There is a number of production scheduling approaches considering changes, both static (Tolio and Urgo 2007) and dynamic (Rasconi, Policella, and Cesta 2006). Another similar example of deployment of Planning & Scheduling techniques for on-line planning and execution in real-world domains can be found in (Ruml et al. 2011), where the authors tackle the problem of controlling production printing equipment by exploiting an on-line algorithm combining state-space planning and partial-order scheduling to synthesize plans. As opposed to (Ruml et al. 2011), the emphasis in the work presented here is more focused on the exploitation of the plan's temporal flexibility during the execution phase to hedge against the environmental uncertainty. More in detail, while in (Ruml et al. 2011) the main effort revolves around the on-line planning, makespan-optimization and dispatching of each new printing requests (goals) with plan abortion in case of a printer module failure, in our work great attention is devoted to the on-line plan readjustment in case exogenous events occur during execution. In our case, less effort is devoted to planning, as a determined sequence of tasks is provided for each different production request (i.e., there is no need to *plan*, in the classical sense); rather, we focus on the production plan AI-based scheduling followed by the on-line rescheduling and/or corrective temporal propagation, should disruptions make the plan resource-unfeasible at execution time.

With specific focus on the RMS management aspects, the production scheduling problem implies the capability to develop a short term production plan based on the inputs generated by the capacity planning problem that can be easily and efficiently adjusted and regenerated once a production or system change occurs. Despite the capability of generating robust and adaptive scheduling plans, the available approaches described above are decoupled by the system automation layer. The work addressed in this paper attempts to fill this gap, by merging the production and automation scheduling modules in a RMS context, and presenting the system resulting from this integration applied to a real industrial case. The paper is structured as follows: after presenting the proposed dynamic production scheduling approach, we analyze a particular case study taken from an industrial application; we then proceed to describe the formulation of the scheduling model, and finally we outline the major benefits of the approach, closing the paper with some final observations about the ongoing work.

The proposed approach

In (Carpanzano et al. 2011) we proposed to address the production scheduling problem using the Constraint Satisfaction Problem (CSP) formalism, as it allows to naturally express the features needed to model scheduling problems under uncertainty (Rasconi, Policella, and Cesta 2006) (e.g., it allows to easily provide the search algorithms with domain-specific heuristic, and to naturally represent *flexible* solutions). This characteristics provide the schedule with strong reconfiguration capabilities during execution, should potentially disrupting events occur. Synthesizing a production plan basically entails assigning the available resources to the jobs that are to be processed in the plant with a temporal horizon of the shift; once jobs are allocated to the resources, the schedule is passed to the automation layer that translates the production scheduling in automation plans.

Modeling the scheduling features

The base scheduling problem model employed in this work conforms to the *Resource Constrained Project Scheduling Problem with Time Lags* (RCPSP/max), this is to open the possibility to import a robust algorithmic experience on the problem (Cesta, Oddi, and Smith 2002; Rasconi, Policella, and Cesta 2006). The RCPSP/max can be formalized as follows: (**i**) a set V of n activities must be executed, where each activity a_j has a fixed duration d_j. Each activity has a start-time S_j and a completion-time C_j that satisfies the constraint $S_j + d_j = C_j$; (**ii**) a set E of temporal constraints exists between various activity pairs $< a_i, a_j >$ of the form $S_j - S_i \in [T_{ij}^{min}, T_{ij}^{max}]$, called start-to-start constraints (time lags or generalized precedence relations between activities); (**iii**) a set R of renewable resources are available, where each resource r_k has a integer capacity $c_k \geq 1$. The execution of an activity a_j requires capacity from one or more resources; for each resource r_k the integer $rc_{j,k}$ represents the required capacity (or size) of activity a_j. A schedule S is said to be *time-feasible* if all temporal constraints are satisfied, while it is *resource-feasible* if all resource constraints are satisfied (let $A(S,t) = i \in V | S_i \leq t < S_i + d_i$ be the set of activities which are in progress at time t and $r_k(S,t) = \sum_{j \in A(S,t)} rc_{j,k}$ the usage of resource r_k at that same time; for each t the constraint $r_k(S,t) \leq c_k$ must hold). The solving process is performed exploiting a makespan optimization scheduling algorithm called ISES (Iterative Sampling Earliest Solutions) (Cesta, Oddi, and Smith 2002) . The ISES solving algorithm basically proceeds by detecting the sets of schedule activities that compete for the same resource beyond the resource maximum capacity (*conflict sets*) and deciding the order of the activities in each set, through the insertion of further temporal constraints between the end time of one activity and the start time of the other, to eliminate conflicting overlaps.

The Dynamic Scheduling Control Architecture

In this work, we present a real-time control architecture (see Fig. 2) endowed with the flexible production scheduling capabilities discussed above in order to dynamically synthesize updated scheduling solutions as required by the continuously changing environmental conditions.

As shown in Fig. 2, the control architecture is designed to provide/receive data to/from the automation layer, and is composed of three different modules, each one holding different responsibilities. The *Controller* is the main component of the architecture and is in charge of: (**i**) invoking the Scheduler in order to ask for new solutions whenever a new job is entering the system (*find solution* command, see also the following point iv); (**ii**) updating the internal model of the system according to the observations received by the Dispatcher (*modify model* command); (**iii**) detecting any possible cause (e.g., anomalous behaviors, failures, etc.) leading to plan unfeasibility; (**iv**) invoking the Scheduler in order to reschedule the current solution and possibly produce a new feasible solution; (**v**) disposing completed tasks from the current model. Whenever invoked by the Controller, the *Scheduler* is responsible for (**i**) producing the initial solution needed to initiate the production process starting from a given problem, and (**ii**) rescheduling the current solution when it becomes unfeasible due to the onset of some exogenous event. Finally, the *Dispatcher* is responsible for (**i**) realizing the communication from the automation level to the rest of the architecture (all messages coming from the field are pre-processed by the Dispatcher and the related data are forwarded to the Controller), and (**ii**) dispatching solution-related plan activation signals to the automation layer.

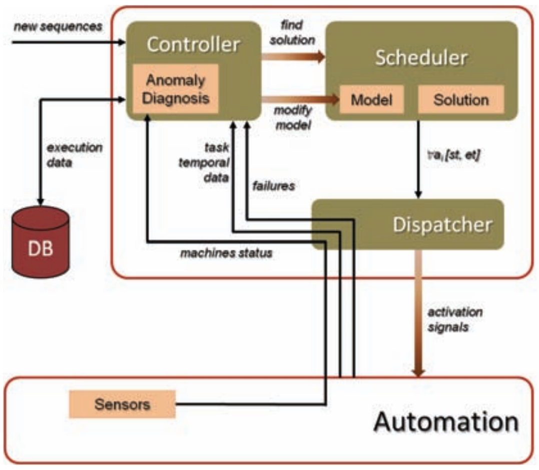

Figure 2: The Overall Control Architecture.

The overall architecture is implemented in Java as a composition of three concurrent and asynchronous processes that interact in a coordinated way to control the production process. In addition, one additional component has been implemented in order to record and store in a database the information flowing within the control system and to provide a human operator with a graphical view of the collected data. Finally, the communication between the control architecture and the automation level has been implemented through the use of the OPC protocol. According to the ISA95 standard, such protocol is fully compatible for SCADA connection.

Representing Maintenances and Recovery actions

In order to make the execution domain as close as possible to the real production system environments, besides the ordinary production tasks the system is able to accommodate *maintenance* activities (ordinary and extraordinary) as well as *recovery* actions that should be executed after a machine failure. Ordinary maintenances are generally scheduled in the plan according to their due frequency, extraordinary maintenances are scheduled in case of anomalous machine behaviors, while recovery actions are instead inserted in the plan on occurrence of particular machine failures. The urgency (i.e., the execution immediacy) of the extra-maintenance will be decided on the basis of the gravity of the occurred anomaly, which is assessed by the Controller's *Anomaly Diagnosis* module (see Fig. 2). It should be noted that as opposed to anomalies (which entail a degraded machine performance), we assume failures entail the complete inoperability of the affected resource until the failure is resolved (see Section *Production and management features of the FRC* for details related to the use case considered in this work).

Industrial case application

The proposed scheduling approach has been applied to an industrial case pertaining to a reconfigurable production line for the manufacturing of customized shoes, representing the European Best Practice in mass customization. The production system is composed by 5 manufacturing cells connected by a flexible transport system composed by rotating tables. The last automated manufacturing island in the shop-floor (Fig. 3) is the Finishing Robotic Cell (FRC), responsible for the shoe finishing before packaging and delivery.

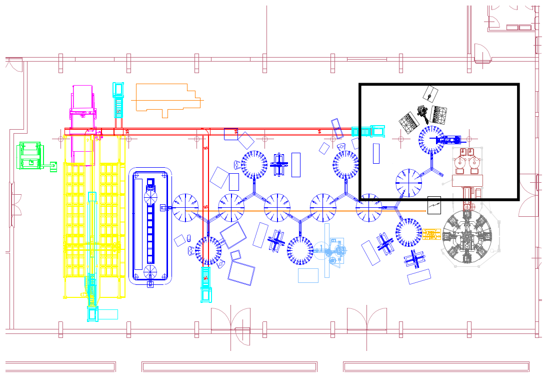

Figure 3: Shop-Floor Layout and Finishing Robotic Cell location.

As illustrated in Fig. 4, the FRC consists of four machine units, respectively an ABB robot (R1), the island from/to which parts are un/loaded (R2), a controlled brushing machine (R3), a creaming machine (R4) and a spraying machine (R5). The robot operates as *pick and place* and fixturing system; it loads the semi-finished shoe from the island (or rotary table) and, according to the part program, transports the part to the related machines, holding the part while the machine is processing it, as a proper fixturing system. Creaming and spraying machines are equipped with two inter-operational buffers with 9 slots each.

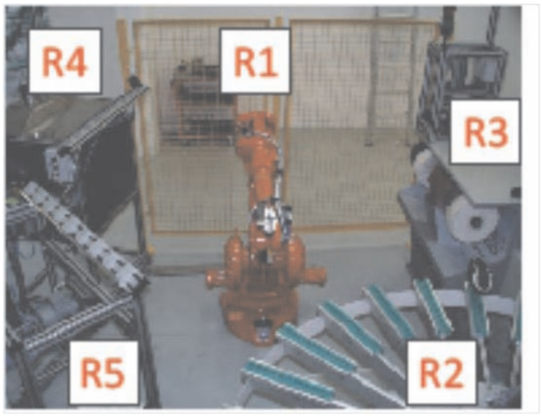

Figure 4: Resource composing the Finishing Robotic Cell.

As far as the FRC automated system is concerned, the FRC controller is connected with the transportation system PLC, the SCADA of the entire line and the low lever cell controller modules. Three types of activities are achieved by means of the existing control architecture: Communication-synchronization with production line controller; Synchronization of tasks in the finishing cell; Control of finishing operations such as rotation speed of the felt rollers, check of spray pressure and drying time, tracking of actual operation execution times compared to nominal expected ones.

Production and management features of the FRC

The FRC finishing process can be clustered in three main families: creaming processes, spraying processes and brushing processes. A typical process sequence is structured in the following steps: part loading; brushing for cleaning the raw piece of dust; finishing by spraying or creaming operations; drying in the buffer; brushing; unloading the finished part.

As highlighted in (Carpanzano et al. 2011), the considered family of products consists of 8 different part types (i.e., 4 woman models and 4 male models). The processing of each part is to be further divided into the left and right subparts of each shoe model. The production of all parts can be described in terms of the task sequences presented in Table 1. Given a specific shoe model, the left and right part of the model can be produced by means of the same sequence type for both female and male items. However, the durations of the sequence tasks can vary depending on the product type, resulting in 16 different process sequences in total.

Table 1: Description of operation sequences.

Sequence #1	Sequence #2	Sequence #3	Sequence #4
Load	Load	Load	Load
Brushing	Brushing	Spraying	Creaming
Spraying	Creaming	Unload	Unload
Unload	Unload	Buffering	Buffering
Buffering	Buffering	Load	Load
Load	Load	Brushing	Brushing
Brushing	Brushing	Unload	Unload
Unload	Unload		

As stated earlier, besides the production tasks a number of maintenance operations need to be foreseen and scheduled to ensure the FRC health. Table 2 synthesizes a few examples of maintenance tasks for FRC resources, considered in this work; in the table, the listed maintenance activities are associated to the related resource, and it is specified whether a stop of the cell is required. The table reports the average expected time (in seconds) for carrying out each maintenance activity as well as the maintenance rate indicated in brackets.

Table 2: Maintenance Operation Time matrix [sec].

Maint. Task [Rate]	R1	R2	R3	R4	R5	Stop	Fqncy
Fill cream tank					90	no	1/day
Creaming M. Clean.					60	no	12/day
Creaming M. Nozzle Clean.					3	no	2/hour
Fill spray tank				60		no	1/day
Spraying M. Clean.				60		no	12/day
Spraying M. Nozzle Clean.				3		no	2/hour
Fill wax in Brushing M.			60			no	1/day
Gripper Calibr.	15					no	1/day

Besides the maintenance tasks, a set of FRC failures have also been systemized and clustered by type in this work (see Table 3). Each failure type mapped upon resources is associated to a number of suitable troubleshooting strategies. An efficient execution of maintenance and/or recovery tasks relies on a persistent signal interpretation to assess the system status. This evaluation is crucial to identify the gap between actual and nominal system behavior and consequently the related actions to be implemented. Table 4 outlines few examples of signal information associated to the need to undertake specific maintenance tasks. For each considered machine maintenance, the table shows: (**i**) the polled sensors, and (**ii**) the predefined signal threshold values beyond which anomalies of different gravity are recognized (e.g., severe (*red*) anomalies are detected when the weighted sum of the anomalous readings obtained from sensors goes below 10%).

Table 3: Failure modes.

Fail. types	R3	R4	R5	Dur. (mins)	Cell Stop
Wax not moving	x			2	no
Brush slider not moving	x			2	no
Brush not rotating	x			2	no
Dosage not working			x	5,15	no,yes
Cream not arising from sponge			x	15,25	no,yes
Spray pistol not responding		x		10,20	no,yes
Air only from spray pistol		x		5	no
Anomalous spray pistol jet		x		10	no

Table 4: Maintenance tasks from signal interpreting.

Maintenance type	Sensors	Orange	Red
Fill cream tank	Level	10-20%	0-10%
Fill spray tank	Level	10-20%	0-10%
Fill wax in Brushing M.	Level	10-20%	0-10%
Gripper calibration	Force sensor	10-20%	0-10%
Creaming M. cleaning	Visual + filter + prod. qlty	15-30%	0-15%
Spraying M. cleaning	Visual + filter + prod. qlty	15-30%	0-15%
Creaming M. nozzle clean.	Cream cons. + valve + prod. qlty	15-30%	0-15%
Spraying M. nozzle clean.	Spray cons. + valve + prod. qlty	15-30%	0-15%

The scheduling-based controller

As explained in (Carpanzano et al. 2011), the FRC scheduling problem is modeled in CSP terms adopting a combination of modeling strategies that allows to capture all the significant aspects of the problem that the solving process must reason upon.

Modeling in the static case

The reader interested in the base model details can refer to (Carpanzano et al. 2011); in that work, we focused on a *model abstraction* suitable for the *static* problem solving case, which has allowed us to: (1) decrease the number of involved tasks guaranteeing no loss of expressiveness, and (2) re-use partially modified, if at all, off-the-shelf scheduling algorithms for the solving process.

The solution provided in (Carpanzano et al. 2011) was taking advantage of the robot acting as a *critical* resource, which allowed the two task subsequences immediately preceding and following the buffering operation to be grouped in two single blocks (the first and the third dashed boxes, in Fig. 5). In order to allow for a finer treatment of machine faults and maintenance operations, in the present work it is necessary to abandon such *aggregated* model and keep

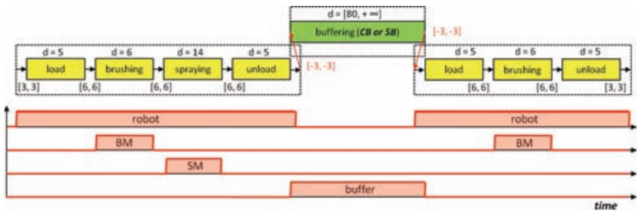

Figure 5: FRC task sequence for the woman #1 shoe part.

each individual sequence task separated. Fig. 5 depicts a typical sequence that entails the utilization of a subset of FRC machines and tools, e.g., the brushing machine and the spraying machine, as well as one of the two available buffers. Each sequence task is characterized by a nominal duration d, and consecutive tasks are separated by temporal constraints $[a, b]$ where a and b are the lower and the upper bound of the separation constraint. The actual constraint values depicted in Fig. 5 are consistent with the real robot transition times (e.g., the 6 value between the brushing and the spraying tasks represents the time that the robot takes to go from the brushing machine to the spraying machine passing through the *home* position), while the negative constraint values shown in red characterize the fact that the buffering operation actually starts 3 seconds *in advance* with respect to the end of the first dashed box, because the robot must however return to its home position before commencing any other action.

The Dynamic Model

Interleaving deliberation and execution in a smooth and effective way is a crucial issue for real time model-based control systems. In particular, integrating deliberative and reactive control is not a straightforward task and, then, suitable mechanisms are needed in order to guarantee a robust and continuous control.

In literature, several solutions have been proposed. For instance, in (Lemaitre and Verfaillie 2007), the authors propose a generic schema for the interaction between reactive and deliberative tasks where reactive and high-level reasoning control tasks are implemented and integrated so as to respectively meet a synchronous behavior assumption (i.e., in case of an exogenous event, a reactive task is always ready to be executed *before* any other event arrives), and an anytime behavior (i.e., a deliberative task is able to produce a first solution quickly, which can be improved later if time allows). Another approach is the one proposed in (Py, Rajan, and McGann 2010) where a hierarchy of reactors is exploited constituting several concurrent sense-plan-act control loops with different deliberation latencies. Both deliberative and reactive controls are implemented by means of, respectively, higher and lower latency reactors. In particular, reactors with small latencies are in charge to quickly react to unexpected events while reactors with long-term goals are managed by reactors with larger latencies.

In our case, given the chosen system latency and the FRC's characteristics, during the rescheduling phases the proposed control architecture is designed so as to (**i**) collect unexpected events (e.g., detected delays) that may occur during the rescheduling phases, and (**ii**) propagate such delays on the new solution generated for execution, by exploiting the solution's temporal flexibility. Such propagations/adjustments are guaranteed to be within the system latency by keeping the number of activities in the current schedule as low as possible, i.e., by eliminating the activities from the plan as they terminate their execution, in order to establish a sort of *dynamic equilibrium* between incoming and outgoing sequences, after an initial transient.

In order to allow the management of the schedule in a dynamic context (i.e., continuously absorbing all the modifications that pertain to the occurrence of exogenous events as well as to the simple passing of time) it has been necessary to extend the model presented in the previous section with online knowledge-capturing and management features. In our framework, such features are added using an asynchronous event-based model. All the information about the environmental uncertainty (e.g., endogenous and/or exogenous events) is organized through an asynchronous message exchange mechanism among the system modules. These messages convey all the information relatively to the deviations between the nominal schedule currently under execution and the *real data* coming from the automation side of the plant. The Controller (see Fig. 2) is in charge of acquiring such information, adapting the plan accordingly, and calling for the necessary rescheduling actions. In particular, a global rescheduling is performed each time a new sequence (i.e., a new production order) is inserted in the plan. However, applying a rescheduling to an executing plan generally presents the technical difficulty arising from the fact that the Scheduler does not have any internal chronological model of the schedule with respect to the passing of time. In other words, it has no knowledge of *past*, *present* and *future* relatively its own activities (i.e., it may decide to reschedule one activity into the past, or postpone the start time of an activity that has already started).

The latter issue is solved by introducing a number of constraint-based pre-processing procedures whose objective is to impose new constraints to the executing schedules prior to the solving process, so as to *force* the Scheduler to produce solutions that reflect the temporal reality of execution. Such procedures are the following: (**i**) *fixActivity()* when the Dispatcher acknowledges from the plant that an activity has started, the Controller must *fix* the activity's start time in the model, so that it is not shifted by the rescheduling process; (**ii**) *fixActivityDuration()* when the Dispatcher acknowledges from the plant that an activity has terminated, the Controller must fix the activity's end time, so that the latter is not modified by any possible rescheduling process before the activity is eliminated from the current plan; (**iii**) *disposeCompletedActivity()* this procedure eliminates a completed activity from the model; (**iv**) *prepareRescheduling()* this procedure performs the very important task of inserting in the plan a set of new *release constraints* relatively to all the activities that will participate to the rescheduling, so as to avoid that such activities will be scheduled in the past w.r.t. to the current execution time. Once all previous preparatory actions are performed, the rescheduling procedure can be safely called by the Controller. The Scheduler will therefore produce an alternative solution that (**i**) is temporally and resource feasible, (**ii**) satisfies all problem-related and execution-related constraints, and (**iii**) complies with the chronological physical requirements.

Experimental Results

In this section, we analyze the dynamic scheduling performances of our architecture by deploying it to control the execution of a series of typical production tasks relatively to the FRC case study. In particular, we will test the dynamic scheduling capabilities of our system by simulating the execution of a determined number of production sequences, which entails the online scheduling of the continuously incoming production tasks (equally distributed among the different process types) and ordinary maintenances (defined in Tab. 2). Both the temporal flexibility of the employed model and the rescheduling efficacy of the solver will be assessed by simulating the onset of perturbing events of random extent during each execution. More specifically, we analyze the performances of our architecture by varying the following settings: (**i**) we consider randomly variable start and end times for each incoming task, which affects the overall stability of the solution and requires the controller to continuously invoke the scheduler in order to adjust the current solution; (**ii**) we introduce a number of anomalies on the basis of the values (described in Tab. 4) detected by the automation layer sensors, and processed by the Diagnosis module. Each time an anomaly is detected, the control architecture reacts by scheduling an extraordinary maintenance activity whose urgency depends on the severity of the anomaly (*orange, red*). Maintenance activities may even cause the complete stop of the cell, and affect in any case the overall makespan; (**iii**) according to Tab. 3, we consider a set of possible failures for each machine, that may occur during execution. In this cases, the control architecture is in charge of scheduling the proper recovery task aimed at restoring full machine operability. As for anomalies, failures may introduce idle production periods, thus reducing production capability.

The experiments are organized in two different settings, both entailing the execution of 130 uniformly distributed production sequences. In the $1^{st}Setting$, 5 runs are executed for each resource R_i of the FRC. Each run requires the dynamic scheduling of the continuously incoming production tasks, including the periodic maintenances. Temporal uncertainty is introduced by considering an average 10% randomic misalignment between the nominal (i.e., dispatched) and the real (i.e., acknowledged) start/end times of the production activities. Each run is characterized by the onset of a number of anomalies and failures that depends on the affected machine R_i: in particular, every brushing machine will undergo 5 anomalies and 3 failures, every creaming machine will undergo 3 anomalies and 2 failures, and every spraying machine will undergo 3 anomalies and 3 failures (such numbers are decided on the basis of the available maintenance and recovery operations for each machine as well as of their durations, as per Tables 2 and 3). In order to appreciate the benefits of a controller that allows the concurrent scheduling and execution of both maintenance and production tasks, a second experimental setting is developed ($2^{nd}Setting$) where all previous runs are performed anew under the assumption that each maintenance and each failure recovery action entails a full FRC cell stop. All runs are performed on a MacBook Pro with a 64-bit Intel Core i5 CPU (2.4GHz) and 4GB RAM. In the following, we illustrate the collected empirical results.

Table 5 summarizes the obtained results; the table is horizontally organized so as to provide the data related to every machine. In particular, for each machine row the table lists data obtained in the first and second experimental settings (first and second row) together with the plain value difference and related percentage (third row). For each setting, the table provides the average values obtained from the five runs executed on each machine of: (**i**) the final makespan (i.e., the completion time of all 130 production sequences), (**ii**) the overall average time spent in reschedulings, (**iii**) the total number of reschedulings.

Table 5: Results from the experimental runs.

	MK (mins)	**Resched. T. (mins)**	**# of Resched.**
Brushing Machine			
$1^{st}Setting$	251	27	129
$2^{nd}Setting$	269	30	157
Δ (Δ %)	18 (7.2%)	3 (11.1%)	28 (21.7%)
Spraying Machine			
$1^{st}Setting$	256	26	130
$2^{nd}Setting$	279	28	155
Δ (Δ %)	23 (9%)	2 (7.7%)	25 (19.2%)
Creaming Machine			
$1^{st}Setting$	250	25	127
$2^{nd}Setting$	278	28	154
Δ (Δ %)	28 (11.2%)	3 (12%)	27 (21.2%)

The obtained results show the advantage of deploying an online reasoner that allows to continue execution during maintenances and recovery actions. Regardless of the machine involved in the performed runs, a significant reduction in makespan can be observed between the two experimental settings, meaning that the cell succeeds in executing all sequences in less time. In the table, makespan gains ranging from 18 up to 28 minutes are observable, which represent a significant improvement when measured against a total run time of 4 hours. Such gains are more evident for the machines that are characterized by longer maintenance and recovery actions (i.e., spraying and creaming). In case of long maintenances or recoveries, the capability to continue the execution of the tasks already scheduled on the unaffected machines is of great importance. Another interesting aspect can be observed by analyzing the higher number of reschedulings necessary in the $2^{nd}Setting$ w.r.t. to $1^{st}Setting$ runs; the reason of this stems from the fact that in order to simulate the absence of the execution controller ($2^{nd}Setting$ runs) we have modeled the cell-blocking condition by considering all maintenances and recoveries as tasks that require the whole cell; this causes a resource conflict that has to be solved by means of a rescheduling each time a maintenance or a recovery must be executed. As a last observation, the table also confirms that the chosen number of failures and anomalies injected during all runs for the different machines was well balanced, as the average total time spent for reschedulings is equally subdivided in all cases of the same type, despite the durations of the recoveries and maintenances varied significantly among the machines (see Tables 2 and 3), the reason being that the longer the recovery/maintenance operation, the higher the possibility of a rescheduling when it is added to the plan.

Conclusions

This work has presented an AI-based online scheduling controller capable of dynamically manage a production plan under execution in uncertain environmental conditions. The capabilities of the proposed scheduling controller have been tested with reference to a real-world industrial application case study. The series of closed-loop experimental tests concerning the execution of reality-inspired production plans (i.e., complete with regular maintenances, as well as random failures and anomalies), demonstrate that thanks to the adopted flexible model, the proposed controller enhances the current production system with the robustness necessary to face a subset of typical real-world production requirement evolutions. The current results confirm that the deployment of continuous rescheduling capabilities on a temporally flexible plan model positively contribute to the overall efficiency of the production plant, by allowing the execution of the planned number of jobs in less time. The authors work is currently ongoing with the further objectives of (**i**) improving the controller's rescheduling optimization capabilities in environments characterized by a higher number of tasks, and (**ii**) expanding the controller's uncertainty management capabilities to the whole actual set of FRC exogenous events, which represents a necessary step before commencing any experimentation on the real field.

Acknowlegments. The research presented in the current work has been partially funded under the Regional Project "CNR - Lombardy Region Agreement: Project 3". Cesta and Rasconi acknowledge the partial support of MIUR under the PRIN project 20089M932N (funds 2008).

References

Carpanzano, E.; Cesta, A.; Orlandini, A.; Rasconi, R.; and Valente, A. 2011. Closed-loop production and automation scheduling in RMSs. In *ETFA. International Conference on Emergent Technologies and Factory Automation*.

Cesta, A.; Oddi, A.; and Smith, S. 2002. A Constraint-based Method for Project Scheduling with Time Windows. *Journal of Heuristics* 8(1):109–136.

Lemaitre, M., and Verfaillie, G. 2007. Interaction between reactive and deliberative tasks for on-line decision-making. In *Proceedings of the ICAPS 3rd Workshop on Planning and Plan Execution for Real-World Systems*.

Py, F.; Rajan, K.; and McGann, C. 2010. A systematic agent framework for situated autonomous systems. In *AAMAS*, 583–590.

Rasconi, R.; Policella, N.; and Cesta, A. 2006. Fix the Schedule or Solve Again? Comparing Constraint-Based Approaches to Schedule Execution. In *COPLAS-06. Proceedings of the ICAPS Workshop on Constraint Satisfaction Techniques for Planning and Scheduling Problems*.

Ruml, W.; Do, M. B.; Zhou, R.; and Fromherz, M. P. J. 2011. On-line planning and scheduling: An application to controlling modular printers. *J. Artif. Intell. Res. (JAIR)* 40:415–468.

Smith, T., and Waterman, M. 1981. Identification of Common Molecular Subsequences. *J. Mol. Biol.* 147:195–197.

Terkaj, W.; Tolio, T.; and Valente, A. 2009. Design of Focused Flexibility Manufacturing Systems (FFMSs). *Design of Flexible Production Systems - Methodologies and Tools* 137–190.

Terkaj, W.; Tolio, T.; and Valente, A. 2010. A Stochastic Programming Approach to support the Machine Tool Builder in Designing Focused Flexibility Manufacturing Systems – FFMSs. *International Journal of Manufacturing Research* 5(2):199–229.

Tolio, T., and Urgo, M. 2007. A Rolling Horizon Approach to Plan Outsourcing in Manufacturing-to-Order Environments Affected by Uncertainty. *CIRP Annals – Manufacturing Technology* 56(1):487–490.

Valente, A., and Carpanzano, E. 2011. Development of multi-level adaptive control and scheduling solutions for shop-floor automation in Reconfigurable Manufacturing Systems. *CIRP Annals - Manufacturing Technology* 60(1):449–452.

Wiendahl, H.-P.; ElMaraghy, H.; Nyhuis, P.; Zah, M.; Wiendahl, H.-H.; Duffie, N.; and Brieke, M. 2007. Changeable Manufacturing - Classification, Design and Operation. *CIRP Annals - Manufacturing Technology* 56(2):783–809.