Cooperative Product Agents to Improve Manufacturing System Flexibility: A Model-Based Decision Framework

Ilya Kovalenko, *Member, IEEE*, Efe C. Balta, *Member, IEEE*, Dawn M. Tilbury, *Fellow, IEEE*, and Kira Barton, *Senior Member, IEEE*

***Abstract*— Due to the advancements in manufacturing system technology and the ever-increasing demand for personalized products, there is a growing desire to improve the flexibility of manufacturing systems. Multi-agent control is one strategy that has been proposed to address this challenge. The multi-agent control strategy relies on the decision making and cooperation of a number of intelligent software agents to control and coordinate various components on the shop floor. One of the most important agents for this control strategy is the product agent, which is the decision maker for a single part in the manufacturing system. To improve the flexibility and adaptability of the product agent and its control strategy, this work proposes a direct and active cooperation framework for the product agent. The directly and actively cooperating product agent can identify and actively negotiate scheduling constraints with other agents in the system. A new modeling formalism, based on priced timed automata, and an optimization-based decision making strategy are proposed as part of the framework. Two simulation case studies showcase how direct and active cooperation can be used to improve the flexibility and performance of manufacturing systems.**

***Note to Practitioners*—An intelligent product is a product in a manufacturing system that is able to make decisions based on a set of specifications and affect its own production process. Intelligent products have often been proposed to address the challenges associated with small-batch manufacturing and highly customized production. Specifically, by using intelligent products, manufacturers would be able to complete small orders without the need to reconfigure or reschedule operations in the manufacturing system. However, one of the major challenges in the implementation of this control strategy is the need to develop methods that allow intelligent products to cooperate with machines, robots, and other products in a manufacturing system. In this work, we propose a novel direct and active cooperation framework that allows intelligent products to communicate and cooperate with other resources and products on the shop floor. Using the proposed cooperation framework, intelligent products can resolve scheduling conflicts and work together to meet individual specifications (e.g., deadlines). Two case studies, a small job shop and a large semiconductor manufacturing system, showcase how the proposed cooperation framework can be leveraged for different types of applications.**

***Index Terms*— Cooperative systems, discrete-event systems, distributed control, dynamic scheduling, multi-agent systems, path planning, smart manufacturing.**

Manuscript received 6 December 2021; accepted 14 February 2022. Date of publication 17 March 2022; date of current version 6 January 2023. This article was recommended for publication by Associate Editor M. Franceschelli and Editor J. Li upon evaluation of the reviewers' comments. This work was supported in part by the National Science Foundation (NSF) Computer and Network Systems (CNS) under Grant 1544678, in part by the NSF Division of Graduate Education (DGE) under Grant 1256260, a gift from Rockwell Automation, and in part by the National Institute of Standards and Technology (NIST) under Award 70NANB19H090. *(Corresponding author: Ilya Kovalenko.)*

Ilya Kovalenko is with the Department of Mechanical Engineering and the Department of Industrial and Manufacturing Engineering, Pennsylvania State University, University Park, PA 16802 USA (e-mail: iqk5135@psu.edu).

Efe C. Balta is with the Automatic Control Laboratory, ETH Zürich, 8092 Zürich, Switzerland (e-mail: ebalta@ethz.ch).

Dawn M. Tilbury and Kira Barton are with the Department of Mechanical Engineering, University of Michigan, Ann Arbor, MI 48109 USA (e-mail: tilbury@umich.edu; bartonkl@umich.edu).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TASE.2022.3156384.

Digital Object Identifier 10.1109/TASE.2022.3156384

I. Introduction

SYSTEM-LEVEL control for the shop floor becomes increasingly more complicated as more advanced technology is integrated into the manufacturing system and the demand for personalized production increases [1]. Therefore, manufacturers are constantly looking to leverage different types of system-level control strategies to improve the flexibility of manufacturing systems and assist with the growing complexity of the manufacturing system. Previously, multi-agent approaches have been proposed as an effective method to improve the flexibility and performance of complex manufacturing systems [2], [3]. A multi-agent control strategy consists of a number of autonomous, cooperating intelligent agents that control various components on the plant floor [2], [3]. These agents use data from the system, information from other agents, and a set of individual goals to make decisions that ultimately determine the performance of the manufacturing system.

Product agents and resource agents are two important agents for the multi-agent control strategy [4]. A resource agent (RA) is a high-level controller for a resource (e.g. robot, machine) on the shop floor and a product agent (PA) makes decisions for a single part in the manufacturing system [5]. PAs cooperate with other PAs and RAs when making decisions. A more detailed explanation of the PAs, RAs, and the cooperation between the two types of agents is provided in Section II.

Frequently, a PA needs to cooperate with the other PAs and RAs to find a sequence of resource actions to complete its associated part's production requirements and not come into conflict with other agents in the system. Prior work has focused on cooperation between only PAs and RAs in the

TABLE I
NOMENCLATURE TABLE

General Terms	
PA	Product agent
RA	Resource agent
DA-PA	Direct, actively cooperative product agent
PTA	Priced Timed Automaton
Priced Timed Automaton Components	
$x \in X$	State x in the set of states X
pp_i	Physical property
Prp	Mapping of states to physical properties
E	Edges of a PTA
$\sigma \in \Sigma$	Event σ in the set of events Σ
$\mathbf{c} \in C$	Clock $\mathbf{c}$ in the space of clocks C
$\mathcal{K}$	Cost function
c_l, c_g	Local clock and global clock
$val(c^x_{\{l,g\}})$	Valuation of a clock (local/global)
I	Mapping of states to their invariants
Ag	Mapping of states, edges, and constraints to agents
X_m	Marked states
Constraint Formulation	
$t_{d,\tau}$	Delay transition of time τ
t_{s_e}	Discrete transition via the $e \in E$
$\mathcal{B}_b, \mathcal{B}_g$	Bound Constraint and Interval Gap Constraint
$t_{b,l,u}$	Time limit for (bound, lower/upper interval gap)
Product Agent Specification	
Dl	Deadline for a physical property
pm_i	Performance metric
α_p	Performance weight for property p
pr	Priority of DA-PA
Product Agent Models	
$\mathcal{A}_{EM}, \mathcal{A}_{DM}$	Environment and Decision Making models
$\mathcal{K}_{nm}, \mathcal{K}_{sft}$	Nominal cost and constraint violation cost
$\mathcal{K}^{nrg}$	Energy cost
$\gamma_j^{x_i} \in \Gamma$	j^{th} slack variable for state x_i
$\mathcal{H}, \mathcal{S}$	Hard and soft constraints
Product Agent Decision Making	
s	A path
$\phi(s, \xi)$	Solution sequence starting at $\xi = (x_i, c_i, \gamma_i)$
$\Pi_G(\phi)$	Projection of ϕ to a subspace $G \subset \mathcal{A}_{DM}$
$\mathcal{L}(\cdot)$	Language of an automata
$\#(\cdot)$	Number of nonzero elements

system (PA-RA cooperation) [5]–[13]. For the PAs in those architectures, the PA finds RAs that accomplish a specification in its process plan (see [14] for an overview of how the PA finds such RAs). After the PA finds the RAs, it schedules a set of material handling and manufacturing process operations necessary to accomplish its specification. This process is either done in a first-come, first-serve basis [5], [7]–[10], [12], [13] or by placing bids for various resources [6], [11] and does not authorize the PA to directly negotiate with other PAs over the resource utilization. This method of PA cooperation is passive, since the PA has to find a set of feasible resource actions without being able to negotiate with other PAs or RAs over existing scheduling constraints. Thus, the PA is not always able to find a sequence of actions that satisfies all of the existing scheduling constraints and accomplishes its production goals [13]. This limitation of the passive cooperation approach reduces the flexibility of the PA and, in turn, reduces the flexibility and adaptability of the multi-agent control strategy.

The main contribution of this paper is a novel decision making approach that uses *direct and active* cooperation for PAs in a multi-agent controlled manufacturing system. The PAs that use the proposed direct and active cooperation approach are denoted as direct, active cooperation product agents (DA-PAs) henceforth. For this cooperation approach, a DA-PA explores the manufacturing environment and identifies any constraints that can be negotiated with other agents (termed *soft constraints* in this work). Using this knowledge of the environment, the DA-PA finds a desired set of resource actions and identifies any potential soft constraints that conflict with its individual goals. If conflicting soft constraints are identified, the DA-PA communicates and cooperates with the respective agents in the system to resolve these conflicts. The integration of the proposed direct and active cooperation can improve the flexibility and adaptability of PAs, allowing them to meet previously infeasible deadlines in the system.

The rest of the paper is organized as follows. Background information about multi-agent control and product agents is provided in Section II. Section III overviews the components of the DA-PA's knowledge base used for the cooperation framework. Section IV describes the framework used for a direct and actively cooperating product agent. Case studies for the cooperation framework are given in Section V. Section VI provides concluding remarks and future directions.

II. BACKGROUND

This section overviews general multi-agent control strategies for manufacturing systems, provides the state-of the art in the development of product agent models, defines existing PA cooperation strategies, and presents background information about the priced timed automaton (PTA) modeling formalism used as part of the DA-PA approach.

A. Multi-Agent Control for Manufacturing Systems

One of the major challenges in this area of system-level control is the development of control strategies that can rapidly adapt to unexpected disturbances on the shop floor. A system-level controller has to handle unexpected machine faults or breakdowns, respond to changing customer orders, and quickly integrate new machines into the shop floor, to name a few objectives [15]. One strategy for responding to these system disturbances is through the use of a centralized, hierarchical control architecture [1], [16]. For this control strategy, the disturbance is handled by a single controller that has access to all of the information in the manufacturing system. However, finding new schedules and coordinating all of the system components becomes more difficult as the complexity of the manufacturing system increases [9]. Therefore, distributed strategies for manufacturing system-level control have been proposed to increase the flexibility of the manufacturing system [2], [3].

The field of agent-based systems and multi-agent control has been developed over the past several decades to address challenges in various application domains, from multi-robot systems [17], [18] to power grids [19], [20]. In the area of manufacturing, agent-based technology has been

used at various levels of production, from the supply chain [21]–[23] to the factory floor [3], [24]. For system-level control of manufacturing systems, a wide variety of multi-agent architectures have been developed [2], [25], [26]. These system-level multi-agent architectures leverage the local decision-making of agents to improve the response of the manufacturing system if there are unplanned disturbances in the system [9], [14] or to increase the customization capabilities of the system [13].

Most of these architectures identify the roles and responsibilities of the different manufacturing system agents and develop the communication requirements for this control strategy. Therefore, as part of these architectures, a number of agents have been proposed to represent customer orders, parts in the manufacturing system, resources (e.g. robots and machines) on the shop floor, and other components [4]. However, for real-time system level control during production, a large majority of the architectures rely on the cooperation and decision making of two types of agents: product agents (PAs) and resource agents (RAs) [1], [4].

A resource agent (RA) is a high-level controller for a resource on the shop floor (see [5], [9], [27] for several examples of RAs). The goal of the RA is to safely and efficiently schedule and complete logistic or manufacturing tasks in the system. The RA captures the capabilities and current status of the associated resource by gathering data from the physical resource on the factory floor. The RA also sends high-level commands to the associated physical resource (e.g. pick up a specific part, start a particular manufacturing operation, etc.), communicates information about the resource to other agents, and cooperates with different agents to achieve its goals.

A product agent (PA) makes decisions for a single part in the manufacturing system [5]. The goal for the product agent (PA) is to schedule and request various operations from the system resources based on customer specifications. The PA receives information about the status of the physical part and the capabilities of the manufacturing system from the RAs. Based on the information received from the RAs and a set of production requirements [25], the PA makes a decision to: (1) obtain the capabilities and plans of other PAs and RAs in the system, (2) plan and schedule future resource actions, or (3) request the execution of a resource action. Figure 1 shows the general flow of information for product agents, resource agents, and the factory floor for a multi-agent control architecture framework.

Integrating intelligent PAs into manufacturing systems leads to several benefits to customers and manufacturers. From the perspective of the customer, the integration of intelligent PAs creates a more customer-centric approach to manufacturing [28]. Customers are able to tune the parameters of individual PAs to ensure that the produced parts meet all of their desired specifications. From the perspective of the manufacturer, intelligent PAs can enable more efficient reconfiguration and improve system flexibility as new manufacturing system schedules can be developed and implemented faster by enabling data analysis and control at the product level [29].

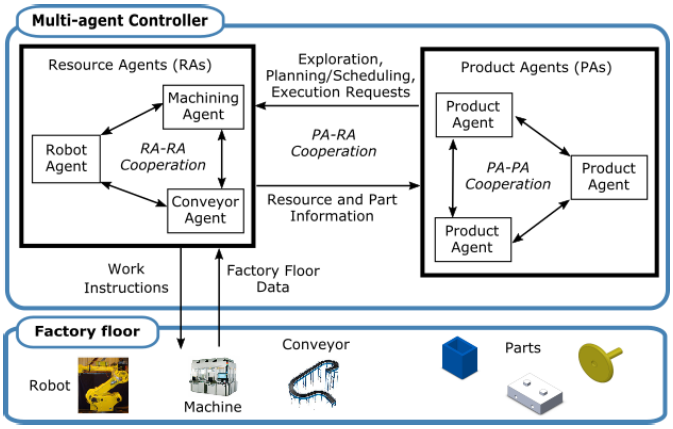


Fig. 1. An overview of the communication and cooperation between product agents, resource agents, and factory floor components. This figure extends the figure in [14] by highlighting the cooperation between the RAs and PAs. Note that each product agent makes decisions with the goal of creating desired products from raw-material or semi-processed parts.

B. Product Agent Cooperation

When planning and scheduling future resource actions, a PA must cooperate with both PAs and RAs to find a sequence of resource actions to complete its associated part's production requirements and not come into conflict with other agents in the system. To put existing PA cooperation strategies into context, the following definitions are provided for *direct*, *indirect*, *passive*, and *active cooperation*.

Using the definitions in [30], [31], we refer to *direct cooperation* between two agents as the utilization of a one-to-one communication link between the agents for the exchange of information. *Indirect cooperation* between two agents implies that agents pass information and cooperate with each other through other agents or through another controller. Using the general agent cooperation descriptions found in [32], [33], we define that a *passively cooperating* agent can only request actions that do not conflict with decisions from other agents. On the other hand, *actively cooperating* agents can identify conflicting actions and, through negotiation, find resolutions for these conflicts during decision making.

Product agent to resource agent (PA-RA) cooperation is often *direct* and *passive*. PAs directly communicate to the RAs in the system. However, during this direct communication, PAs can only request resource actions that fulfill the scheduling constraints provided by the RA. Examples of this type of cooperation can be found in most multi-agent frameworks for manufacturing systems [5], [8]–[10], [12]–[14], [34], [35].

Most of the product agent to product agent (PA-PA) cooperation is *indirect* and *passive* [8], [9], [13], [14], [35]–[38]. In these examples, a PA captures the plans of other PAs in the system through information provided by another agent. For example, when queried by a PA, RAs provide other PA schedules as scheduling constraints that cannot be violated. These types of multi-agent architectures are usually first-come-first-serve for the PAs in the system. Therefore, there is never direct communication between PAs in the system and there is no negotiation over the scheduling constraints.

Direct and *active* cooperation would allow a PA to negotiate with other agents when the PA identifies any conflicting events. This type of cooperation provides increased flexibility for the PA, allowing the agent to more effectively achieve its goals and react to unexpected disturbances.

C. Model-Based Product Agents

To enable *direct* and *active* PA cooperation, it is necessary to develop and improve the intelligence of the product agent. In a number of multi-agent architectures, PAs use rule-based reasoning to make planning and scheduling decisions in a manufacturing system [5], [39]–[41]. However, it is difficult to scale these rule-based reasoners to more complex manufacturing systems, where a PA has to cooperate with various different PAs and RAs in the system.

Recent work has looked at developing model-based reasoning to develop the decision making of PAs [12], [13], [38], [42], [43]. Model-based PAs use a model of the system and optimization techniques to drive the decision making of the PA [13]. These model-based PAs store and update a model of the manufacturing environment in their knowledge base [27], [44]. Using this model of the manufacturing environment and a set of goals provided during initialization, the PAs autonomously run optimization algorithms to decide which actions to schedule with the RAs in the system.

Finite state machines (FSMs) [45] are frequently used to encode the capabilities of the manufacturing systems in model-based PAs [12], [13], [42], [43], [46]. For the FSMs used in model-based PAs, the states represent potential locations and completed manufacturing operations (i.e. new features [47]) for the part in the manufacturing system. The events in the FSMs represent resource actions (e.g. logistic or manufacturing operations) that allow the part to transition between the states in the system.

Other models that have been used to encode manufacturing system capabilities include Markov Decision Process (MDP) models [38], [42], [43], [48] and Petri nets [9], [49]. The MDP models used by PAs are extensions to PA FSM models that also capture some of the stochastic elements of manufacturing systems, e.g., processing time [38], [42], [43], [48]. Petri nets models have also been used for decision making by PAs [9], [49]. The tokens in these Petri net models were used to track the availability of raw material and to ensure that another agent (a task agent) completes a requested task [49]. Note that it is possible to create FSM models that would be equivalent to the Petri net models used by the PAs in [49] and [9].

All of these models encode the scheduling constraints of the PA as *hard constraints*, i.e. these constraints cannot be violated at any point by the PA. However, these scheduling constraints in a manufacturing system are flexible in certain cases and can be relaxed through negotiation with other PAs and RAs in the system. Current model-based PAs do not capture the negotiable actions, i.e. *soft constraints,* in their planning and scheduling models.

D. Priced Timed Automata

This work extends existing model-based PAs for capturing the constraints that can be negotiated by the PAs, i.e., the *soft constraints*. Since most of the models used by existing PAs are either FSMs or can be stated as FSMs, we will extend the FSMs modeling formalism to capture the *soft constraints* in the system. Specifically, we propose to leverage the priced timed automaton (PTA) modeling formalism [50], [51] to extend existing PA models and capture the PA's scheduling constraints.

Here we provide a formal definition of PTA, based on [51]. A PTA is composed of a set of states, X, connected by a set of edges, E. Each edge is labeled with an event, $\sigma \in \Sigma$. A PTA also has a set of clocks, continuous variables $\mathbf{c} \in C = \mathbb{R}^{n_c}_{\geq 0}$, where n_c is the number of clocks in the system. All clocks have the same constant, positive growth rate and an initial value of zero.

A finite set of Boolean indicator functions of clock values, $\mathcal{B}$, represents the constraints. For example, for a clock value $\mathbf{c} \in C$, the Boolean indicator function $\mathcal{I}_{\mathbf{c}^i \leq a}(\mathbf{c}) \in \mathcal{B}$ evaluates to *true* if and only if the i^{th} element of $\mathbf{c}$ is less than or equal to a. Each element of $\mathcal{B}$ is either an invariant for a location, which must evaluate to true for the system to occupy the location, or a guard on an edge, which must evaluate to true to traverse an edge. Reset maps on edges set clocks to predetermined values when the edge is traversed.

Additionally, a PTA has costs on states and edges. Costs on edges are discrete increments added to the total running cost when the edge is traversed, while the states have cost rates, and steadily accumulate cost over time.

Definition 1 (Priced Timed Automata): A PTA $\mathcal{A}$ is defined as an 8-tuple $\mathcal{A} = (X, C, \Sigma, E, I, R, P, x_0)$, where

- $X = \{x_0, x_1, \ldots, x_{n^x}\}$ *is a finite set of states of the PTA*
- $C = c^0 \times c^1 \times \ldots \times c^{n_c} = \mathbb{R}^{n_c}_{\geq 0}$ *is the clock state space*
- $\Sigma = \{\sigma_0, \sigma_1, \ldots, \sigma_{n_\sigma}\}$ *is a finite set of events*
- $E \subseteq X \times \mathcal{B} \times \Sigma \times X$ *is a finite set of edges*
- $I : X \rightarrow \mathcal{B}$ *is the invariant operator*
- $R : E \times C \rightarrow C$ *is a reset operator*
- $\mathcal{K} : X \cup E \rightarrow [0, \infty)$ *maps the locations and edges to their costs*
- $x_0 \in X$ *is the initial state*

The guards are embedded in the edge definition, with $\mathcal{B}$ denoting constraints on the clocks for each edge, i.e., the guard conditions. A transition is a formal description of a change in the system state and the transition type. There are two types of transitions, discrete transitions (changes in location) and delay transitions (changes in clock values). Let $t_{y,z}$ denote a transition with $y \in \{s, d\}$ denoting if the transition is a discrete (s) or a delay (d) transition, and $z = \{e, \tau\}$ where $e \in E$ and $\tau \in \mathbb{R}$. We notate transitions with $x \xrightarrow{t_{y,e}} x'$, where $x, x' \in X$. We provide further definitions on the clock structure before formally defining the two types of transitions.

We utilize a two-clock structure in this work. The two-clock state space is given as $C = c_l \times c_g$ through the rest of the paper, noting that additional clocks may be added for various applications. The local clock, c_l, represents the time spent at a single state and, thus, is always reset to 0 when entering a state via a transition in the system. The global clock, c_g, represents the absolute time for the system and is never reset in the system. Additionally we use the valuation operator $val(\cdot)$ to denote the value of a clock. The valuation operator captures

the values of clocks for a state $x \in X$ at the time when a discrete transition corresponding to an edge $e \in E$ is *taken* from the state x to a new state x'. For example $val(c_g^x)$ denotes the value of the global clock at state x at the time of transition to a new state x' by taking an edge. Then, the two types of transitions for PTA are defined as follows.

Definition 2 (Discrete Transition): A transition $x \xrightarrow{t_{s,e}} x'$ *is a discrete transition if* $e = (x, \sigma, x') \in E$, *the guard is satisfied* $\mathbf{c}_i \models g$, *and clock valuations are incremented as* $val(c_g^{x'}) := val(c_g^x)$ *and* $val(c_l^{x'}) := 0$. *Note that we say a discrete transition is "taken" when an edge in the model is traversed.*

Definition 3 (Delay Transitions): A transition $x \xrightarrow{t_{d,\tau}} x'$ *is a delay transition if* $x = x'$, *the invariant* $I(x)$ *is true, and clock valuations are incremented as* $val(c_g^{x'}) := val(c_g^x) + \tau$ *and* $val(c_l^{x'}) := \tau$.

Note that by explicitly defining the two transition types with the specific clock valuation updates, we implicitly defined a reset operator, R, for the PTA. We omit R in later definitions for brevity.

III. DA-PA Knowledge Base

This section provides a description of the direct and actively cooperating product agent's (DA-PA's) knowledge base (i.e., the goals, environment model, and decision making model).

A. Goals

During initialization, a direct, actively cooperative product agent (DA-PA) is provided a process plan, exit plan, performance weights, and the priority of the associated part with respect to other parts in the system.

1) Process Plan: The process plan is formulated based on a customer order and consists of an ordered list of desired physical properties for the associated part with deadlines [25]. This ordered list representation of the process plan captures the precedence constraints of product features and is often used to specify the required processes for a part in the manufacturing system [47]. Physical properties include part locations or part features, e.g. "part at exit loading dock" or "part has hole with taper" [13], [47], [52]. Formally, the process plan is defined as: $Plan = (PP_d, Dl)$, where $PP_d = (pp_1, pp_2, \ldots)$ is the ordered list of desired physical properties pp_i, and $Dl : PP_d \rightarrow \mathbb{R}_{\geq 0}$ is the latest allowable finish time for a property.

The process plan is developed offline and provided to the DA-PA (e.g. [53]). Since the plan is computed offline and, additionally, to ensure that communication between agents is kept in a local neighborhood [14], we assume that the DA-PA accomplishes the process plan one physical property at a time. Therefore, the DA-PA will always look to complete the next unfinished property in the process plan. Note that the DA-PA keeps track of the completed physical properties and can identify the next unfinished physical property when required.

Future work will focus on encoding the process plan in other modeling formalisms, such as finite state machines and priced timed automata. Encoding the process plan as an automata will improve the expressiveness of the process plan, allow greater product personalization, and enhance the DA-PA decision making flexibility. For example, an automata representation would allow for encoding partial ordering of production steps. Consider three processes $p1$, $p2$, and $p3$. By defining appropriate guards and invariants, we can specify that both $p1$ and $p2$ to be completed before $p3$, but no strict specification on the ordering between $p1$ and $p2$.

2) Exit Plan: The DA-PA is initialized with an exit plan that is used if the DA-PA cannot find a sequence of resource actions to fulfill the process plan [13]. The DA-PA can exit the manufacturing system by calling an exit agent in the exit plan. For example, an exit agent can be a human agent [54] who can retrieve the part from the manufacturing system. The decision of the DA-PA to exit the system is discussed in detail in Section IV.

3) Performance Weights: There are various, measurable performance metrics, $PM = \{pm_1, pm_2, \ldots, pm_n\}$, for resources on the plant floor, e.g. energy and material cost. These metrics are tracked and stored by the associated RAs in the system. The RAs share these performance metrics when they communicate their capabilities with the DA-PA. These metrics are stored in the environment model of the DA-PA. A formal description of their encoding in the environment model is provided in Section III-B.

A set of performance weights is provided during the DA-PA's initialization. These performance weights are computed offline based on the customer order or the manufacturer requirements. Each performance weight corresponds to a metric. Formally, the set of performance metrics is defined as $\{\alpha_{p_i} \in \mathbb{R}_{\geq 0} \mid p_i \in PM\}$. The magnitude of the weight represents the relative importance of the corresponding metric for the DA-PA. For example, if a DA-PA favors minimizing material cost over energy usage, the following relation will hold $\alpha_{material} > \alpha_{energy}$. The DA-PA uses the performance weights and metrics during its decision making to identify desirable resources and resource actions in the system.

4) Agent Priority: The DA-PA is provided an importance value, $pr \in \mathbb{N}$, which represents the priority of the associated physical part when compared to other parts in the system. An importance value of 1 signifies that the part is of the highest priority. Therefore, a part i has a higher priority than part j if $pr_i < pr_j$.

Note that while the process plan, performance weights, and agent priority are all part of the DA-PA's individual goal, they are linked to the overall performance of the manufacturing system. The process plan is created based on customer and manufacturer specification (including order of desired physical properties and deadline) [25], [53]. The performance weights can be set by the manufacturer to prioritize certain actions for the DA-PA (e.g. use less energy or material). Finally, the agent priority can be set by the needs of the manufacturer and based on the cost-benefit of the customer order. For example, if a customer sends in a highly profitable order with harsh penalties for production delays, the DA-PAs associated with the order would be given a high priority.

Future work will look at developing effective methods to integrate higher-level controllers, such as a Manufacturing Execution System (MES), with DA-PA performance weights and agent priority parameters. Since these high-level controllers have access to a global view of the system, they would be able to supervise the DA-PAs and tune the performance weights and priorities to formally guarantee that the behavior of the DA-PAs meet the system-level manufacturing requirements.

B. Environment Model

The capabilities and scheduling constraints of the local manufacturing environment are captured by the DA-PA's environment model [13], [55]. The DA-PA updates its environment model as other agents (RAs and PAs) provide the capabilities and scheduling constraints, as described in Section IV-A. The environment model is defined as the following tuple:

$\mathcal{A}_{EM} = (X, Prp, x_0, \Sigma, E, C, \Gamma, I, \mathcal{K}_{nm}, \mathcal{K}_{sft}, Ag)$:

$X = \{x_1, x_2, \ldots\}$ is the set of states for the associated physical part

$Prp : X \rightarrow PP$ maps states to physical properties

$x_0 \in X$ is the current state of the part

$\Sigma = \{\sigma_0, \sigma_1, \ldots\}$ is a finite set of events, where each event represents the start or completion of a resource action (e.g., a logistic or manufacturing operation)

$E \subseteq Q \times \Sigma \times Q$ is a finite set of edges

$C = \{c_l, c_g\}$ are the local and the global clocks

$\Gamma : \{\ldots, \gamma_1^{x_i}, \gamma_2^{x_i}, \ldots, \gamma_1^{x_{i+n}}, \ldots\}$ is the set of slack variables, where $\gamma_j^{x_i} \in \mathbb{R}$ is the j^{th} slack variable for state x_i.

$I : X \rightarrow \mathcal{B}[val(C), val(\Gamma)]$ maps states to their constraints as a function of the clock and slack variables

$Ag : X \cup E \cup \mathcal{B}[val(C), val(\Gamma)] \rightarrow Agents$ maps states, edges, and constraints to corresponding agents

$\mathcal{K}_{nm} : X \times PM \times val(C) \rightarrow \mathbb{R}_{\geq 0}$ maps states, performance metrics, and valuations of clocks to nominal cost values

$\mathcal{K}_{sft} : X \times PM \times val(\Gamma) \rightarrow \mathbb{R}_{\geq 0}$ maps states, performance metrics, and valuations of slacks to cost-of-constraint-violation values

where X, Prp, x_0, E are the discrete event dynamics, C, Γ, I encode the time-based state constraints, Ag is the agent association function, and $\mathcal{K}_{nm}, \mathcal{K}_{sft}$ are the state costs for the DA-PA. The vectors $val(C)$ and $val(\Gamma)$ are valuations of the variables C and Γ for each state in the system. These valuations capture the values of the clocks and slack constraints for a state $x \in X$ at the time when a discrete transition corresponding to an edge $e \in E$ is taken *from* state x *to* another state. To ensure unique clock and slack valuations, we enforce the $\mathcal{A}_{EM}$ to be acyclic, while noting that cyclic graphs can be "flattened" into acyclic graphs efficiently [56].

$\mathcal{A}_{EM}$ is an extension of the PTA modeling formalism presented in Section II-D. A description of how the manufacturing environment is mapped to each component is discussed in the rest of this section. Note that there are several differences when $\mathcal{A}_{EM}$ is compared to the PTA from Definition 1. As previously discussed, the reset operator, R, is left out for brevity. Only states (not edges) of $\mathcal{A}_{EM}$ have constraints and these state constraints are encoded in the invariant, I. Finally, there are a number of extensions to the constraints and costs of $\mathcal{A}_{EM}$, which are discussed here.

1) Discrete Event Dynamics: X, Prp, x_0, E represent the capabilities of the manufacturing system. X are the states of the physical part in the system. Each state is a combination of several physical properties of the part. Physical properties are locations and physical composition of the part (e.g. "part at machine" or "part feature completed") [52] or operations performed on the part (e.g. "moving the part" or "working on a manufacturing process"). Prp maps each state of the environment model to one or more of these physical properties. A current state, x_0, is updated when an RA informs the DA-PA that it has started or finished a resource action. Each event, Σ, represents an instantaneous (with delay 0) start or completion of a manufacturing or logistic operation [57].

2) Time-Based Constraints: The state constraints, $\mathcal{B}$, limit the amount of time that the associated part can be in the state (e.g. how long the part can stay at a location). These constraints are split into hard and soft constraints, $\mathcal{B} = \mathcal{H}[val(C)] \cup \mathcal{S}[val(C), val(\Gamma)]$. Hard constraints, $\mathcal{H}[val(C)]$, cannot be violated by the DA-PA (e.g. minimum time required to complete a manufacturing operation). Soft constraints, $\mathcal{S}[val(C), val(\Gamma)]$, contain slack variables, Γ. As described in Section IV-C, these soft constraints may be violated by the DA-PA through negotiation with other agents in the system. Note that there is no limit for the number of constraints at each state. Therefore, all of the constraints in the model are stored in the invariant mapping, I. This mapping links a state to all of its constraints.

All time constraints, $\mathcal{B}$, are logic expressions [58]. There are two types of constraints used in the environment model: *bound constraint* and *interval gap constraint*. The bound constraint is a time limit that represents an absolute limit when the part can enter a state or leave a state. The gap constraint represents an interval when the part cannot be at a certain state. Formally, the two constraints are defined as follows:

Definition 4 (Bound Constraint): A bound constraint, $\mathcal{B}_b$, for state, x, is satisfied (i.e. evaluates to true) if and only if a clock valuation (either local or global) at the state, $val(c_b^x)$, for $c_b \in C$ satisfies:

$$val(c_b^x) \bowtie t_b \pm a_\gamma \, val(\gamma_b^x) \tag{1}$$

where $t_b \in \mathbb{R}_{\geq 0}$ is the time limit for the clock, $a_\gamma \in [0, 1]$ is the hardness coefficient, γ_b^x is a slack variable associated with the constraints on state x, and $\bowtie$ is one of the following logical operators: $<, \leq, =, \geq, >$.

Note that a bound constraint is a hard constraint if $a_\gamma = 0$.

Definition 5 (Interval Gap Constraint): An interval gap constraint, $\mathcal{B}_g$, for state x and interval (t_l, t_u) is satisfied if and only if the local and global clock valuations at the state, $val(c_l^x)$ and $val(c_g^x)$, satisfy:

$$Z_1 \vee Z_2 \tag{2a}$$

$$Z_1 := val(c_g^x) \leq t_l + a_{\gamma,l} \, val(\gamma_1^x) \tag{2b}$$

$$Z_2 := val(c_g^x) - val(c_l^x) > t_u + a_{\gamma,u} \, val(\gamma_u^x) \tag{2c}$$

where $t_l, t_u \in \mathbb{R}_{\geq 0}$ *and* $t_u > t_l$, $a_{\gamma_l}, a_{\gamma_u} \in [0, 1]$ *are the hardness coefficients,* ${\gamma_l}^x, {\gamma_u}^x$ *are unique slack variables associated with the constraint, and* $\neg, \wedge, \vee$ *are logic operators NOT, AND, OR, respectively.*

As defined previously in Section III-B, $val(c_g^x)$ indicates the global clock time when a transition is taken *from* state x to another state. Z_1 is satisfied when the transition from state x is taken before the lower bound, t_l. Additionally, c_l^x is reset to 0 after each transition (defined in Section II-D). Thus, $val(c_l^x)$ is the time *between* the transition to state x and the transition from state x, i.e., the amount of time the part is in state x. Therefore, $val(c_g^x) - val(c_l^x)$ represents the *global clock time* when a transition is taken *to* state x from another state. Z_2 is satisfied when the transition to state x is taken after the upper bound, t_u. $Z_1 \vee Z_2$ is satisfied when the transition is taken *from x before* the interval or a transition is taken *to x after* the interval, i.e., the part is not in the state at that interval.

Note that the constraints should encode some of the stochastic properties of the system. For example, for manufacturing operations, the maximum operating time (including any uncertainties) can used to create the bound constraint for the system. An example is provided in Section III-C.

3) Transitions: The objective of the DA-PA's decision making is to find clock and slack variable valuations that satisfy all of the state constraints in the system. To satisfy these constraints, the DA-PA must choose a sequence of *discrete* transitions from E that move the associated part from one state x to a new one x' and *delay* transitions that keep the part at the same state x but take a finite time $\tau \in \mathbb{R}$. The definitions of these two transitions were provided in Section II. Detail about how a DA-PA chooses these transitions is provided in Section IV-B.

Remark 1: It is always possible to denote delay and discrete transitions in alternating sequence since delay transitions are additive and we can define a zero time delay transition between two consecutive discrete transitions.

4) Agent Association Function: The DA-PA builds the environment model by requesting capabilities from the RAs in the system [10], [13]. If the RAs provide their capabilities as PTA-based models, the DA-PA can put together these models to create the proposed environment model [59]. However, to enable the proposed cooperation framework, the DA-PA must keep track of the agents associated with the components in the environment model.

The agent association function, Ag, maps states, edges, and constraints to an associated agent. The states can be mapped to one or more agents. If a state is mapped to more than one agent, it is a shared state [14]. The presence of these shared states allows for the modular composition of the environment model. The edges are mapped to a single agent so that the DA-PA can identify which agent to contact when requesting a discrete transition. Similarly, constraints are mapped to a single agent, representing which agent is responsible for the scheduling constraint in the system.

5) State Costs: The nominal cost, $\mathcal{K}_{nm}^{p_i}(x_j)$, for performance metric p_i is the cost for a part to stay at a state with respect to the corresponding metric (p_i). Formally, the nominal cost for performance metric p_i for a single state x_j is:

$$\mathcal{K}_{nm}^{p_i}(x_j) = A_{nm} \cdot val(C^x) + b_{nm} \tag{3}$$

where $A_{nm} \in \mathbb{R}_{\geq 0}^2$, $b_{nm} \in \mathbb{R}_{\geq 0}$, and $val(C^x) = [val(c_g^x), val(c_l^x)]^T$ are the valuations of the clocks for state x.

Similarly, the soft constraint violation cost, $\mathcal{K}_{sft}^{p_i}(x_j)$, is the cost for the part to stay at a state if there is a constraint violation. The soft constraint violation cost for performance metric p_i for a state is:

$$\mathcal{K}_{sft}^{p_i}(x_j) = \sum_{k=1}^{m} a_{sft}^k \, val({\gamma_k}^{x_j}) + b_{sft}^k \delta(val({\gamma_k}^{x_j})) \tag{4}$$

where $m = |I(x_j)|$ is the number of constraints for the state, ${\gamma_k}^{x_j}$ is the slack variable associated with the k^{th} constraint on state x_j, $a_{sft}^k, b_{sft}^k \in \mathbb{R}_{\geq 0}, \forall 1 \leq k \leq m$, and $\delta(\cdot)$ denotes an indicator operator with 1 if the argument is nonzero and 0 otherwise. Note that if $a_{sft}^k = 0$ and $b_{sft}^k = 0$, then there is no penalty for violating constraint k. Similarly, if the associated valuation of the slack variable is 0, then there is no violation cost added, i.e. if $val({\gamma_k}^{x_j}) = 0$, then $\delta(val({\gamma_k}^{x_j})) = 0$ and $val({\gamma_k}^{x_j}) + b_{sft}^k \delta(val({\gamma_k}^{x_j})) = 0$. Thus, the sum (4) is a summation of slack costs for the nonzero slack valuations.

For each state, RAs use their own internal data-driven and physics-based models to estimate the nominal cost parameters. PAs obtain and update A_{nm}, b_{nm} after querying RAs and obtaining the nominal cost parameters for each state. While the methods to develop and update the necessary RA models are out of the scope of this paper, we envision that existing modeling frameworks, e.g., [14], [60], can be extended to calculate A_{nm}, b_{nm}. For the soft constraint violation cost, the agent associated with each soft constraint, ag_{sc}, provides the a_{sft}^k, b_{sft}^k to the DA-PA. a_{sft}^k, b_{sft}^k represent how undesirable the constraint violation would be to ag_{sc}, in terms of a performance metric k. Currently, the cost of constraint violations is determined using feedback from an operator or a subject matter expert. Future work will develop methods to effectively estimate these soft constraint violation costs for both PAs and RAs. To ensure desirable performance of other agents, the DA-PA will minimize the cost of constraint violations ensuring that constraints are only violated if necessary (see Section IV-B).

C. Decision Making Model

A decision making model is the following tuple: $\mathcal{A}_{DM} = (X, Prp, x_0, \Sigma, E, C, \Gamma, I, Ag, X_m, \mathcal{K})$, where $(X, Prp, x_0, \Sigma, E, C, \Gamma, I, Ag)$ are obtained from the environment model, $X_m \in X$ is a set of marked states, and $\mathcal{K} : X \times val(C) \times val(\Gamma) \rightarrow \mathbb{R}_{\geq 0}$ denotes the state costs. The set of marked states, X_m, represents desired states for the DA-PA [45]. A discussion of how the decision making model is put together is presented in Section IV-A.

Figure 2 shows an example of a decision making model for a DA-PA in a manufacturing system composed of 2 robots (R1 and R2), 2 machines (M1 and M2), and 2 buffers (B1 and B2). There are 4 RAs in the system, one for each robot and machine. The buffers are used by the robots to pick and

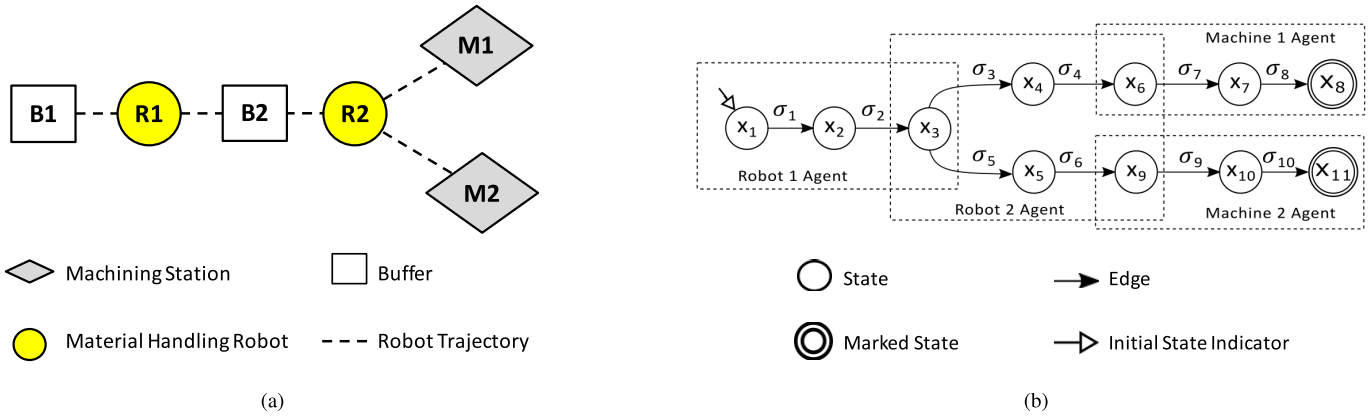


Fig. 2. (a) An overview of a system with 2 robots (R1, R2), 2 machines (M1, M2), 2 buffers (B1, B2). (b) A visualization of the DA-PA's decision making model for this system. The system has 4 resource agents representing R1, R2, M1, and M2. R1 moves the part from B1 (x_1) to B2 (x_3). R2 moves the part from B2 (x_3) to M1 (x_6) or M2 (x_9). M1 and M2 complete manufacturing process 1 (P1), represented by x_8 and x_{11}. The part is at B1 and needs P1 completed (x_8 or x_{11}) due to its process plan. Note that x_2, x_4, x_5 represent states when the part is moving and x_7, x_{10} represent states when the part is undergoing a physical change via a manufacturing process. See Tables II and III for more information about states and events.

TABLE II
STATE PROPERTIES, INVARIANTS, AND COSTS FOR FIG. 2

State, (x)	Properties, $Prp(x)$	Invariant, $I(x)$	Cost, $\mathcal{K}(x)$
x_1	At B1	None	$\alpha_t \mathcal{K}^t(x)$
x_2	Moving to B2	$val(c_l^x) > t^{sft}_{R1,B2} - val(\gamma_l^x)$ $val(c_l^x) > t^{nm}_{R1,B2}$	$\alpha_t \mathcal{K}^t(x) +$ $\alpha_e \left(\mathcal{K}^{nrg}_{nm}(x) + \mathcal{K}^{nrg}_{sft}(x) \right)$
$x_3/$ x_6/x_9	At B2/ M1/M2	None	$\alpha_t \mathcal{K}^t(x)$
$x_4/$ x_5	Moving to M1/M2	$val(c_l^x) > t^{nm}_{R2,M1/M2}$	$\alpha_t \mathcal{K}^t(x) + \alpha_e \mathcal{K}^{nrg}_{nm}(x)$
$x_7/$ x_{10}	At M1/M2 P1 Working	$val(c_l^x) > t^{nm}_{M1/M2,P1}$	$\alpha_t \mathcal{K}^t(x) + \alpha_e \mathcal{K}^{e}_{nm}(x)$
$x_8/$ x_{11}	At M1/M2 P1 Finished	$val(c_g^x) < t_d$	$\alpha_t \mathcal{K}^t(x)$

TABLE III
EVENT DESCRIPTIONS FOR FIG. 2

Event, (σ)	Event Name	Event, (σ)	Event Name
$\sigma_1, \sigma_3, \sigma_5$	Start moving to B2/M1/M2	$\sigma_2, \sigma_4, \sigma_6$	Finish moving to B1/M1/M2
σ_7, σ_9	Start P1 at M1/M2	σ_8, σ_{10}	Finish P1 at M1/M2

place parts in the system and do not have an associated agent in this example. B1 is used by R1 and B2 is used by both R1 and R2. The machines complete a manufacturing process (P1) required as part of the DA-PA's process plan. This initial state, x_0, and marked states, X_m are shown in Figure 2b. Note that in Figure 2b the initial state represents the current location of the part associated with DA-PA and the marked states represent the next desired process based on the DA-PA's process plan. The dashed rectangles outline which states and events are associated with a specific agent in the agent association function, e.g. states x_1, x_2, x_3 and events σ_1, σ_2 are associated with the R1 agent.

Table II shows the physical properties, constraints, and costs for each of the states in the example decision making model. All of the constraints (i.e., invariants) in Table II are bound constraints (see Definition 4). These constraints represent both (1) desirable, but flexible, operating limits (soft constraints) and (2) physical/safety time-based limits (hard constraints) for the resources in this example. For example, state x_2 contains both a soft and a hard constraint. The soft constraint, $val(c_l^x) > t^{sft}_{R1,B2} - val(\gamma_l^x)$, represents the *desired* operating time limit, $t^{sft}_{R1,B2}$, for R1 to take the part from B1 to B2. If necessary, R1 can perform the material handling operation faster (i.e., violate the soft constraint) if the value of the slack variable is greater than 0 for that state, $val(\gamma_l^x) > 0$. However, as discussed later in this section, there is an energy penalty for this constraint operation. In addition to the soft constraint, a hard constraint for state x_2, $val(c_l^x) > t^{nm}_{R1,B2}$, represents the absolute minimum time, $t^{nm}_{R1,B2}$, required by R1 to take the part from B1 to B2. Note that $t^{nm}_{R1,B2}$ is the upper bound of the time that it takes for R1 to perform this action. In this example, $t^{nm}_{R1,B2}$ is a guaranteed time limit for the robot to finish moving a part from B1 to B2, even with stochastic uncertainties for the action. The constraints for states x_4, x_5, x_7, and x_{10} also represent similar physical limits for corresponding resource operations. Furthermore, there is a hard constraint on x_8 and x_{11}, $val(c_g^x) < t_d$, that represents a customer deadline of time $t - d$ to complete the manufacturing process.

An overview of the costs is also provided in Table II. The state cost is a combination of two metrics: processing time and energy expenditure. The processing time cost is the amount of time spent in that state: $K^t(x) = val(c_l^x)$. The energy expenditure cost is provided by the RAs and represents the energy used by the RA for the part to stay at the state. For the purpose of simplification, we assume that RAs can save energy when they take a longer time to accomplish tasks. Note that this relationship can be true in specific regions of operation for both industrial robots [61] and manufacturing tools [62], but does not necessarily hold for all parameters and tasks.

Assuming that the speed of operation and the energy cost are inversely proportional for Robot 2, the energy cost for moving to B2 is evaluated to the following: $\mathcal{K}_{nm}^{nrg}(x) = c_{nm} \cdot val(c_l^x)$ and $\mathcal{K}_{sft}^{nrg}(x) = c_{sft} \cdot val(\gamma_l^x)$, where c_{nm} and c_{sft} are energy usage coefficients. The penalty term, $\mathcal{K}_{sft}^{nrg}(x)$, represents the extra energy required by the RA to accomplish this operation if it has to operate faster than its desired operating time limit. Note that $c_{sft} > c_{nm}$ to ensure that there is a penalty for violating the soft constraint. Future work will test the proposed approach when there are more complex relationships between the speed and the energy cost of accomplishing a task, such as described in [61] and [62].

Table III provides information about the events in the system. Note that each event is associated with a single agent and the DA-PA has to (1) schedule time at the resource, and (2) requests these events at a scheduled time. For example, the DA-PA can send a request to the M1 agent to schedule the use of M1 at a specific time. Then, once the associated part arrives at M1 (state x_6), the DA-PA can send a request to start the P1 manufacturing process (event σ_7, "Start P1 at M1") to the machine, thus transition the part to the next state, x_7. Further details on this process are provided in Section IV-D.

IV. Cooperation Framework

The DA-PA uses the goals, environment model, and decision making model to find a new path, or sequence of resource actions, in the system after receiving an update to its environment model. An update to the DA-PA's environment model occurs either (1) when the DA-PA is provided information about the environment during initialization (2) another agent contacts the DA-PA with new information due to an unexpected disturbance or (3) when RAs reply to a PA query for new information about the environment (e.g. [13]). Direct, active cooperation is used by the DA-PA to find paths in the system that allow all of the agents to meet their goals, if possible. Once a path is found, the DA-PA will start to schedule events with the RAs in the system. If a path is not found, the DA-PA will contact an RA to exit the system [13].

After the DA-PA receives an update to the environment model, the DA-PA goes through the 4 steps in the cooperation framework: (1) model creation, (2) path planning, (3) coordination, and (4) scheduling. A high-level overview of the cooperation framework and the steps are shown in Figure 3 and described in detail in this section. Once cooperation is finished, the DA-PA executes the plan developed during the scheduling phase through requests to the corresponding RAs.

Note that if the DA-PA cannot find, schedule, or execute a path, the DA-PA will try to re-build the environment model and retry each of these steps. If a DA-PA does not find a solution within a certain number of iterations or within a bounded time, it will take the exit plan described in Section III-A to provide a recommendation or help the part exit the system [13].

A. Model Creation

The DA-PA goes into the model creation phase after receiving an update for its environment model. In this phase, the DA-PA updates the environment model to capture the current capabilities and constraints of the environment. Then, the DA-PA creates the decision making model used to find the next sequence of resource actions.

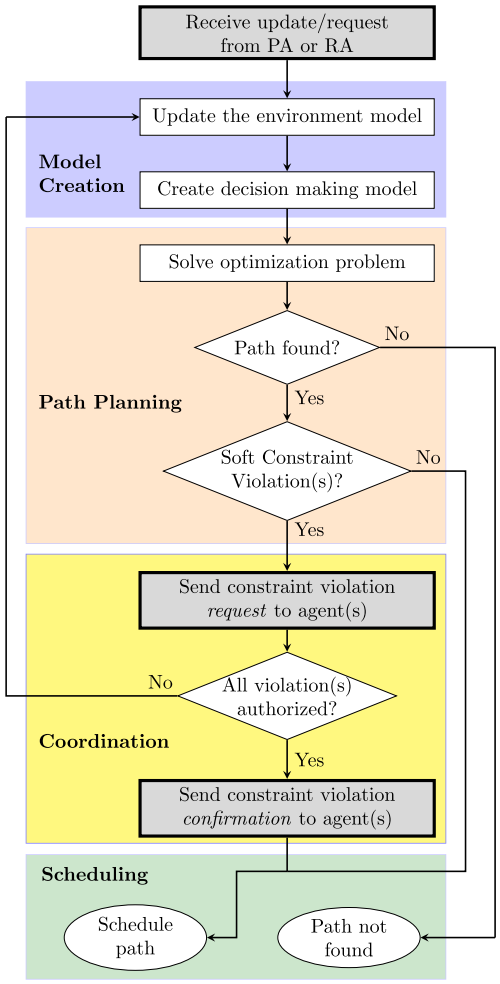


Fig. 3. An overview of the direct, active cooperation framework. The framework includes 4 steps: model creation, path planning, coordination, and scheduling. The gray boxes indicate when the DA-PA communicates with other agents.

1) Update the Environment Model: The DA-PA's initial step in the model creation phase is to update the environment model using the information provided to the DA-PA by another PA or RA in the system. Other agents can send a message to the DA-PA to update any component of the $\mathcal{A}_{EM}$, i.e. $(X, Prp, x_0, E, Tr, C, \Gamma, I, \mathcal{K}_{nm}, \mathcal{K}_{sft}, Ag)$. These updates represent changes in the manufacturing environment, e.g. new resources, updated scheduling constraints, changes to the associated part's current state, etc. While the communication and messaging protocols are out of scope of this paper, examples of the modular composition of an environment model can be found in [10], [59], and [14].

2) Create the Decision Making Model: The DA-PA combines $\mathcal{A}_{EM}$ and its goals to create the decision making model, $\mathcal{A}_{DM}$, by taking the following steps. To create $\mathcal{A}_{DM}$, the DA-PA computes a set of marked states and updates the constraints and costs of $\mathcal{A}_{EM}$.

The DA-PA computes the set of marked states, X_m by obtaining the next unfinished physical property, pp_d, from its process plan. The DA-PA marks a state (i.e. adds the state to the set of marked states in $\mathcal{A}_{DM}$) if pp_d matches the physical property of a state in the environment model: $pp_d \in Prp(x_i), \forall x_i \in X_m$. Deadlines are incorporated in $\mathcal{A}_{DM}$ by adding a hard constraint, $val(c_g) < Dl(pp_d)$, to the constraints of the marked states, $I(x_i), \forall x_i \in X_m$.

To account for the priority of other the parts in the system, the DA-PA updates the soft constraint violation costs from $\mathcal{A}_{EM}$ before incorporating these costs into $\mathcal{A}_{DM}$. Let the DA-PA be denoted as a_c and its priority as pr_{a_c}. Since each constraint is mapped to an agent with Ag, the DA-PA obtains a constraint's associated agent and the agent's priority, a_i, $pr_{a_i} = Ag(\mathcal{B}_i)$. Soft constraints with a higher priority DA-PA (i.e. $pr_{a_i} < pr_{a_c}$) are hardened by setting the corresponding $a_\gamma = 0$ (see Eq. 1). For soft constraints with a lower priority DA-PA (i.e. $pr_{a_i} > pr_{a_c}$), the penalty for constraint violation is removed by setting $a^k_{sft} = 0$ and $b^k_{sft} = 0$ (see Eq. 4). Soft constraints with an equal priority DA-PA (i.e. $pr_{a_i} = pr_{a_c}$) are not changed.

The state cost, $\mathcal{K}(x_j), \forall x_j \in X$ is a function that depends on the valuation of soft constraint violations and the clock valuations. The DA-PA computes the state cost by weighing the state cost for each metric in $\mathcal{A}_{EM}$, $p_1, p_2, \ldots, p_n$, with its performance weights, $\alpha_{p_1}, \alpha_{p_i}, \ldots, \alpha_{p_n}$. Formally, the cost for each state is defined as:

$$\mathcal{K}(x_j) = \sum_{i=1}^{n} \alpha_{p_i} \left(\mathcal{K}^{p_i}_{nm}(x_j) + \mathcal{K}^{p_i}_{sft}(x_j) \right), \tag{5}$$

where $\mathcal{K}^{p_i}_{nm}(x_j)$ and $\mathcal{K}^{p_i}_{sft}(x_j)$ are defined in Section III-B. Note that $\mathcal{K}^{p_i}_{sft}(x_j)$ is a summation of slack costs for nonzero slack valuations. Thus, the state cost is dependant on the soft constraint violation cost only if there is a constraint violation, i.e., nonzero slack variable valuation.

B. Path Planning

After building the decision making model, the DA-PA solves an optimization problem to find the path on the decision making model. A path on the decision making model PTA is a sequence of transitions that will take the associated part from its current state to a marked state in the decision making model, while satisfying the constraints in the system.

Definition 6 (Path): A path s on a PTA is an ordered set of discrete and delay transitions in which the post-transition state of each transition is equal to the pre-transition state of the subsequent transition. Thus a path can be notated as

$$s = \left\langle t_{d,\tau_1}\ t_{s,e_1}\ t_{d,\tau_2}\ t_{s,e_2}\ \ldots\ t_{d,\tau_n}\ t_{s,e_m} \right\rangle \tag{6}$$

and the path's total cost is given by the sum of the costs for all discrete and delay transitions in the path.

As stated in Remark 1, a path will always start with a delay transition, alternate between discrete and delay transitions, and end with a discrete transition so that $n = m$ in (6).

Let $\xi_i = (x_i, c_i, \gamma_i)$ denote a shorthand for the state, clock values, and slack values respectively at the i^{th} transition of the path s, $s(i)$. Note that $s(i)$ can be either a delay or a discrete transition. To pose the path planning problem as an optimization problem, we additionally define

$$\phi(s, \xi_1) = \left\langle \xi_1 \xrightarrow{s(1)} \xi_2, \xi_2 \xrightarrow{s(2)} \xi_3, \ldots, \xi_N \xrightarrow{s(N)} \xi_{N+1} \right\rangle$$

as the *solution sequence*, where $\xi_1 = (x_1, c_1, \gamma_1)$ is the initial state, clock values, and slack values respectively, and $N = 2n$ is the total number or transitions of (6), with $n = m$. We define the projection operator $\Pi_G(\phi(\cdot,\cdot))$ that projects the solutions sequence to a subspace $G \subset \mathcal{A}_{DM}$. For example $\Pi_X(\phi(s, \xi_1)) = \{\langle x_1, \ldots, x_f \rangle \mid x_i \in \phi(s, \xi_1)\}$ projects the solution subspace to the states in $\mathcal{A}_{DM}$. We use $\phi(s)$ instead of $\phi(s, \xi_i)$ for brevity when the arguments are clear from the context.

Thus, the DA-PA solves the following optimization problem to find a cost-optimal path:

$$s^* \in \arg\min_{s_k} \sum_{i=1}^{N} \mathcal{K}(x_i) \tag{7a}$$

$$\text{s.t. } x_i \in \Pi_X(\phi(s_k, \bar{\xi})), \tag{7b}$$

$$s_k \in \mathcal{L}(\mathcal{A}_{DM}), \tag{7c}$$

$$\#(\Pi_\Gamma(\phi(s_k, \bar{\xi}))) \leq \zeta, \tag{7d}$$

$$\Pi_X(\phi(s_k, \bar{\xi})) \cap X_m \neq \emptyset \tag{7e}$$

where $\mathcal{L}(\mathcal{A}_{DM})$ denotes the set of all feasible paths (i.e. the language) of $\mathcal{A}_{DM}$, $\bar{\xi} = (\bar{x}, \bar{c}, 0)$ is the initial state and clock value, $\#(\cdot)$ denotes the number of nonzero elements in the argument, $\zeta \in \mathbb{Z}_{>0}$ is the number of allowable constraint violations for the solution, and s_k is a path as defined in Definition 6. The solution s^* is a path that minimizes the cost function $\mathcal{K}(x_i)$, where x_i are the states in the solution sequence defined by constraint (7b). Constraint (7c) ensures that the path is in the language of $\mathcal{A}_{DM}$, ensuring that all the constraints in $\mathcal{A}_{DM}$ are satisfied and all the transitions are well-defined. Constraint (7d) defines an upper bound on the number of constraint violations so the solution sequence s^* can violate at most ζ constraints. Note that we recover the total number of constraint violations in the optimal solution using $\#(\Pi_\Gamma(\phi(s^*)))$. Constraint (7e) ensures that at least one of the marked states is in the solution path, i.e., $x \in \Pi_X(\phi(s^*))$, $x \in X_m \subset \mathcal{A}_{DM}$. The choice of N in (7) depends on the computational time requirements of the DA-PA. If there are no specific computation time requirements, the DA-PA can use longest path length of the PTA to guarantee feasibility of the problem (see [63] for different choices of N in a receding horizon setting). In Section V-D, we show that solving (7) by using the longest path of the PTA for N is justified for a PTA with 50 states and up to 10^4 constraints.

If the number of constraint violations, $\#(\Pi_\Gamma(\phi(s^*)))$, is zero, then the DA-PA can schedule the obtained sequence of actions without the need to negotiate over conflicting events, as described in Section IV-D. If a solution contains a nonzero number of constraint violations, the DA-PA has to coordinate with other agents to find a solution to the conflicting events prior to scheduling actions. More detail about coordination is discussed in the following section. Finally, if a solution to the above problem cannot be obtained or this optimization problem is not solved without a bounded time, the DA-PA will

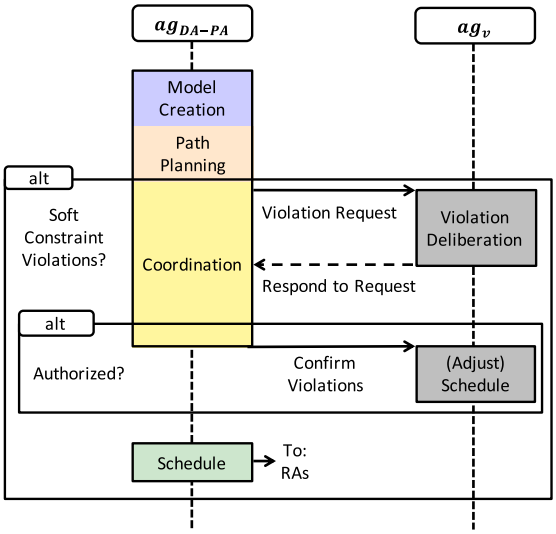


Fig. 4. A sequence diagram for DA-PA coordination.

move to the scheduling step without a new plan. As described in Section IV-D, the lack of a new plan will force the DA-PA to either find a new plan or call an exit agent.

C. Coordination

The DA-PA has to coordinate its optimal path with other agents if there are any constraint violations found during its path planning phase, i.e., $\#(\Pi_\Gamma(\phi(s^*))) > 0$. An overview of the coordination procedure is shown in Figure 4. The DA-PA finds the set of corresponding agents (PA or RA) for each constraint that has a non-zero slack valuation (violated constraint): $Agents = \{Ag(\mathcal{B}_v) \mid \mathcal{B}_v \in \mathcal{S}[val(C), val(\Gamma)], val(\gamma_{v,i}) > 0, val(\gamma_{v,i}) \in Val(\Gamma_v)\}$, where Γ_v are the slack variables associated with this constraint. Note that this mapping is not one-to-one as a single agent in the set $Agents$ can be associated with multiple violated constraints.

The DA-PA sends a *violation request* to each of the agents in the set $Agents$, where each agent in this set is either a PA or an RA. For every state x_i^* in the state sequence of the optimal path $\Pi_X(\phi(s^*))$, the DA-PA finds the corresponding delay transition t_{d,τ_j} such that $(x_i^* \xrightarrow{t_{d,\tau_j}} x_i^*) \in \phi(s^*)$. The *violation request* contains all of the states in the optimal path and the corresponding delay transitions. Thus, the *violation request* represents new scheduling constraints to be considered by each agent, $ag_v \in Agents$. The contacted agent, ag_v, should be able to understand this request, deliberate, and respond to the DA-PA. Examples of how other PAs and RAs respond to request violations is provided in the supplementary material [64].

If all of the contacted agents authorize their constraint violations, the DA-PA sends a message confirming new scheduling constraints that correspond to the delay transitions in its optimal path to each of the contacted agents. This confirmation message is handled by each respective agent and can result in the need for this agent to re-schedule its own plans. After the confirmation message is sent out, the DA-PA will start to schedule the path with the RAs, as described in Section IV-D.

Note that any agent can deny, i.e. not authorize, a constraint violation request. Examples of why another agent would deny a constraint violation request are provided in the supplementary material [64]. If the constraint violation is not authorized, the DA-PA may still be able to find a path in the environment that can fulfill its goals. If a constraint violation is denied (i.e., non-authorized), the DA-PA will update the environment model to avoid violating the non-authorized constraint. Non-authorized constraints are hardened by setting the hardness coefficient in the constraint to 0 (see Definitions 4 and 5) until the environment model is re-updated at a future time. Then, the DA-PA goes through each of the steps in the cooperation framework in Figure 3 with the new hardened constraints.

D. Scheduling

If the DA-PA finds a path s^* and all of the constraint violations are authorized by other agents, the DA-PA starts to schedule the transitions with the RAs in the system. Similar to the previous step, for every state x_i^*, the DA-PA finds the corresponding delay transition, t_{d,τ_j}. Then, the DA-PA sends a scheduling request to the agents associated with the state, $x_i^* \in Ag(x_i^*)$. The request is an interval, $[l_{x_i}, u_{x_i}]$ with l and u representing the resource utilization start time and end time, respectively. Note that for the initial state in the solution sequence, x_1, the resource utilization start time is 0, $l_{x_1} = 0$, and the end time is the valuation of the global clock for the initial state, $u_{x_1} = \tau_1$. For subsequent delay transitions, the enter time corresponds to the exit time of the previous delay transition, $l_{x_i} = u_{x_{i-1}}$ and the exit time is calculated as $u_{x_i} = l_{x_i} + \tau_i$. After checking for any scheduling conflicts, the RAs confirm or reject each scheduled actions.

If a DA-PA goes to the scheduling stage without a new plan, the DA-PA can try to find a new plan or call an exit agent. If a solution was not found in a bounded time, the DA-PA can try to find a sub-optimal optimal path for the decision-making model. The methodology to find this sub-optimal path is not in the scope of this paper and will be considered in future work. Another method that the DA-PA can use to find a new plan is to try to capture some changes in the manufacturing system by re-exploring and re-building the environment model [14]. This method allows the DA-PA to address various disturbances, faults, and failures in the manufacutring system [64]. Finally, if a DA-PA is not able to find a solution within a bounded time or a bounded number of iterations, it will have to exit the system via the exit plan by calling an exit agent provided in its exit plan (described in Section III-A).

E. Execution

Once the plan is calculated by a DA-PA and confirmed by the RAs, the DA-PA executes the plan by requesting events from the RAs at the planned times. Starting at the initial state of the solution sequence, x_1, the DA-PA transitions between states by sending an event request to the RAs in the system at the planned time. To transition between states x_i^* and x_{i+1}^*, the DA-PA sends an event request, σ_e, to the agent, $Ag(e)$ where $e \in E$ is the corresponding edge in $t_{s,e} \in s^*$, i.e. $(x_i^* \xrightarrow{t_{s,e}} x_{i+1}^*) \in \phi(s^*)$, and σ_e is the event associated with

that edge. For example, after waiting at the initial state x_1 for τ_1, the DA-PA will request the event σ_1 from $Ag(e_1)$, where e_1 is the edge of the first discrete transition of s^* and σ_1 is the associated event. To improve the robustness of the DA-PA, the optimization for (7) is run after an event is requested by the DA-PA. If the optimal path s^* changes between subsequent solutions of (7), the DA-PA reschedules the time constraint schedules with the corresponding RAs, accordingly. Re-running the optimization after every event ensures that the DA-PA will be able to adapt its plan in the presence of small changes (e.g., due to system uncertainties). Note that for more disruptive changes to the manufacturing system (e.g., machine failures, changes to customer order), the DA-PA will need to re-build the environment model and go through each step of the decision making framework, as described in [64]. The DA-PA continues to request the discrete transitions until it completes all of the events in the path s^*.

Since the DA-PA plans for a single manufacturing process at a time, the path s^* ends at a marked state $x_i^* \in X_m$ that signifies the completion of a manufacturing process. After all of the transitions in the solution sequence are finished, the DA-PA finds the next unfinished manufacturing processes and then continues following the steps in the direct, active cooperation framework shown in Figure 3 (model creation, path planning, coordination, and scheduling).

F. Benefits of the Proposed Methodology

The direct, active cooperation methodology focuses on improving the conflict resolution of multi-agent controllers by extending the decision making process of a single PA, i.e., none of the other agents have access to any of the models, plans, or performance weights of the DA-PA. This direct, active cooperation approach has several benefits when compared to common conflict resolution approaches in existing multi-agent systems for manufacturing that use a supervisor agent to resolve conflicts, e.g., [6], [35]. One of the major benefits is that it is not necessary to combine multiple, distinct, models from different PAs and RAs during cooperation. These various models may be difficult to combine since they may have been gathered at different times and may represent different sets of resources in the manufacturing system [14]. A second benefit of this approach is that the optimization will scale reasonably with more PAs in the system, as shown in Section V-D. Finally, the DA-PA retains a lot of the information internally since only violated constraints are shared with other agents. If a manufacturing system contains a large number of different PAs that represent various parts from different customers, it may be highly beneficial and preferable for a customer to ensure that all of their process plans, models, and performance weights are not shared with other agents.

V. Case Studies

This section presents how direct, active cooperation can be used to improve the performance of manufacturing systems using simulations of the manufacturing environment. The first case study describes how the cooperation framework is applied to the system introduced in Section III-C. The second case study shows how the DA-PA can be used to improve the flexibility of a larger manufacturing environment. The scalability study analyzes how the developed optimization methodology can be scaled to larger manufacturing systems.

The case studies provided in this section overview how the system controlled by a multi-agent system responds to a new, unexpected high priority order. The benefit of using DA-PAs for this specific unexpected disturbance is then discussed. Note that it is possible for the DA-PA to respond to other unexpected disturbances and faults in the system, as detailed in the supplementary material [64].

A. Simulation Setup

The Repast Simphony agent-based simulation platform [65] was used to test and analyze the behavior of a DA-PA in a manufacturing environment. This agent-based simulation software has been previously used to test agent-based controllers for manufacturing systems [13], [34]. Repast Simphony is a discrete-time simulation platform, as the simulation is updated every time step (also known as every tick).

The developed manufacturing simulation contains high-level discrete-event dynamics for material handling robots and machines used for manufacturing. The proposed cooperation framework extended the simulation of the model-based PAs described in [13]. In this updated simulation, each PA builds and uses a finite state machine (FSM) model of a local neighborhood of the manufacturing system to make decisions. The PAs build the FSM by exploring the manufacturing system through queries to the RAs in the system, as described in [14]. Each RA provides an FSM that corresponds to its individual capabilities and the DA-PA combines this information into a PTA that represents the capabilities of the local manufacturing environment [13], [14]. A custom PTA class that extends the FSM class described in [13] was developed. This custom PTA class was used to encode the DA-PA environment model, the DA-PA decision making model, and the capability model used by the RAs when providing capability and scheduling information during exploration (see [14] for a detailed explanation of the exploration process). The communication framework native to Repast Simphony was used to encode messages and enable cooperation between agents.

A Cost Optimal Reachability Analysis (CORA) solver was developed to solve the optimization in (7). CORA has been previously used to find cost-optimal paths for PTA models. Two software tools that have been used to solve CORA for PTAs are UPPAAL CORA [51], [66] and satisfiability modulo theories (SMT) solvers [67], [68]. A custom implementation of the z3 SMT solver was used to encode soft constraints and find paths in the DA-PA's decision making model. A detailed overview and description of the solver implementation is found in [63]. For this implementation, the dynamics and constraints of the decision making model were transformed into first-order logic expressions. The solver output was the optimal path for the DA-PA and a set of constraint violations used in the cooperation framework. JavaScript Object Notation (JSON)

TABLE IV
TIME LIMITS FOR RESOURCE ACTIONS IN CASE STUDY 1: SMALL MANUFACTURING SYSTEM SIMULATION

Resource	Program	Time (ticks)
Robot 1	Move to Buffer 2	15
Robot 1	Move to Buffer 2 (Fast)	5
Robot 2	Move to Machine 1	17
Robot 2	Move to Machine 2	17
Machine 1	Process 1	100
Machine 2	Process 1	150

was as an interface between the Repast Simphony simulation and the z3 SMT solver.

B. Case Study 1: Small Manufacturing System

A simulation of the manufacturing system introduced in Section III-C is used to analyze the behaviour of DA-PAs in a small manufacturing system environment. The system contains two robots (R1 and R2), two machines (M1 and M2), and two buffers (B1 and B2). New parts entering the system start at buffer 1 (B1) and, once the part enters the system, a new DA-PA is initialized. During initialization, the DA-PA is provided the process plan and deadlines, the exit plan, the performance weights, and the priority for the physical properties in the process plan.

In this example, the process plan contains one physical property: complete process 1 (P1). The exit plan is an agent that can take the part associated with the DA-PA out of the system. For this example, the DA-PAs attempts to minimize both the time and energy expenditure in the system with $\alpha_t = 1$ and $\alpha_e = 0.5$, respectively. To test the DA-PA cooperation framework, various priority values and deadlines were provided to the DA-PAs during initialization, as detailed in the following examples.

DA-PAs use the exploration methodology described in Section V-A to send requests to RAs in the system to start the cooperation process described in Section IV. For example, a PTA that is created by a DA-PA for a part in the system without any work-in-progress is shown in Figure 2 and described in detail in Section III-C. The time limits for the resource actions in the simulation are shown in Table IV. The time limits in Table IV are used to build the constraints in the decision making model shown in Table II from Section III-C. Note that for robot 1, the energy expenditure is much greater if the soft constraint is violated, i.e. if robot 1 has to move the part to B2 faster than 15 ticks. Thus, $c_{nm} = 10$ kWh and $c_{sft} = 100$ kwH are the costs for staying at state x_2, $\mathcal{K}^{nrg}_{nm}(x_2) = c_{nm} \cdot val(c_l^{x_2})$ and $\mathcal{K}^{nrg}_{sft}(x_2) = c_{sft} \cdot val(\gamma_l^{x_2})$. To conform with the simulation environment, we require the part to stay in every state for at least one tick. Therefore, an additional constraint, $val(c_l^x) \geq 1$, is added for each state in the decision making model, $x \in X$. The following three examples illustrate the benefits of direct, active cooperation.

1) Non-Cooperative PAs: This example illustrates the scenario when two DA-PAs do not need to cooperate with each other to complete their individual goals and satisfy the system objectives. In this example, a part comes into the system with no deadline provided for P1. A DA-PA, ag^1_{pa}, is created with an importance value of 2. The DA-builds the decision making model shown in Figure 2 and calculates its optimal path by solving (7). The solver described in the previous section is used to find a cost-optimal path in the system. Then, after solving the optimization problem, ag^1_{pa} schedules the path that consists of delay and discrete transitions to complete P1 at M1. Examples of delay transitions that a DA-PA can request include "stay at B1 for 1 tick" from buffer 1 agent, "stay moving to B2 for 15 ticks" from the robot 1 agent, or "work on process 1 for 100 ticks" from the machine 1 agent. Examples of discrete transitions that a DA-PA can request include "start moving to B2/M1/M2" from the robot agents or "start p1 at M1/M2" from the machine agents. These discrete, delay transitions allow the DA-PA to communicate with RAs, enabling the DA-PA to make dynamic decisions for the associated part and accomplish the desired objectives in the manufacturing system.

A second part comes into the system 10 ticks after the first part. This is an unexpected disturbance for the system that was not captured by ag^1_{pa} during its initial model creation and, therefore, not considered by ag^1_{pa} during path planning. The second DA-PA, ag^2_{pa}, does not have deadlines and has an importance value of 2. The optimal path for ag^2_{pa} does not have any constraint violations. Thus, ag^2_{pa} schedules resource actions to take the associated part to the slower machine, M2, to complete its process plan. The flow times for the ag^1_{pa} and ag^2_{pa} to complete their process plans are 141 and 197 ticks, respectively.

2) DA-PA to DA-PA Cooperation With Order Prioritization: For this example, the manufacturer prioritizes the completion of the part associated with ag^2_{pa} over other parts in the system. Both ag^1_{pa} and ag^2_{pa} are set-up in the same way as in the previous example. However, ag^2_{pa} is given an importance value of 1, making it higher priority than ag^1_{pa}. As in the previous example, ag^1_{pa} initially chooses to go to M1 to complete the process plan. After 5 ticks (when the part associated with ag^1_{pa} is using Robot 1), the more important part enters the system and ag^2_{pa} is initialized. By following the proposed cooperation framework, ag^2_{pa} requests ag^1_{pa} to reschedule its plan. ag^1_{pa} authorizes the request after finding a feasible path that takes the associated part to M2 to complete P1. In this scenario, the flow times for the ag^1_{pa} and ag^2_{pa} are 221 and 145 ticks, respectively. Compared to the first example, the process plan of ag^2_{pa} was completed faster, even though the total flow time of the two parts is longer than the first example. Note that, even with the increased flow time, both agents still accomplished their process plan successfully.

3) DA-PA to DA-PA Cooperation to Meet Deadlines: Both ag^1_{pa} and ag^2_{pa} are set-up in the same way as Example 1. Unlike example 2, the priority of both parts is the same for this example. However, ag^2_{pa} is given a deadline, or hard constraint, of 180 ticks to complete P1. As in examples 1 and 2, ag^1_{pa} initially chooses to go to M1 to complete the process plan. However, when the part associated with ag^2_{pa} comes into the system, ag^2_{pa} requests the utilization of M1 from ag^1_{pa}. This request is authorized by ag^1_{pa} after it finds another

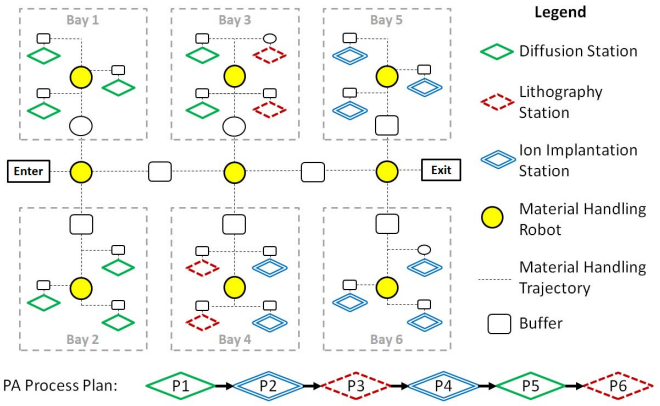


Fig. 5. Set-up for case study 2: semiconductor manufacturing.

TABLE V
PROCESS TIME LIMITS FOR CASE STUDY 2: SEMICONDUCTOR MANUFACTURING

Manufacturing Process	Time (ticks per lot)
Process 1 (Diffusion)	150
Process 2 (Ion Implementation)	60
Process 3 (Lithography)	110
Process 4 (Ion Implementation)	100
Process 5 (Diffusion)	170
Process 6 (Lithography)	20

feasible path in the system (complete P1 at M2). The flow times for ag^1_{pa} and ag^2_{pa} are 221 and 145 ticks, respectively. The process plans for both DA-PAs, including the deadline constraint of 180 ticks for ag^2_{pa}, are accomplished successfully due to the cooperation between the two DA-PAs.

4) Example 4: DA-PA to RA Cooperation to Meet a Deadline: Only one DA-PA is considered for this example. In this example, a part comes into the system and DA-PA is initialized with a deadline of 135 ticks. To accomplish this task, ag^1_{pa} cooperates with the robot 1 (R1) RA and requests to move to B2 fast (in 5 ticks), violating R1's soft constraint. Since this will ensure that ag^1_{pa} meets its deadline, the RA confirms this violation at the expense of a higher energy expenditure. In this scenario, the flow time for ag^1_{pa} is 135 ticks and the DA-PA is able to successfully accomplish its process plan due to the cooperation between the DA-PA and the RA.

C. Case Study 2: Semiconductor Manufacturing Facility

A simulation of a semiconductor manufacturing facility was used to study the behavior of direct and active cooperation in a larger manufacturing system. The set-up for the simulation is shown in Figure 5. This simulated system contains 20 manufacturing stations that perform different manufacturing operations (diffusion, ion implementation, and lithography). The process times for each of these manufacturing operations are shown in Table V. Each machine has a local buffer where parts stays before starting a requested operation and after the operation is finished.

These stations are connected by a network of material handling robots that move products, known as lots, around the system. The robots take 10-20 ticks to move the lots between the various buffers, depending on the distance in the simulation. The process plan for the lots is provided in Figure 5. This simulation has been previously used to test the behavior of multi-agent controllers for manufacturing systems [13], [55].

A customer requests an order (order A) of 10 lots with the process plan shown in Figure 5. To complete this order, 10 parts, each with their own DA-PA, enter the system with the same priority every 50 ticks. A higher priority order (order B) of 10 lots with the same process plan is requested 500 ticks after the first order. Similar to the first order, 10 new parts controlled by 10 DA-PAs enter into the system every 50 ticks. The order B DA-PAs have a higher importance value than the order A DA-PAs. Neither order has a specific deadline, but the manufacturer would prefer to complete order B parts faster than order A parts. The introduction of new parts and their DA-PAs into the system is an unexpected disturbance for the work-in-progress parts as the DA-PAs for these work-in-progress parts did not capture the presence of new parts during model creation and path planning. The number of allowable constraint violations was set to 2, i.e., $\zeta = 2$, when the DA-PA solved the optimization problem in (7).

Similar to the previous case study, DA-PAs use the exploration methodology described in Section V-A to send requests to RAs in the system to start the cooperation process described in Section IV. For example, the PTA that is created by the first part from Order A that required process 1 is illustrated in Figure 6. Initially, the part is at the Enter buffer and the PTA contains states that represent buffering, moving, and processing operations. The marked states represent how the DA-PA can accomplish process 1 (diffusion) in the local manufacturing environment. Each of these states contains an invariant that represents the time required to stay at state x, as a function of $val(c^x_l)$. As new parts come into the system and DA-PAs complete their desired processes, the PTAs that are built by new DA-PAs will contain more interval gap constraints that represent the other DA-PA schedules in the system.

The DA-PAs use the PTAs from model creation to make decisions in the manufacturing system. After finding the desired path on the PTA, the DA-PA schedules and executes this desired path, as described in Section IV. This path contains discrete and delay transitions that represent the various buffering, handling, and processing operations in the manufacturing system, e.g., "hold at enter buffer" is a delay transition, "start moving to bay 1" is a discrete transition, "start the diffusion operation" is a discrete transition. These DA-PAs continue to build PTAs and use these PTAs until all of the desired physical processes have been completed.

During the simulation, multiple DA-PAs will sometimes request the utilization of the same resource because they are performing the scheduling task at the same time. In the simulation, the RA that is associated with the resource agent will utilize a first-come first-serve strategy toward accepting these requests. The RA will then reject all of the other DA-PA requests. If a DA-PA is rejected, it will re-explore the system, re-create the environment model, and proceed with the steps in

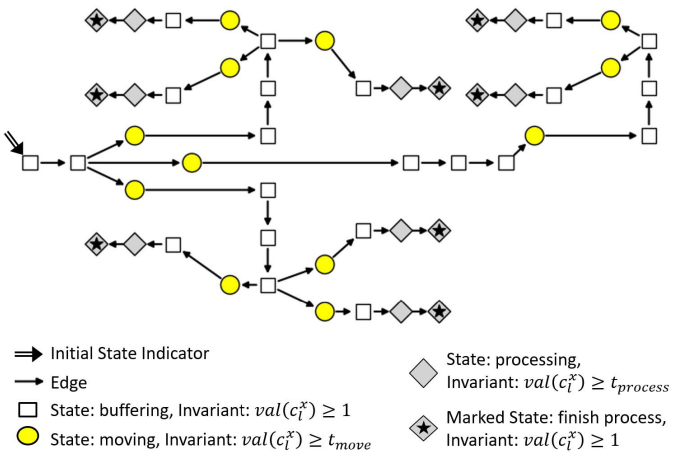


Fig. 6. An illustration of the initial PTA created by the DA-PA associated with the first part in Order A. This PTA is created by exploring the system through queries to RAs for process 1 and combining the information provided through these queries, as described in [14].

the cooperation framework. Note that the rejected DA-PA will end up either coordinating with the DA-PA that scheduled the initial resource or finding a new resource to utilize. If there are multiple unsuccessful cooperation requests, the DA-PA will call the exit plan, as described in Section III-A.

The process times shown in Table V are deterministic and represent the maximum possible time that the machines will take to finish the operation, i.e., the operation is guaranteed to be finished in the number of ticks in Table V. These time limits are provided by the RAs to the PAs as the limits for hard bound constraints. If the machine finishes the operation before the time limit, the part will be moved to the local machine buffer and the machine remains idle or accepts another job if one is requested. Future work will look to encode more stochastic uncertainties in the PTA constraints and test these uncertainties in this simulation. While the processing times are deterministic, there is some stochasticity in the simulation due to the choices made by the DA-PAs. Sometimes, the DA-PAs will have multiple optimal paths in the PTA, i.e., there are several paths that take the same amount of time to reach a desired state. For this case, the DA-PAs randomly select one of these paths when making their decision.

Twenty five trials were conducted to test the effect of direct and active PA to PA cooperation. Each set of five trials had a different number of order B DA-PAs that would request violations from order A DA-PAs. The number of order B DA-PAs that request violations was 0, 2, 4, 6, 8, and 10 for each set of five trials, respectively.

For every trial, the average flow time of the 10 parts of order A and order B was calculated. Figure 7 shows the distribution for the average flow time of orders A and B based on the number of cooperative products in order B. Each data point in Figure 7 represents the minimum, maximum, and mean of the average flow time for 5 trials. Note that as more cooperation is allowed for order B DA-PAs, the average flow time of these parts decreases. However, as order B DA-PAs utilize the resources required by order A DA-PAs, the average flow time of order A parts increases. Thus, direct and active

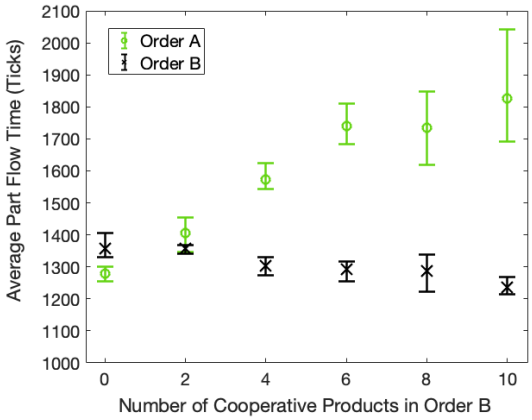


Fig. 7. Distribution of order A and order B average flow times for the semiconductor manufacturing case study.

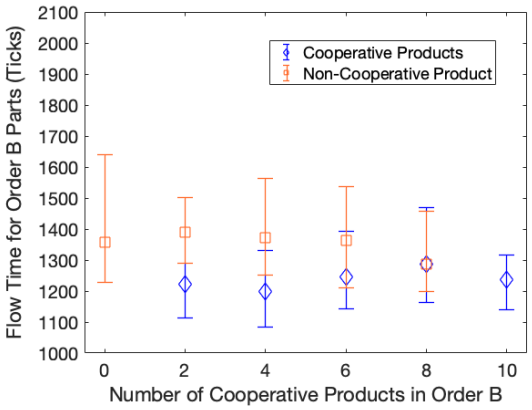


Fig. 8. Distribution of individual part flow times for order B in the semiconductor manufacturing case study.

cooperation will allow higher priority parts to be completed quicker, but slow down the production of lower priority parts.

Figure 8 shows the resulting flow times for order B parts for all of the trials. Note that Figure 8 represents the distribution of flow time for *individual* order B parts, while Figure 7 represents the distribution of average flow times for orders A and B in the system. On average, order B DA-PAs with cooperation capabilities had a flow time of 1238 ticks. Order B DA-PAs that did not cooperate had a flow time of 1354 ticks. Thus, the DA-PAs that were able to cooperate reduced the flow time in the manufacturing system by 8.5%.

In this case study, order A DA-PAs always accepted an order B DA-PA violation request. Thus, if an order B DA-PA found multiple optimal paths with different constraint violations, order B DA-PAs would randomly choose one of the paths and request the corresponding violation. Hence, the order A DA-PAs would sometimes have to significantly change their schedules due to these requests. Therefore, (1) including hard deadlines for the DA-PAs and/or (2) accurate estimation of soft constraint violation costs could reduce the variability of the flow times and decrease average flow times for order A DA-PAs. Note that even with this behavior, all of the DA-PAs

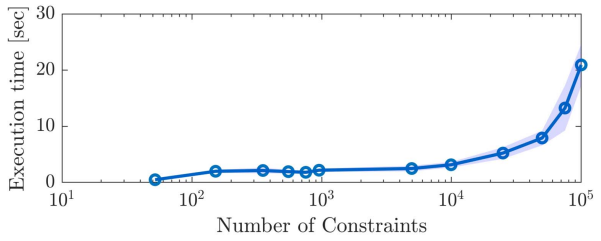


Fig. 9. Computational scale study for the optimization problem in the case study using the PTA model in Fig. 6. Mean and one standard deviation over ten repetitions are shown.

successfully accomplished their process plan since neither order A nor order B DA-PAs had any process plan deadlines.

D. Scalability Study

We perform a computational study to illustrate the scalability of the proposed optimization approach for solving (7). In this study, we utilize the PTA shown in Figure 6, which is created by the DA-PA associated with the first part in Order A. We study the case with a fixed number of RAs and an increasing number of DA-PAs in the system. To simulate the effect of increasing DA-PAs, we implement an increasing number of interval gap constraints (see (2)) and solve the optimization problem outlined in (7).

The results of the scaling study are presented in Figure 9. The simulation runs are repeated 10 times for each number of constraints shown on the horizontal axis of Figure 9. All simulations are performed on a personal laptop computer with Intel Core i7-7700HQ CPU. The longest path of the PTA in Figure 6 includes 14 discrete transitions and a horizon length corresponding to 16 discrete transitions is used in all simulations. The horizon of 16 discrete transitions corresponds to $N = 32$ for (7). The nominal optimization problem without the injected interval gap constraints has 52 bound constraints in total, which takes 0.73 ± 0.22 seconds to compute. The results show that the number of constraints has a pronounced effect on the computation time only after approximately 5000 interval gap constraints, which results in over 3 seconds of computation time to solve (7). The computation time remains well below 30 seconds for more than 10^5 constraints. Each DA-PA typically requests 10-50 constraints when scheduling events in the semiconductor manufacturing case study described in Section V-C. Therefore, 10^5 interval constraints would correspond to somewhere between 10^3 to 10^4 DA-PAs in the system. At these scales, both communication and physical bottlenecks may be more pronounced than the computational bottleneck caused by solving (7).

VI. Conclusion and Future Work

Multi-agent control has been proposed to improve the flexibility of manufacturing systems. One of the primary agents in this control strategy is the product agent. The product agent must cooperate with other product agents and resource agents in the system when making scheduling and planning decisions. However, existing product agent cooperation strategies have been primarily passive, reducing the flexibility and adaptability of the product agent and, in turn, of this control strategy.

A model-based decision making framework to enable direct, actively cooperative product agents is introduced. For this framework, a new environment model that leverages the priced timed automata modeling formalism is proposed. This model explicitly represents the scheduling constraints in the system and separates the desirable, but flexible, time limits (soft constraints) from the physical/safety time limits (hard constraints). An optimization problem is formulated that allows the direct and active product agent to identify any conflicting scheduling constraints. A strategy for product agent coordination and negotiation with other agents in the system is also presented. This strategy resolves the conflicts in the system, allowing the agents to effectively control and coordinate the components on the factory floor. This framework is tested in a simulated manufacturing environment, showing how direct, active cooperation can improve the flexibility and performance of manufacturing systems.

Future work will include the following: developing and incorporating new process plan formalisms to capture a larger variety of production orders; creating methods to effectively estimate soft constraint violations costs for product and resource agents; developing algorithms for agent violation deliberation after a violation request; and integrating further time-based specifications in the product agent process plan. Further simulation testing of the proposed methodology will include incorporating hard constraints for larger manufacturing systems, varying the number of allowable constraint violations, and increasing the number of different priority orders in the system. The direct, actively cooperating product agent will also be integrated into a physical manufacturing testbed that run agent-based controllers to test the proposed framework with real industrial hardware and software.

References

[1] D. M. Tilbury, "Cyber-physical manufacturing systems," *Annu. Rev. Control, Robot., Auto. Syst.*, vol. 2, no. 1, pp. 427–443, 2019.

[2] P. Leitão, "Agent-based distributed manufacturing control: A state-of-the-art survey," *Eng. Appl. Artif. Intell.*, vol. 22, no. 7, pp. 979–991, 2009.

[3] B. Vogel-Heuser, J. Lee, and P. Leitão, "Agents enabling cyber-physical production systems," *Automatisierungstechnik*, vol. 63, no. 10, pp. 777–789, 2015.

[4] P. Vrba *et al.*, "Rockwell automation's holonic and multiagent control systems compendium," *IEEE Trans. Syst., Man, Cybern., C (Appl. Rev.)*, vol. 41, no. 1, pp. 14–30, Jan. 2011.

[5] A. M. Farid and L. Ribeiro, "An axiomatic design of a multiagent reconfigurable mechatronic system architecture," *IEEE Trans. Ind. Informat.*, vol. 11, no. 5, pp. 1142–1155, Oct. 2015.

[6] N. K. C. Krothapalli and A. V. Deshmukh, "Design of negotiation protocols for multi-agent manufacturing systems," *Int. J. Prod. Res.*, vol. 37, no. 7, pp. 1601–1624, May 1999.

[7] T. N. Wong, C. W. Leung, K. L. Mak, and R. Y. K. Fung, "Dynamic shopfloor scheduling in multi-agent manufacturing systems," *Expert Syst. Appl.*, vol. 31, no. 3, pp. 486–494, Oct. 2006.

[8] P. Leitao, A. W. Colombo, and F. Restivo, "An approach to the formal specification of holonic control systems," in *Proc. 1st Int. Conf. Ind. Appl. Holonic Multi-Agent Syst.*, 2003, pp. 59–70.

[9] J. Barbosa, P. Leitão, E. Adam, and D. Trentesaux, "Dynamic self-organization in holonic multi-agent manufacturing systems: The ADACOR evolution," *Comput. Ind.*, vol. 66, pp. 99–111, Jan. 2015.

[10] C. Legat, D. Schütz, and B. Vogel-Heuser, "Automatic generation of field control strategies for supporting (re-)engineering of manufacturing systems," *J. Intell. Manuf.*, vol. 25, no. 5, pp. 1101–1111, Oct. 2014.

[11] M. K. Lim, Z. Zhang, and W. T. Goh, "An iterative agent bidding mechanism for responsive manufacturing," *Eng. Appl. Artif. Intell.*, vol. 22, no. 7, pp. 1068–1079, Oct. 2009.

[12] S. Rehberger, L. Spreiter, and B. Vogel-Heuser, "An agent-based approach for dependable planning of production sequences in automated production systems," *At-Automatisierungstechnik*, vol. 65, no. 11, pp. 766–778, 2017.

[13] I. Kovalenko, D. Tilbury, and K. Barton, "The model-based product agent: A control oriented architecture for intelligent products in multi-agent manufacturing systems," *Control Eng. Pract.*, vol. 86, pp. 105–117, May 2019.

[14] I. Kovalenko, D. Ryashentseva, B. Vogel-Heuser, D. Tilbury, and K. Barton, "Dynamic resource task negotiation to enable product agent exploration in multi-agent manufacturing systems," *IEEE Robot. Autom. Lett.*, vol. 4, no. 3, pp. 2854–2861, Jul. 2019.

[15] K. Barton, F. Maturana, and D. Tilbury, "Closing the loop in IoT-enabled manufacturing systems: Challenges and opportunities," in *Proc. Annu. Amer. Control Conf. (ACC)*, Jun. 2018, pp. 5503–5509.

[16] F. Lopez, Y. Shao, Z. M. Mao, J. Moyne, K. Barton, and D. Tilbury, "A software-defined framework for the integrated management of smart manufacturing systems," *Manuf. Lett.*, vol. 15, pp. 18–21, Jan. 2018.

[17] J. Ota, "Multi-agent robot systems as distributed autonomous systems," *Adv. Eng. Informat.*, vol. 20, no. 1, pp. 59–70, Jan. 2006.

[18] J. A. Fax and R. M. Murray, "Information flow and cooperative control of vehicle formations," *IEEE Trans. Autom. Control*, vol. 49, no. 9, pp. 1465–1476, Sep. 2004.

[19] S. D. J. McArthur *et al.*, "Multi-agent systems for power engineering applications—Part I: Concepts, approaches, and technical challenges," *IEEE Trans. Power Syst.*, vol. 22, no. 4, pp. 1743–1752, Nov. 2007.

[20] A. Kantamneni, L. E. Brown, G. Parker, and W. W. Weaver, "Survey of multi-agent systems for microgrid control," *Eng. Appl. Artif. Intell.*, vol. 45, pp. 192–203, Oct. 2015.

[21] J. Du, V. Sugumaran, and B. Gao, "RFID and multi-agent based architecture for information sharing in prefabricated component supply chain," *IEEE Access*, vol. 5, pp. 4132–4139, 2017.

[22] F. Hsieh, "Dynamic configuration and collaborative scheduling in supply chains based on scalable multi-agent architecture," *J. Ind. Eng. Int.*, vol. 15, pp. 249–269, Sep. 2019.

[23] N. Mishra, A. Singh, S. Kumari, K. Govindan, and S. I. Ali, "Cloud-based multi-agent architecture for effective planning and scheduling of distributed manufacturing," *Int. J. Prod. Res.*, vol. 54, no. 23, pp. 7115–7128, Dec. 2016.

[24] P. Leitão, S. Karnouskos, L. Ribeiro, J. Lee, T. Strasser, and A. W. Colombo, "Smart agents in industrial cyber–physical systems," *Proc. IEEE*, vol. 104, no. 5, pp. 1086–1101, May 2016.

[25] W. Shen, L. Wang, and Q. Hao, "Agent-based distributed manufacturing process planning and scheduling: A state-of-the-art survey," *IEEE Trans. Syst., Man, Cybern., C (Appl. Rev.)*, vol. 36, no. 4, pp. 563–577, Jul. 2006.

[26] P. Leitao, V. Marik, and P. Vrba, "Past, present, and future of industrial agent applications," *IEEE Trans. Ind. Informat.*, vol. 9, no. 4, pp. 2360–2372, Nov. 2013.

[27] W. Lepuschitz, A. Zoitl, M. Vallée, and M. Merdan, "Toward self-reconfiguration of manufacturing systems using automation agents," *IEEE Trans. Syst., Man, Cybern., C (Appl. Rev.)*, vol. 41, no. 1, pp. 52–69, Jan. 2011.

[28] D. McFarlane, V. Giannikas, A. C. Y. Wong, and M. Harrison, "Product intelligence in industrial control: Theory and practice," *Annu. Rev. Control*, vol. 37, no. 1, pp. 69–88, Apr. 2013.

[29] W. Dai, H. Nishi, V. Vyatkin, V. Huang, Y. Shi, and X. Guan, "Industrial edge computing: Enabling embedded intelligence," *IEEE Ind. Electron. Mag.*, vol. 13, no. 4, pp. 48–56, Dec. 2019.

[30] L. Panait and S. Luke, "Cooperative multi-agent learning: The state of the art," *Auto. Agents Multi-Agent Syst.*, vol. 11, no. 3, pp. 387–434, Nov. 2005.

[31] M. Owliya, M. Saadat, G. G. Jules, M. Goharian, and R. Anane, "Agent-based interaction protocols and topologies for manufacturing task allocation," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 43, no. 1, pp. 38–52, Jan. 2013.

[32] M. Polycarpou, Y. Yang, Y. Liu, and K. Passino, "Cooperative control design for uninhabited air vehicles," in *Cooperative Control: Models, Applications and Algorithms*. Boston, MA, USA: Springer, 2003, pp. 283–321.

[33] J. Leitner, "Multi-robot cooperation in space: A survey," in *Proc. Adv. Technol. Enhanced Quality Life*, Jul. 2009, pp. 144–151.

[34] G. Jules and M. Saadat, "Agent cooperation mechanism for decentralized manufacturing scheduling," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 12, pp. 3351–3362, Dec. 2017.

[35] S. Wang, J. Wan, D. Zhang, D. Li, and C. Zhang, "Towards smart factory for industry 4.0: A self-organized multi-agent system with big data based feedback and coordination," *Comput. Netw.*, vol. 101, pp. 158–168, Jun. 2016.

[36] Y. Sallez, T. Berger, and D. Trentesaux, "A stigmergic approach for dynamic routing of active products in FMS," *Comput. Ind.*, vol. 60, no. 3, pp. 204–216, 2009.

[37] P. Valckenaers, M. Kollingbaum, and H. Van Brussel, "Multi-agent coordination and control using stigmergy," *Comput. Ind.*, vol. 53, no. 1, pp. 75–96, Jan. 2004.

[38] C. Hofmann, C. Krahe, N. Stricker, and G. Lanza, "Autonomous production control for matrix production based on deep Q-learning," *Proc. CIRP*, vol. 88, pp. 25–30, Jan. 2020.

[39] A. Lüder, A. Klostermeyer, J. Peschke, A. Bratoukhine, and T. Sauter, "Distributed automation: PABADIS vs. HMS," *IEEE Trans. Ind. Informat.*, vol. 1, no. 1, pp. 31–38, Jul. 2005.

[40] M. G. Marchetta, F. Mayer, and R. Q. Forradellas, "A reference framework following a proactive approach for product lifecycle management," *Comput. Ind.*, vol. 62, no. 7, pp. 672–683, Sep. 2011.

[41] H. Tang, D. Li, S. Wang, and Z. Dong, "CASOA: An architecture for agent-based manufacturing system in the context of industry 4.0," *IEEE Access*, vol. 6, pp. 12746–12754, 2018.

[42] P. G. Ansola, A. García, J. de las Morenas, and J. de las Morenas, "Aligning decision support with shop floor operations: A proposal of intelligent product based on BDI physical Agents," in *Service Orientation in Holonic and Multi-agent Manufacturing*, T. Borangiu, A. Thomas, and D. Trentesaux, Eds. Cham, Switzerland: Springer, 2015, pp. 91–100.

[43] J. de las Morenas, A. Garcia-Higuera, and P. Garcia-Ansola, "Shop floor control: A physical agents approach for PLC-controlled systems," *IEEE Trans. Ind. Informat.*, vol. 13, no. 5, pp. 2417–2427, Oct. 2017.

[44] H. Kaindl, M. Vallée, and E. Arnautovic, "Self-representation for self-configuration and monitoring in agent-based flexible automation systems," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 43, no. 1, pp. 164–175, Jan. 2013.

[45] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, 2nd ed. Boston, MA, USA: Springer, 2009.

[46] I. Kovalenko, K. Barton, and D. Tilbury, "Design and implementation of an intelligent product agent architecture in manufacturing systems," in *Proc. 22nd IEEE Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, Sep. 2017, pp. 1–8.

[47] E. C. Balta, Y. Lin, K. Barton, D. M. Tilbury, and Z. M. Mao, "Production as a service: A digital manufacturing framework for optimizing utilization," *IEEE Trans. Automat. Sci. Eng.*, vol. 15, no. 4, pp. 1483–1493, Oct. 2018.

[48] S. Baer, J. Bakakeu, R. Meyes, and T. Meisen, "Multi-agent reinforcement learning for job shop scheduling in flexible manufacturing systems," in *Proc. 2nd Int. Conf. Artif. Intell. Industries (AI4I)*, Sep. 2019, pp. 22–25.

[49] P. Leitão, A. W. Colombo, and F. Restivo, "A formal specification approach for holonic control systems: The ADACOR case," *Int. J. Manuf. Technol. Manage.*, vol. 8, nos. 1–3, pp. 37–57, 2006.

[50] K. Larsen *et al.*, "As cheap as possible: Efficient cost-optimal reachability for priced timed automata," in *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 2102. Berlin, Germany: Springer, 2001, pp. 493–505.

[51] G. Behrmann, K. G. Larsen, and J. I. Rasmussen, "Priced timed automata: Algorithms and applications," in *Proc. Int. Symp. Formal Methods Compon. Objects*. Berlin, Germany: Springer, 2004, pp. 162–182.

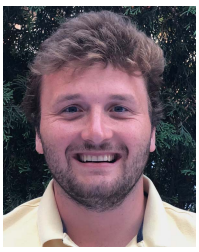
[52] D. L. Pepyne and C. G. Cassandras, "Optimal control of hybrid systems in manufacturing," *Proc. IEEE*, vol. 88, no. 7, pp. 1108–1123, Jul. 2000.

[53] F. Ocker, I. Kovalenko, K. Barton, D. Tilbury, and B. Vogel-Heuser, "A framework for automatic initialization of multi-agent production systems using semantic web technologies," *IEEE Robot. Automat. Lett.*, vol. 4, no. 4, pp. 4330–4337, Oct. 2019.

[54] G. Zheng, I. Kovalenko, K. Barton, and D. Tilbury, "Integrating human operators into agent-based manufacturing systems: A table-top demonstration," *Proc. Manuf.*, vol. 17, pp. 326–333, Jan. 2018.

[55] I. Kovalenko, D. Tilbury, and K. Barton, "Priced timed automata models for control of intelligent product agents in manufacturing systems," in *Proc. 15th Workshop on Discrete Event Syst. (WODES)*, 2020, pp. 136–142.

[56] C. J. Myers and T. H.-Y. Meng, "Synthesis of timed asynchronous circuits," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 1, no. 2, pp. 106–119, Jun. 1993.
[57] B. R. Ferrer, B. Ahmad, A. Lobov, D. A. Vera, J. L. Martinez Lastra, and R. Harrison, "An approach for knowledge-driven product, process and resource mappings for assembly automation," in *Proc. IEEE Int. Conf. Autom. Sci. Eng. (CASE)*, Aug. 2015, pp. 1104–1109.
[58] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints," *Automatica*, vol. 35, no. 3, pp. 407–427, Mar. 1999.
[59] J.-F. Pétin, D. Gouyon, and G. Morel, "Supervisory synthesis for product-driven automation and its application to a flexible assembly cell," *Control Eng. Pract.*, vol. 15, no. 5, pp. 595–614, May 2007.
[60] S. Rehberger, L. Spreiter, and B. Vogel-Heuser, "An agent approach to flexible automated production systems based on discrete and continuous reasoning," in *Proc. IEEE Int. Conf. Autom. Sci. Eng. (CASE)*, Aug. 2016, pp. 1249–1256.
[61] A. Liu, H. Liu, B. Yao, W. Xu, and M. Yang, "Energy consumption modeling of industrial robot based on simulated power data and parameter identification," *Adv. Mech. Eng.*, vol. 10, no. 5, 2018, Art. no. 1687814018773852.
[62] J. Lv, R. Tang, W. Tang, Y. Liu, Y. Zhang, and S. Jia, "An investigation into reducing the spindle acceleration energy consumption of machine tools," *J. Cleaner Prod.*, vol. 143, no. 1, pp. 794–803, 2017.
[63] E. C. Balta, I. Kovalenko, I. A. Spiegel, D. M. Tilbury, and K. Barton, "Model predictive control of priced timed automata encoded with first-order logic," *IEEE Trans. Control Syst. Technol.*, vol. 30, no. 1, pp. 352–359, Jan. 2022.
[64] I. Kovalenko, E. C. Balta, D. M. Tilbury, and K. Barton, "Cooperative product agents to improve manufacturing system flexibility: A model-based decision framework," Barton Res. Group, Univ. Michigan, Ann Arbor, MI, USA, Tech. Rep., 2021. [Online]. Available: https://brg.engin.umich.edu/wp-content/uploads/sites/131/2022/02/DA-PA-agentResponse.pdf
[65] M. J. North *et al.*, "Complex adaptive systems modeling with repast simphony," *Complex Adapt. Syst. Model.*, vol. 1, no. 1, p. 3, Dec. 2013.
[66] G. Behrmann. (2014). *UPPAAL CORA*. [Online]. Available: http://people.cs.aau.dk/~adavid/cora/index.html
[67] D. Bhave, S. N. Krishna, and A. Trivedi, "On nonlinear prices in timed automata," *Electron. Proc. Theor. Comput. Sci.*, vol. 232, p. 65, Mar. 2016.
[68] M. Hekmatnejad, G. Pedrielli, and G. Fainekos, "Task scheduling with nonlinear costs using SMT solvers," in *Proc. IEEE 15th Int. Conf. Autom. Sci. Eng. (CASE)*, Aug. 2019, pp. 183–188.



Ilya Kovalenko (Member, IEEE) received the B.S. degree in mechanical engineering from the Georgia Institute of Technology, Atlanta, GA, USA, in 2015, and the M.S. and Ph.D. degrees in mechanical engineering from the University of Michigan, Ann Arbor, MI, USA, in 2018 and 2020, respectively. He held a post-doctoral research position at the University of Michigan, from Fall 2020 to Fall 2021. In January 2022, he joined the Department of Mechanical Engineering and the Department of Industrial and Manufacturing Engineering with Pennsylvania State University, where he currently holds the position of Assistant Professor. His research interests include control theory and application, with a focus on multi-agent systems, advanced manufacturing, cyber-physical systems, and robotics.

Efe C. Balta (Member, IEEE) received the B.S. degree in manufacturing engineering from the Faculty of Mechanical Engineering, Istanbul Technical University, Turkey, in 2016, and the M.S. and Ph.D. degrees in mechanical engineering from the University of Michigan, Ann Arbor, in 2018 and 2021, respectively. Since June 2021, he has been a Post-Doctoral Researcher with the Automatic Control Laboratory (IfA), ETH Zürich, Switzerland. His research interests include control theory, learning-based control, and optimization methods with applications to additive manufacturing, cyber-physical systems, robotics, and advanced manufacturing processes.

Dawn M. Tilbury (Fellow, IEEE) received the B.S. degree *(summa cum laude)* in electrical engineering from the University of Minnesota in 1989 and the M.S. and Ph.D. degrees in electrical engineering and computer sciences from the University of California at Berkeley in 1992 and 1994, respectively. In 1995, she joined the Department of Mechanical Engineering, University of Michigan, Ann Arbor, where she is currently a Professor, with a joint appointment as a Professor of EECS. In June 2017, she became the Assistant Director of Engineering at the National Science Foundation, while maintaining her position at the University of Michigan. She has published more than 150 articles in refereed journals and conference proceedings. Her research interest includes control systems, including applications to robotics and manufacturing systems.

Kira Barton (Senior Member, IEEE) received the B.Sc. degree in mechanical engineering from the University of Colorado at Boulder in 2001 and the M.Sc. and Ph.D. degrees in mechanical engineering from the University of Illinois at Urbana-Champaign, in 2006 and 2010, respectively. She held a post-doctoral research position at the University of Illinois at Urbana-Champaign, from Fall 2010 to Fall 2011, at which point she joined the Department of Mechanical Engineering, University of Michigan, Ann Arbor. She is currently an Associate Professor with the Department of Mechanical Engineering and a Core Robotics Faculty Member at the University of Michigan. She conducts research in modeling, sensing, and control for applications in advanced manufacturing and robotics, with a specialization in cooperative learning and smart manufacturing. She was selected as a Miller Faculty Scholar from 2018 to 2021, and was a recipient of the NSF CAREER Award in 2014; the 2015 SME Outstanding Young Manufacturing Engineer Award; the 2015 University of Illinois, Department of Mechanical Science and Engineering Outstanding Young Alumni Award; the 2016 University of Michigan, Department of Mechanical Engineering Department Achievement Award; and the 2017 ASME Dynamic Systems and Control Young Investigator Award.