Dynamic Resource Allocation Using Multi-Agent Control for Manufacturing Systems [★]

Mingjie Bi [*] **Ilya Kovalenko** [**] **Dawn M. Tilbury** [***] **Kira Barton** [***]

[*] *Robotics Institute, University of Michigan, Ann Arbor, MI 48109 USA (e-mail: mingjieb@umich.edu)*
[**] *Department of Mechanical Engineering, University of Michigan, Ann Arbor, MI 48109 USA (e-mail: ikoval@umich.edu)*
[***] *Department of Mechanical Engineering and Robotics Institute, University of Michigan, Ann Arbor, MI 48109 USA (e-mail: {tilbury, bartonkl}@umich.edu)*

Abstract: The COVID-19 pandemic brings highly dynamic effects to manufacturing environments, such as frequently shifting markets and unexpected disruptions. Such dynamic environments increase the demand for flexible and real-time manufacturing decision-making strategies. One essential problem is dynamic resource allocation to complete production tasks, especially when a resource disruption (e.g. machine breakdown) occurs. Multi-agent frameworks have been proposed to improve the flexibility and responsiveness of manufacturing systems in a distributed decision-making manner. This work introduces a clustering method based on resource agent (RA) capabilities and an RA coordination strategy that enables dynamic resource reallocation when the manufacturing system is subject to resource disruptions.

Keywords: Flexible manufacturing systems, agents, resource allocation, coordination, discrete-event dynamic systems

1. INTRODUCTION

The COVID-19 pandemic highlighted the dynamic characteristics of the manufacturing environment, such as a frequently shifting market, product customization requirements, and unexpected disruptions (e.g. machine breakdowns) (Kumar et al., 2020). Therefore, the demand for flexible and real-time decision-making strategies was particularly magnified during the COVID-19 pandemic (Li et al., 2020). Manufacturing enterprises need to make prompt and feasible decisions regarding the effective allocation of limited resources and flexibly respond to unexpected disruptions (Cardin et al., 2017).

In current manufacturing systems, centralized decision-making strategies have been widely used to provide optimized resource allocation decisions based on specific objectives (e.g. throughput) but often require significant computational time to calculate (Leitão, 2009). As such, these centralized strategies often lack the ability to dynamically respond to unexpected disruptions in a timely manner (Leitão, 2009). To improve the flexibility and responsiveness of manufacturing systems, distributed strategies, where multiple system entities interact collaboratively to make decisions, have been proposed in several works (Cardin et al., 2017; Leitão, 2009).

Recently, multi-agent control strategies have been proposed to address distributed and intelligent manufacturing decision-making (Leitão, 2009). Two agents that are used in most existing architectures include a product agent (PA) and a resource agent (RA). PAs and RAs are responsible for controlling an associated physical part and resource, respectively (Kovalenko et al., 2017, 2019a,b). In a multi-agent architecture, PAs and RAs coordinate for resource allocation and flexible response to disruptions (Zhang and Wong, 2017; Wong et al., 2006). An overview of the multi-agent control strategy is shown in Fig. 1.

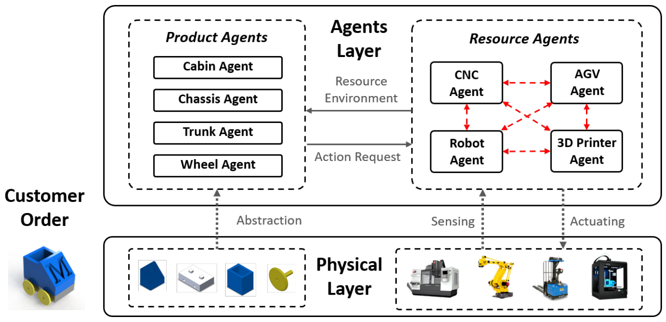


Fig. 1. A diagram of a multi-agent manufacturing system

This work focuses on reallocating resources dynamically to complete the manufacturing tasks affected by a resource disruption. An RA is assumed to identify this disruption by continuously collecting data from its associated resource. Some existing multi-agent architectures address this problem through either PA-RA or RA-RA coordination. In several works (Kovalenko et al., 2019a,b; Wong et al., 2006; Lepuschitz et al., 2011), the disrupted RA informs the PAs and triggers the PA-RA coordination to generate a new resource allocation schedule based on the remaining tasks and resource capacity without preserving the initial schedule. Therefore, these methods have a high probability of causing deviations from the initial schedule, limiting the scheduling robustness (Vieira et al., 2003).

For RA coordination methods, Farid and Ribeiro (2015) introduce a reconfiguration agent (RCA) as a mediator for RA coordination. Rodrigues et al. (2018) provide a collaborative mechanism to enable a disrupted RA to request all of the other

★ This work was funded in part by NSF 1544678

RAs to make allocation decisions. However, both of the two methods above create a significant communication load that will limit the agility of the system in response to a disruption. To reduce agent communication, clustering approaches have been used to provide a structured coordination process. In Maturana et al. (1999), an RA cluster is defined as a set of dissimilar RAs that collaborate to complete a sub-task. Barata and Camarinha-Matos (2003) define an RA cluster based on resource proximity. However, considering only nearby resources might cause resource overload, reducing throughput and resource utilization.

To address the problems identified above, this paper provides the following contributions: (1) the extension of an RA knowledge base to enable capability clustering, (2) the development of a capabilities-based clustering scheme for multi-agent manufacturing systems in dynamic environments, and (3) the demonstration of this framework through a simulation study.

The rest of the paper is organized as follows. Section 2 describes the resource allocation problem and agent formulation. Section 3 presents the RA knowledge base. Section 4 describes resource reallocation via RA coordination. A case study is provided in Section 5, and conclusions are stated in Section 6.

2. PROBLEM OVERVIEW AND AGENT FORMULATION

In this section, a resource allocation problem in the form of a rescheduling task is described and formulated.

2.1 Problem Overview

Resource allocation can be formulated as a production scheduling problem (Shen et al., 2006), which refers to the process of specifying resources to perform operations on the parts with constraints. However, unexpected resource disruptions (e.g. breakdown) often occur in dynamic manufacturing systems, leading the initial production schedule to not be executed completely (Zhang and Wong, 2017). Therefore, rescheduling, the process of reallocating resources for the parts affected by the disruption (Abumaizar and Svestka, 1997), becomes necessary.

The method proposed in this paper focuses on resource allocation for affected operations through agent-based rescheduling when a resource disruption occurs. To formulate the problem, the following assumptions are provided:

A.1 The initial production schedule for each part is determined before the manufacturing system starts.
A.2 Production demand will be met if the resources follow the initial production schedule.
A.3 Resource disruptions occur unexpectedly and are detectable by the associated RAs.
A.4 A resource disruption results in the specific resource becoming unavailable for a certain amount of time.
A.5 The manufacturing system contains resource redundancy and is operating with available capacity.

A1 and A2 ensure that an initial schedule has been set and if followed, will ensure that the demand is met. A3 guarantees that a disruption will be identified by a resource. A4 designates how a resource will be impacted by a disruption. Finally, A5 is necessary to enable agent coordination and part rerouting.

2.2 Agent and Problem Formulation

Product Agent A PA is responsible for controlling an associated physical part to fulfill the desired production requirements. Thus, the PA needs to track the production progression of the associated part through the manufacturing system. In this work, the PA stores the status of the part at any given instance as a discrete state in the set $X = \{x_0, x_1, ..., x_f\}$. Each state is comprised of two elements, $x_i = (x_i^l, x_i^c)$, where x_i^l represents the part's location and x_i^c denotes the part's physical composition. For example, a PA state can be denoted as: $x = (x^l$: "at milling machine1", x^c : "with a milled pocket"). Note that x_f is the final state of the part in the manufacturing system. x_f^c should meet the customer specifications, and x_f^l represents the location where the final physical composition task is performed, or in some cases, where a material handling buffer is located.

Resource Agent An RA provides high-level control for a physical resource to perform operations on a part. In this work, RAs are grouped into two RA classes: transportation and transformation RAs, in which the RA's operations (i.e. events) drive a state change in either the location or physical composition. For a transportation RA, the events, denoted by $E^l = \{e_0^l, e_1^l, ..., e_m^l\}$, represent changes in the location of a part. Similarly, the events of a transformation RA are denoted by $E^c = \{e_0^c, e_1^c, ..., e_n^c\}$, representing changes in the physical composition of a part. Each event represents a transition between two states, which is discussed in detail in section 3.2. Note that RAs in the same RA class can have the same events (e.g., two milling machines with similar capabilities).

Production Schedule A production schedule for one part specifies a sequence of resource operations (i.e., events) that leads to part state transitions from the initial state, x_0, to the final state, x_f. The entire production schedule for the manufacturing system is composed of the production schedules for every part. A production schedule for one part is defined as:
$P_s = (s_x, s_e, Ag_p, T_p)$, where
$s_x =< x_0, ..., x_f >$: a sequence of states describing how the part states change as the part moves through the system
$s_e =< e_0, ..., e_{f-1} >$: the scheduled sequence of events that trigger the state transitions
$Ag_p : s_e \to RA$: the event-agent associate function
$T_p : s_e \to (\mathbb{R}_+, \mathbb{R}_+)$: a function that maps events to start and end times, which indicates the resource working time
The functions T_p and Ag_p map each event in s_e to its occurring time and to the RA that performs the event, respectively. Note that an event type (e.g. milling a pocket) might occur multiple times in s_e, but at varying occurrence times and with different RAs. The sequences s_x and s_e satisfy the transition relationship:

$$x_{i+1} = T_r(x_i, e_i), \text{ for } 0 \le i \le f-1 \tag{1}$$

where T_r is a state transition function. Due to the transition relation, the start and end times in T_p indicate the times when the part is associated with a specific state.

Problem Formulation From A1, the initial production schedule specifies the resource operations for each part. Each RA stores its scheduled events as a sequence $S_c =< e_{s0}...e_{sk} >$. Each event $e_{si} \in S_c$ corresponds to a specific part on which the RA will perform the event. The start and end times of event e_{si} are identified in the production schedule P_s of the corresponding PA. Once a resource disruption occurs, the associated RA is able to identify the disruption (A3). Due to this disruption, the associated RA cannot perform a sub-sequence of scheduled events (A4), denoted by $E_d =< e_{d0}e_{d1}...e_{dl} > \subseteq S_c$. All the events in E_d need to be assigned to alternative resources, which requires resource redundancy and available capacity (A5).

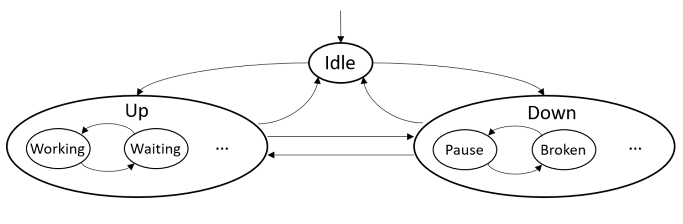


Fig. 2. An example of an RA state model

For each event that needs rescheduling, $e_d \in E_d$, the rescheduling may introduce changes to its sequential events (scheduled before and after e_d) within the PA's initial production schedule, P_s. As such, the rescheduling process aims to search for a new sequence of events (s_{new}) that achieves the necessary transitions to successfully complete the manufacturing transitions:

$$Tr(x_{prior}, s_{new}) = x_{post} \tag{2}$$

where x_{prior} and x_{post} refer to the states before and after the affected event sequence in P_s. Note that the new sequence aims to satisfy, as best it can, the production requirements.

In the manufacturing environment, RAs that have the same capabilities as the disrupted RA may be able to provide potential solutions for the rescheduling problem (A.5). Therefore, a detailed RA architecture needs to be designed to describe RA capabilities and manage the rescheduling requirements. The requirements for the disrupted RA are as follows: R1) Detect the resource disruption; R2) Identify and coordinate a cluster of RAs that contain the same capabilities as the disrupted RA for a given event; R3) Determine alternative RAs to perform the affected events based on certain metrics.

R1 requires the use of a state model to detect the status of an RA at any given time. For R2, a detailed capabilities model needs to be developed to identify RAs that have the same capabilities for a given event type. To coordinate with these RAs, a clustering method will be used to provide a more structured coordination process. R3 is satisfied by solving an optimization problem. To address all of these requirements, this paper proposes a capabilities-based clustering approach and a rescheduling strategy via RA coordination.

3. RESOURCE AGENT KNOWLEDGE BASE

In this section, the RA knowledge base, composing of state model, capabilities model, and environment model, is provided.

3.1 RA State Model

To capture the resource disruption, the RA state model is developed to describe the states and transitions of a resource. The event-driven transitions allow one to model the resource as a finite state machine (FSM), which has been previously used in Qamsane et al. (2019). Similarly, the RA state model in this paper is defined as an FSM with the states *Idle*, *Up*, and *Down*. Transitions occur between these three states and also inside the *Up* and *Down* states, as shown in Fig. 2.

3.2 RA Capabilities Model

The capabilities model describes the operations that the resource can perform on the part to change its location or physical composition. Based on the state and event description in Section 2.2, the FSM can be used to represent the RA resource capabilities (Kovalenko et al., 2019a; Balta et al., 2021).

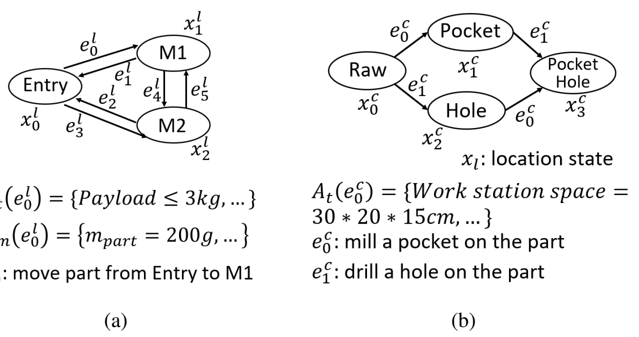

Entry
M1
M2
x_0^l
x_1^l
x_2^l
e_0^l
e_1^l
e_2^l
e_3^l
e_4^l
e_5^l
$A_t(e_0^l) = \{Payload \le 3kg, \ldots\}$
$R_m(e_0^l) = \{m_{part} = 200g, \ldots\}$
e_0^l: move part from Entry to M1
(a)
Raw
Pocket
Hole
Pocket Hole
x_0^c
x_1^c
x_2^c
x_3^c
e_0^c
e_1^c
x_l: location state
$A_t(e_0^c) = \{Work\ station\ space = 30 * 20 * 15cm, \ldots\}$
e_0^c: mill a pocket on the part
e_1^c: drill a hole on the part
(b)

Fig. 3. Examples of capabilities model for (a) mobile robot agent (b) machine agent, where A_t represents the resource attributes, R_m represents the task requirements, and e represents the event that the RAs can perform

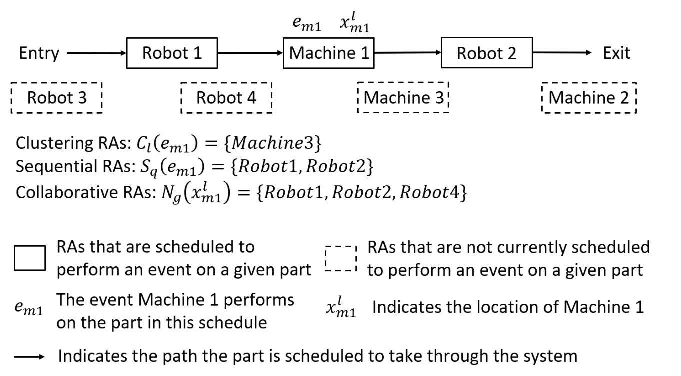
Transformation RA The capabilities model for a transformation RA in this paper is defined as:
$C_M = (x_l, X^c, E^c, T_r, T, A_t, T_a, S_c, R_m)$:

- x_l: a description of the transformation resource's location
- $X^c = \{x_0^c, \ldots, x_n^c\}$: a set of states representing all of the physical compositions that can be achieved on a part
- $E^c = \{e_0^c, \ldots, e_m^c\}$: a set of events representing actions that change the physical composition of a part
- $T_r : X^c \times E^c \to X^c$: a state transition function
- $T : E^c \to \mathbb{R}_+$: the cost it takes for the event to occur
- $A_t : E^c \to C_a$: a function that maps events to the physical resource attributes of each event (e.g. payload limitations)
- $T_a : E^c \to I_t$: a set of available times for each event
- $S_c =< e_{s0}^c \ldots e_{sk}^c >$: a sequence of scheduled events
- $R_m : S_c \to R_t$: a function that maps scheduled events to the task requirements for each scheduled event (e.g. precision)

Transportation RA The capabilities model of a transportation RA is similar to transformation RA with the following changes:

- $X^l = \{x_0^l, \ldots, x_n^l\}$: a set of states representing locations that can be reached by a part utilizing the resource
- $E^l = \{e_0^l, \ldots, e_m^l\}$: a set of events representing actions that change the location of a part.

In the capabilities model, the state sets, X^l and X^c, represent the locations and physical compositions that are associated with a transportation or transformation RA, respectively. Event sets, E^l and E^c, represent the operations that the resource can perform on a part. The transition function, T_r, encodes how the events lead to state transitions. T represents the nominal cost (denoted as operation time) for each event to occur. A_t provides the resource attributes for each event. Note that multiple RAs in the same class could have the same events, but the attributes might be different. The characteristics of the attributes are described by parameters, such as the limitation of speed, payload, and part dimensionality. S_c is a dynamic sequence of events that has been scheduled to be performed in a predefined time horizon. Each scheduled event is associated with a part. R_m maps each scheduled event to task requirements, such as the part's weight and size, acceptable resolution, etc. $T_a : E^c \to I_t$ provides a set of time intervals when the resource is available for each event. $I_t = \{(t_0, t_1), \ldots, (t_{2p}, t_{2p+1})\}$, with $t_0 < t_1 < \ldots < t_{2p+1}$, where p is determined based on the current scheduled events S_c and their time costs. Thus, event e can be scheduled between t_0 and t_1, between t_2 and t_3, etc. The proposed capabilities model is initialized based on the physical manufacturing system. As an RA receives information constantly from the resources on



e_{m1}
x^l_{m1}
Entry
Robot 1
Machine 1
Robot 2
Exit
Robot 3
Robot 4
Machine 3
Machine 2
Clustering RAs: $C_l(e_{m1}) = \{Machine3\}$
Sequential RAs: $S_q(e_{m1}) = \{Robot1, Robot2\}$
Collaborative RAs: $N_g(x^l_{m1}) = \{Robot1, Robot2, Robot4\}$
RAs that are scheduled to perform an event on a given part
RAs that are not currently scheduled to perform an event on a given part
e_{m1} The event Machine 1 performs on the part in this schedule
x^l_{m1} Indicates the location of Machine 1
Indicates the path the part is scheduled to take through the system

Fig. 4. Example of an environment model for a machine agent

the shop floor and other agents in the system, the capabilities model will be updated if the capabilities and schedules of the resource change. The elements X^l, X^c, E^l, E^c, T and A_t describe resource intrinsic capabilities. These elements are updated as the physical resource capabilities change. The changes could be manual, such as tool replace/removal, or automatic, such as machine breakdown. The elements T_a, S_c and R_m represent the current schedule status of the resource. These elements are dynamically updated as other agents request to change, add, or remove resource schedules.

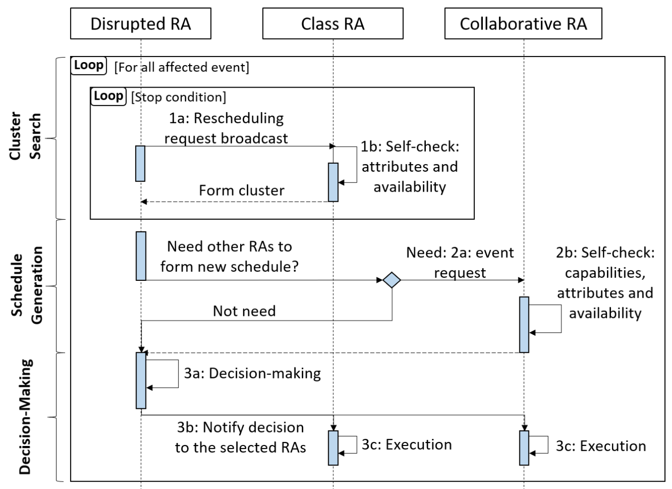
Two examples of capabilities models C_M for a mobile robot agent and a machining agent are provided in Fig. 3. In this scenario, the mobile robot has capabilities to move parts between an entry and two machines. The machine $M1$ can mill a pocket and a hole on a part.

3.3 RA Environment Model

The environment model contains several mapping functions that map events or states to three sets of RAs, namely clustering, sequential, and collaborative RAs.

Clustering RAs In this work, a clustering method based on the capabilities of the RAs is proposed. RA clusters are formed when rescheduling is needed, i.e. when an RA identifies a resource disruption. The disrupted RA, termed RA_d, cannot perform the affected events, denoted by E_d, which need to be reallocated to other RAs. Note that $E_d \subseteq S_c$ should be scheduled events. The RA_d broadcasts a rescheduling request for each event $e_{di} \in E_d$ and its associated task requirements. The request can be accessed by RAs in the same class as RA_d (e.g. transformation RAs). RAs that can satisfy the event and task requirements in a specified time horizon will respond to the broadcast from RA_d. Importantly, each affected event will correspond to a unique cluster of RAs. RA_d stores the relationship between each affected event $e_{di} \in E_d$ and the associated cluster of RAs in a cluster map. The cluster map is defined as $C_l : E_d \to 2^{RA^p}$ for transportation RAs; $C_l : E_d \to 2^{RA^m}$ for transformation RAs. 2^{RA^p} and 2^{RA^m} denote the power sets of the transportation and transformation RAs in the manufacturing system, respectively. All of the RAs in this map, $RA_{clst} = \{C_l(e_d) : e_d \in E_d\}$, are clustered RAs for a given set of affected events. A detailed cluster formulation process is described in Section 4.1.

Sequential RAs In a part's production schedule T_s, the event sequence s_e corresponds to a sequence of RAs that perform events sequentially on the part. Therefore, replacing the disrupted RA cannot guarantee that the new schedule is executable. For the example shown in Fig. 4, if $Machine1$ breaks down, $Robot1$ should not move the part to $Machine1$. Hence, to generate a feasible schedule, the RA_d should have knowledge of its sequential RAs. A sequential RA is an RA that performs the event before or after e_d in the event sequence s_e of the part's production schedule T_s. Therefore, each scheduled event corresponds to one or two sequential RAs. An RA stores the information about the sequential RAs in a map that relates the affected event to specific RAs: $S_q : E^c \to RA^2$, where RA^2 denotes a set containing one or two sequential RAs.



Disrupted RA
Class RA
Collaborative RA
Loop [For all affected event]
Loop [Stop condition]
1a: Rescheduling request broadcast
1b: Self-check: attributes and availability
Form cluster
Cluster Search
Need other RAs to form new schedule?
Need: 2a: event request
2b: Self-check: capabilities, attributes and availability
Not need
Schedule Generation
3a: Decision-making
3b: Notify decision to the selected RAs
3c: Execution
3c: Execution
Decision-Making

Fig. 5. The coordination of RAs for rescheduling process

Collaborative RAs To represent the collaboration between RAs, Kovalenko et al. (2019a) introduced collaborative RAs, which have shared states X_s in the capabilities model. For the example shown in Fig. 4, $Machine1$ and $Robot1$ share the state x^l_{m1}. Therefore, each shared state $x_s \in X_s$ corresponds to a set of RAs that contain the same state x_s. Each RA stores the relationship between a shared state $x_s \in X_s$ and the associated set of RAs in a collaborative map. The map is defined as $N_g : X_s \to 2^{RA}$, where RA is the set of all the collaborative RAs.

Since the clustering and sequential RAs are defined according to the affected event set E_d, the RA_d's knowledge of clustering and sequential RAs will be added to or removed from the RA_d's environment model when an event needs to be rescheduled or the rescheduling process for an event is completed. The collaborative RAs are defined based on the shared states X_s in an RA's capabilities model. Therefore, the collaborative RAs are initialized with the capabilities model and will remain static unless the capabilities model changes.

4. RESCHEDULING PROCESS

This section describes the development and utilization of a dynamic RA cluster to enable rescheduling via RA coordination.

An overview of the rescheduling process via RA coordination is shown in Fig. 5. The goal of the rescheduling process is to find alternative resources to perform the affected events of the disrupted RA. The proposed method reschedules the affected events one by one in the order of their scheduled start time in the initial schedule. Therefore, the rescheduling decision-making is realized on the fly in a dynamic manner. Since each affected event follows the same rescheduling steps; for simplicity, the following description focuses on one affected event e_d.

4.1 Cluster search

Once a disrupted RA (RA_d) identifies the resource disruption, the RA state goes to *Down*, and the process of forming a cluster is initiated. In this phase, RA_d searches for the RAs that have the capabilities to perform the affected event e_d. A broadcast technique is used for RAs to communicate information (Azuma et al., 2012). RA_d broadcasts the rescheduling request, $R_q = (e_d, R_t)$, where R_t denotes the task requirements of e_d.

Based on the resource capabilities, only the RAs in the same class with RA_d, i.e. transportation (RA^p) or transformation RAs (RA^m), have access to the broadcast request. To replace RA_d and perform e_d, an RA needs to satisfy the following requirements: 1) the RA must contain the affected event e_d in their capabilities model; 2) the RA must have the physical resource attributes that can satisfy the task requirements R_t:

$$\exists\, e_{new} \in E_{cl},\ s.t.\ e_{new} = e_d,\ R_t \subseteq A_t(e_{new}) \tag{3}$$

where E_{cl} and $A_t(e_{new})$ represents the set of events and the resource attributes of e_{new} in the capabilities model of the RA.

As such, the RAs that satisfy Eqn. 3 form a cluster for RA_d with respect to the affected event e_d:

$$C_l(e_d) = \{RA^{p/m} \mid e_d \in E_{cl},\ R_t \subseteq A_t(e_{new})\} \tag{4}$$

The RAs in the cluster $C_l(e_d)$ are the RAs that can perform e_d while satisfying the task requirements of e_d. These RAs in $C_l(e_d)$ will respond to RA_d with their resource attributes $A_t(e_{new})$ and available times $T_a(e_{new})$ for the event e_{new}.

4.2 Schedule generation

Once the cluster $C_l(e_d)$ is formed, each RA in the cluster follows the same steps to generate new schedules. For simplicity, the following description focuses on one RA (RA_{cl}) in the cluster $C_l(e_d)$. As mentioned in Eqn. 2, a new event sequence, s_{new}, needs to achieve the transitions from x_{prior} to x_{post}:

$$Tr((x^l_{prior}, x^c_{prior}), s_{new}) = (x^l_{post}, x^c_{post}) \tag{5}$$

The event, e_{new}, belongs to s_{new}. However, since e_{new} can only achieve either location or physical composition transitions, RA_{cl} needs to check whether e_{new} satisfies Eqn. 5.

In the case that RA_d is a transportation RA, the affected event e_d only achieves part location transition. e_d does not change the physical composition of the part (i.e. $x^c_{prior} = x^c_{post}$) Therefore, the transition for physical composition is not required in the rescheduling process. In this case, the Eqn. 5 is rewritten as:

$$Tr(x^l_{prior}, e_{new}) = x^l_{post},\ x^c_{prior} = x^c_{post} \tag{6}$$

Therefore, replacing e_d by e_{new} can achieve the required transition. In this scenario, RA_{cl} can replace RA_d by itself, and no further coordination is needed to form a feasible new schedule.

However, in the case that RA_d is a transformation RA, RA_d will instruct sequential RAs to not perform the events of moving the part into and out of RA_d. In this case, simply replacing e_d by e_{new} cannot fulfill the required transitions in Eqn. 5. To find other events needed to form s_{new}, the states $x_{prior} = (x^l_{prior}, x^c_{prior})$ and $x_{post} = (x^l_{post}, x^c_{post})$ are identified as follows:

- x^l_{prior}: the location before the part is moved to RA_d
- x^c_{prior}: the physical composition before RA_d performs e_d
- x^l_{post}: the location after the part is moved out RA_d
- x^c_{post}: the physical composition after RA_d performs e_d

By Eqn. 3, the RA_{cl} can only achieve the required physical composition changes by performing event e_{new}:

$$Tr(x^c_{prior}, e_{new}) = x^c_{post} \tag{7}$$

To form a feasible new schedule, the location changes from x^l_{prior} to x^l_{cl} and x^l_{cl} to x^l_{post}, where x^l_{cl} denotes the location of RA_{cl}, need to be found. Therefore, RA_{cl} sends request to its collaborative RAs, $N_g(x^l_{cl})$, to check their capabilities:

$$\begin{aligned} &\exists\, RA_{ng1}, RA_{ng2} \in N_g(x^l_{cl}),\ e_{ng1} \in E_{ng1},\ e_{ng2} \in E_{ng2} \\ &\text{s.t. } Tr(x^l_{prior}, e_{ng1}) = x^l_{cl},\ Tr(x^l_{cl}, e_{ng2}) = x^l_{post} \end{aligned} \tag{8}$$

where E_{ng} represents the event set in the RA's capabilities model. RA_{ng1} performs e_{ng1} to deliver the part to RA_{cl}, and RA_{ng2} performs e_{ng2} to take the part out of RA_{cl}.

Therefore, different combinations of RA_{cl} and RA_{ng} form several new schedules, denoted by $S_{new} = \{s_{new1}, ..., s_{newq}\}$, to replace RA_d in performing the event e_d. For each schedule $s_{new} =< e_{n1} e_{n2} ... e_{ns} >$, the post event should always occur after the prior event ends:

$$t_{s,i} + T(e_{ni}) \leq t_{s,j},\ 1 \leq i < j \leq s \tag{9}$$

where $t_{s,i}$ and $T(e_{ni})$ represent the start time and time cost of event e_{ni}, respectively. Note that in the time interval $[t_{s,i} + T(e_{ni}),\ t_{s,j}]$, the part remains with the RA that performs e_{ni} until the RA that performs e_{nj} is available.

4.3 Decision-making Process

New schedules found The RA_d will choose a new schedule from $S_{new} = \{s_{new1}, ..., s_{newq}\}$ based on the cost. The cost of a new schedule s_{new} is evaluated by n performance metrics $M = \{m_1, m_2, ..., m_n\}$ and corresponding weights, $\alpha = [\alpha_1 \alpha_2 \cdots \alpha_n]$. The metrics could be operation time, finish time, energy cost, resolution, etc., which are pre-defined. For every performance metric, there is a nominal cost function $C(e) = [C_1(e)\ C_2(e)\ \cdots\ C_n(e)]^T$, which maps the nominal cost to perform event e without violating task requirements. The total cost of a new schedule $s_{new} =< e_{n1} e_{n2} ... e_{ns} >$ is calculated as:

$$\mathcal{K}(s_e) = \sum_{i=1}^{s} \alpha C(e_{ni}) \tag{10}$$

where a new schedule that replaces RA_d is determined by solving the minimization problem:

$$s^*_{new} = \underset{s_{new} \in S_{new}}{\arg\min}\ \mathcal{K}(s_{new}) \tag{11}$$

where s^*_{new} is the event sequence minimizing the costs in S_{new}.

As shown in Fig. 5, the RA_d informs the RAs associated with the events in s^*_{new}. These RAs update their schedule information in the capabilities model and provide high-level control for their associated physical resource to perform the events.

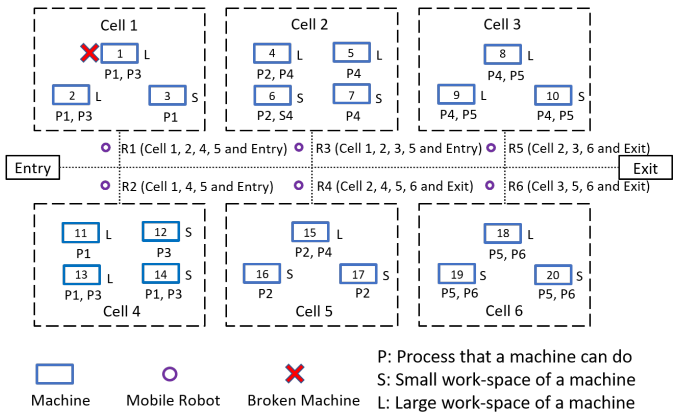
No schedules found If no schedule is found within the required constraints, the RA_d will request the central controller of the manufacturing system to complete the rescheduling.

5. CASE STUDY

In this section, the set-up of the system used for the case study is described and the results are provided.

5.1 Case Study Set-up

The Repast Symphony (RepastS) platform (Macal and North, 2006) can be used to model and simulate a multi-agent system.



Cell 1
1 L
P1, P3
2 L
P1, P3
3 S
P1
Cell 2
4 L
P2, P4
5 L
P4
6 S
P2, S4
7 S
P4
Cell 3
8 L
P4, P5
9 L
P4, P5
10 S
P4, P5
Entry
Exit
R1 (Cell 1, 2, 4, 5 and Entry)
R3 (Cell 1, 2, 3, 5 and Entry)
R5 (Cell 2, 3, 6 and Exit)
R2 (Cell 1, 4, 5 and Entry)
R4 (Cell 2, 4, 5, 6 and Exit)
R6 (Cell 3, 5, 6 and Exit)
11 L
P1
12 S
P3
13 L
P1, P3
14 S
P1, P3
Cell 4
15 L
P2, P4
16 S
P2
17 S
P2
Cell 5
18 L
P5, P6
19 S
P5, P6
20 S
P5, P6
Cell 6
Machine
Mobile Robot
Broken Machine
P: Process that a machine can do
S: Small work-space of a machine
L: Large work-space of a machine

Fig. 6. The set-up for the case study

Table 1. Tick costs for machine processes

Process	P1	P2	P3	P4	P5	P6
Tick Cost	150	120	110	100	170	200

In this work, the set-up for the case study is a modified version of previous work (Kovalenko et al., 2019b). The simulated system contains 20 machines that are connected via a network of 6 mobile robots. The capabilities of the machines and mobile robots are labeled near the icons in Fig. 6. For example, the labels for machine 1 indicate it can perform the processes P1 and P3 and its workstation space is large. For mobile robot R1, its labels indicate that it can move parts between any machines located in Cell 1, 2, 4, 5 and the Entry buffer. The time costs in ticks (RepastS unit of time) for the machine processes are shown in Table. 1. The mobile robots take 18-29 ticks to move the parts between different locations.

Two types of parts are fed into the system, with each has the following process requirements: 1) S-part: P1 $\rightarrow$ P2 $\rightarrow$ P3 $\rightarrow$ P6, and 2) L-part: P1 $\rightarrow$ P3 $\rightarrow$ P4 $\rightarrow$ P5, respectively. The machines with label L can operate both L-parts and S-parts, while the machines with label S can only operate S-parts. Parts enter the system from the Entry buffer and leave the system through the Exit buffer after completing the desired processes.

5.2 Case Study and Results

In this case study, 50 L-parts and 50 S-parts are fed into the system alternatively and the simulation is stopped at tick 7000. Designed to simulate a rescheduling scenario, machine 1 breaks down at tick 500, remaining off-line for 3000 ticks.

The simulation included the following scenarios: S1) No machine breakdown; S2) No rescheduling to react to machine breakdown; S3) Rescheduling based on a proximity cluster; S4) Rescheduling based on a capabilities cluster.

The system performance under the different scenarios is evaluated using the following metrics:

- Throughput: the number of the completed parts within a specified period of time
- Resource utilization: the working time of each machine within a specified period of time

Figures 7 and 8 show the throughput for the two different types of parts. S.1 represents the optimal throughput of both part types if there are no disruptions in the system. The vertical line is added to check the throughput at the end time of S1.

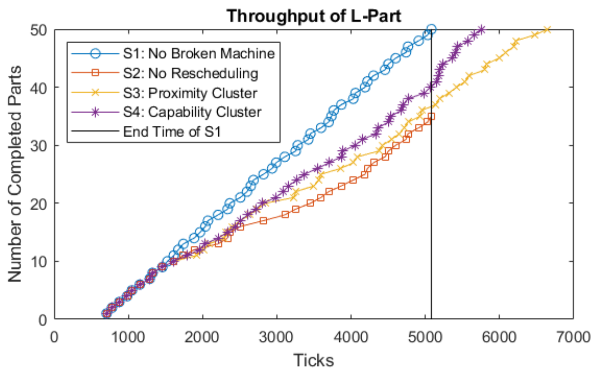


Fig. 7. The throughput of the L-parts

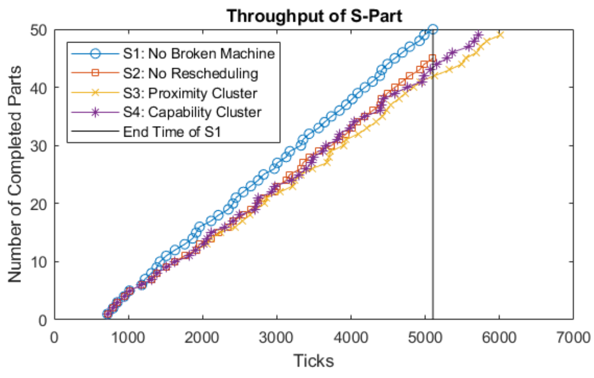


Fig. 8. The throughput of the S-parts

The breakdown heavily impacts the throughput of the L-parts in the system. As shown in Fig. 7, S2 leads to the worst throughput with the completed part rate - of 70% compared to S1. At the tick when S1 ends, both proximity (S3) and capability clusters (S4) achieve production recovery by applying a rescheduling process with the completed part rate - of 72% and 80%, respectively. However, if time permits, both S3 and S4 are able to complete all the parts with increasing time of 30.7% and 13.2%, respectively. The results show that the proposed rescheduling method with capability cluster provides better throughput recovery. By defining the RA capability cluster, the disrupted RA can directly coordinate with RAs that have the same capabilities without the physical distance limitation of the proximity clustering method. In this case, more machines are used to replace the broken machine, reducing the waiting time of the parts.

Figure 8 shows the effect that the breakdown had on S-parts in the system. Only 7 S-parts are affected by this disruption. In this case, generating new schedules for the affected parts increases the occupancy of other resources, which lead to the time delay in the parts entering the system. Therefore, the rescheduling methods cannot compensate to improve the production rate. However, if time permits, both S3 and S4 are able to complete 49 parts with increasing time of 17.8% and 12.7%, respectively.

Figure 9 visualizes how the different methods made decisions regarding resource utilization in the case study. In this figure, the total processing times of the machines in Cells 1 and 4 are shown. By using the proximity cluster (S3), only the machines in Cell 1 were identified as a replacement for the broken machine. By using the capability clustering (S4), the operations of the broken machine were redistributed to machines in both Cell 1 and 4. This result shows that applying capabilities-based clusters provides a more balanced approach during reconfiguration.

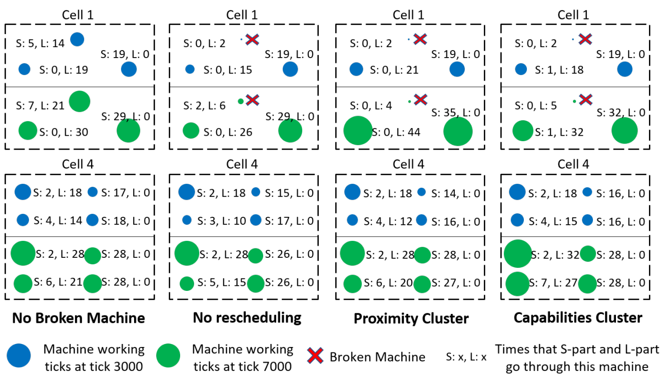


Fig. 9. The working times of the machines in Cell 1 and Cell 4

The case study presented above showcases the feasibility and performance of the proposed rescheduling strategy via coordination using clustered RAs based on their capabilities. Though there is no production optimality preserved, the proposed method achieves a better recovery of the production in the occurrence of a resource disruption. The rescheduling performance is affected by the severity of the disruption.

6. CONCLUSION

A multi-agent control strategy has been proposed to solve dynamic rescheduling problems in manufacturing systems. In this work, the design of an extended RA knowledge base and a rescheduling strategy via RA coordination are presented. The proposed RA knowledge base contains the belief models of the RA state, capabilities, and environments. Based on this model, a capabilities-based clustering scheme is developed to enable dynamic rescheduling via RA coordination to adapt to a resource disruption. The feasibility and performance of the proposed rescheduling strategy are showcased in a simulation study, where the proposed strategy provides better recovery of throughput and more balanced resource utilization. Future work will include analyzing the effects of resource redundancy and available capacity on the performance of the proposed strategy, as well as exploring the system conditions when switching to a centralized method can reduce the rescheduling effort.

REFERENCES

Abumaizar, R.J. and Svestka, J. (1997). Rescheduling job shops under random disruptions. *International Journal of Production Research*, 35(7), 2065–2082.

Azuma, S., Tanaka, Y., and Sugie, T. (2012). Multi-agent consensus under communication-broadcast mixed environment. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, 94–99. doi:10.1109/CDC.2012.6426830.

Balta, E.C., Kovalenko, I., Spiegel, I.A., Tilbury, D.M., and Barton, K. (2021). Model predictive control of priced timed automata encoded with first-order logic. *IEEE Transactions on Control Systems Technology*.

Barata, J. and Camarinha-Matos, L.M. (2003). Coalitions of manufacturing components for shop floor agility the cobasa architecture. *Int. J. Netw. Virtual Organ.*, 2(1), 50–77.

Cardin, O., Trentesaux, D., Thomas, A., Castagna, P., Berger, T., and Bril El-Haouzi, H. (2017). Coupling predictive scheduling and reactive control in manufacturing hybrid control architectures: state of the art and future challenges. *Journal of Intelligent Manufacturing*, 28(7), 1503–1517.

Farid, A.M. and Ribeiro, L. (2015). An Axiomatic Design of a Multiagent Reconfigurable Mechatronic System Architecture. *IEEE Transactions on Industrial Informatics*, 11(5), 1142–1155. doi:10.1109/TII.2015.2470528.

Kovalenko, I., Barton, K., and Tilbury, D. (2017). Design and Implementation of an Intelligent Product Agent Architecture in Manufacturing Systems. 9(3), 10–16.

Kovalenko, I., Ryashentseva, D., Vogel-Heuser, B., Tilbury, D., and Barton, K. (2019a). Dynamic Resource Task Negotiation to Enable Product Agent Exploration in Multi-Agent Manufacturing Systems. *IEEE Robotics and Automation Letters*, 4(3), 2854–2861. doi:10.1109/LRA.2019.2921947.

Kovalenko, I., Tilbury, D., and Barton, K. (2019b). The model-based product agent: A control oriented architecture for intelligent products in multi-agent manufacturing systems. *Control Engineering Practice*, 86, 105–117.

Kumar, A., Luthra, S., Mangla, S.K., and Kazançoğlu, Y. (2020). Covid-19 impact on sustainable production and operations management. *Sustainable Operations and Computers*, 1, 1–7.

Leitão, P. (2009). Agent-based distributed manufacturing control: A state-of-the-art survey. *Engineering Applications of Artificial Intelligence*, 22(7), 979 – 991.

Lepuschitz, W., Zoitl, A., Vallée, M., and Merdan, M. (2011). Toward Self-Reconfiguration of Manufacturing Systems Using Automation Agents. 41(1), 52–69.

Li, X., Wang, B., Liu, C., Freiheit, T., and Epureanu, B.I. (2020). Intelligent manufacturing systems in COVID-19 pandemic and beyond: Framework and impact assessment. *Chinese Journal of Mechanical Engineering*, 33(1), 1–5.

Macal, C.M. and North, M.J. (2006). Introduction to agent-based modeling and simulation. In *Proceedings of the MCS LANS Informal Seminar*.

Maturana, F., Shen, W., and Norrie, D. (1999). Metamorph: An adaptive agent-based architecture for intelligent manufacturing. *International Journal of Production Research*, 37(10), 2159–2173.

Qamsane, Y., Balta, E.C., Moyne, J., Tilbury, D., and Barton, K. (2019). Dynamic rerouting of cyber-physical production systems in response to disruptions based on SDC framework. In *2019 American Control Conference (ACC)*, 3650–3657.

Rodrigues, N., Oliveira, E., and Leitão, P. (2018). Decentralized and on-the-fly agent-based service reconfiguration in manufacturing systems. *Computers in Industry*, 101(October 2017), 81–90. doi:10.1016/j.compind.2018.06.003.

Shen, W., Wang, L., and Hao, Q. (2006). Agent-based distributed manufacturing process planning and scheduling: a state-of-the-art survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 36(4), 563–577.

Vieira, G.E., Herrmann, J.W., and Lin, E. (2003). Rescheduling manufacturing systems: a framework of strategies, policies, and methods. *Journal of scheduling*, 6(1), 39–62.

Wong, T.N., Leung, C.W., Mak, K.L., and Fung, R.Y.K. (2006). Integrated process planning and scheduling/rescheduling—an agent-based approach. *International Journal of Production Research*, 44(18-19), 3627–3655.

Zhang, S. and Wong, T.N. (2017). Flexible job-shop scheduling/rescheduling in dynamic environment: a hybrid MAS/ACO approach. *International Journal of Production Research*, 55(11), 3173–3196.