Computer Networks 101 (2016) 158–168

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Towards smart factory for industry 4.0: a self-organized multi-agent system with big data based feedback and coordination

Shiyong Wang[a], Jiafu Wan[a,*], Daqiang Zhang[b], Di Li[a], Chunhua Zhang[a]

[a] School of Mechanical and Automotive Engineering, South China University of Technology, Guangzhou, China
[b] School of Software Engineering, Tongji University, Shanghai, China

ARTICLE INFO

Article history:
Received 18 June 2015
Revised 18 November 2015
Accepted 15 December 2015
Available online 6 January 2016

Keywords:
Industry 4.0
Smart factory
Cyber-physical system
Multi-agent system
Deadlock prevention

ABSTRACT

The proliferation of cyber-physical systems introduces the fourth stage of industrialization, commonly known as Industry 4.0. The vertical integration of various components inside a factory to implement a flexible and reconfigurable manufacturing system, i.e., smart factory, is one of the key features of Industry 4.0. In this paper, we present a smart factory framework that incorporates industrial network, cloud, and supervisory control terminals with smart shop-floor objects such as machines, conveyers, and products. Then, we provide a classification of the smart objects into various types of agents and define a coordinator in the cloud. The autonomous decision and distributed cooperation between agents lead to high flexibility. Moreover, this kind of self-organized system leverages the feedback and coordination by the central coordinator in order to achieve high efficiency. Thus, the smart factory is characterized by a self-organized multi-agent system assisted with big data based feedback and coordination. Based on this model, we propose an intelligent negotiation mechanism for agents to cooperate with each other. Furthermore, the study illustrates that complementary strategies can be designed to prevent deadlocks by improving the agents' decision making and the coordinator's behavior. The simulation results assess the effectiveness of the proposed negotiation mechanism and deadlock prevention strategies.

1. Introduction

The application of automation and information systems such as enterprise resource planning (ERP) and manufacturing execution system (MES), significantly improves factory productivity. However, the current industrial production faces many critical challenges. The end users continuously require highly customized products in small batches. Moreover, the current production paradigm is not sustainable [1]. On one hand, the impact of industrial production on environment in terms of global climate warming and environmental pollution is severe. On the other hand, the consumption of non-renewable resources such as petroleum and coal increases and the industry suffers an ever-shrinking workforce supply because of population aging. Therefore, industrial processes need to achieve high flexibility and efficiency as well as low energy consumption and cost. Many advanced manufacturing schemes have already been proposed aiming to overcome the drawbacks of the current production lines, e.g., the flexible manufacturing system (FMS) and the agile manufacturing system (AMS). Among these schemes,

* Corresponding author. Tel.: +86 13143737141.
E-mail address: jiafuwan_76@163.com (J. Wan).

http://dx.doi.org/10.1016/j.comnet.2015.12.017
1389-1286/

the multi-agent system (MAS) is the most representative one [2], where the manufacturing resources are defined as intelligent agents that negotiate with each other to implement dynamic reconfiguration to achieve flexibility. However, due to the lack of some global coordination, the proposed MAS schemes could not handle complex manufacturing systems in an efficient way [3].

Nowadays, the emerging cyber-physical system (CPS) presents a significant opportunity to implement smart manufacturing. The CPS can arm the MAS with the emerging technologies (e.g., Internet of Things (IoT) [4–6], wireless sensor networks (WSN) [7,8], big data [9], cloud computing [10–12], embedded systems [13], and mobile Internet [14]). Consequently, a strategic initiative called "Industrie 4.0" (Industry 4.0) has been proposed and adopted by the German government as part of the "High-Tech Strategy 2020 Action Plan" [15]. Similar strategies have also been proposed by other main industrial countries, e.g., "Industrial Internet" [16] by USA and "Internet +" [17] by China. The Industry 4.0 describes a CPS oriented production system [18–20] that integrates production facilities, warehousing systems, logistics, and even social requirements to establish the global value creation networks [21].

The smart factory is an important feature of Industry 4.0 that addresses the vertical integration and networked manufacturing systems for smart production [15]. For smart factory to be implemented, it should combine the smart objects with big data analytics. The smart objects can dynamically reconfigure to achieve high flexibility whereas the big data analytics can provide global feedback and coordination to achieve high efficiency. Therefore, the smart factory might be able to produce customized and small-lot products efficiently and profitably. Considering that FMS tries to allocate manufacturing resources to a family of product types with a kind of central computerized controller [22], and MAS relies on autonomous agents to reconfigure dynamically [23], the smart production enabled by smart factory features high interconnection, mass data, and deep integration.

In this article, based on a novel smart factory framework that integrates the autonomous agents with big data based feedback and coordination [24], we focus on the key algorithms that can operate the smart factory. The contributions of this study mainly include two aspects. First, we model the smart shop-floor objects such as machines, conveyers, and products as agents, and propose an intelligent negotiation mechanism for them to cooperate with each other. Second, we identify the conditions that enable deadlocks, and design four complementary strategies to prevent the deadlocks by improving agents' decision making and the coordinator's behavior. Finally, we test and validate the proposed negotiation mechanism and deadlock prevention strategies through numerical simulation.

The article is organized as follows. Section 2 introduces the framework and operational mechanism of smart factory. Section 3 proposes an intelligent negotiation mechanism to enable self-organization of agents. Section 4 describes four deadlock prevention strategies. Section 5 reports the main simulation results. Finally, Section 6 concludes the work.

2. System architecture and operational mechanism

The smart factory is a manufacturing cyber-physical system that integrates physical objects such as machines, conveyers, and products with information systems such as MES and ERP to implement flexible and agile production. In this section, a framework for smart factory is proposed and its operational mechanism is investigated.

2.1. System architecture

Figure 1 summarizes the smart factory framework consisting of four tangible layers, namely physical resource layer, industrial network layer, cloud layer, and supervisory control terminal layer. The physical resources are implemented as smart things which communicate with each other through the industrial network. The integrated information system exists in the cloud which collects massive data from the physical resource layer and interacts with people through supervisory control terminals [14]. Thus, the tangible framework enables a networked world for intangible information to flow freely. This actually forms a CPS where physical objects and informational entities are deeply integrated.

2.2. Operational mechanism

From the perspective of the control engineer, the smart factory can be viewed as a dual closed-loop system, as shown in Fig. 1. One loop consists of physical resources and cloud, while the other loop consists of supervisory control terminals and cloud.

The smart shop-floor object has 3C's capabilities and it is autonomous and social. The term autonomous means that the smart object makes decisions by itself and no other entities can directly control its behavior. The term social means that the smart objects understand and share a common set of knowledge and negotiate according to a common set of rules. Therefore, a society of smart objects can yield a highly flexible manufacturing system, i.e. a self-organized and reconfigurable system that seems to be humanoid or smart.

Through cooperation, the smart objects try to align their behaviors to approach a system-wide goal. However, the system performance is generally not the optimum. This is because the smart objects make decisions based on local information. Thus, as to manufacturing, for example, load may not be balanced, efficiency may not be the highest, or deadlocks may occur. One of the big data analytics blocks (the coordinator) can solve this issue. The smart machines communicate their state and process information to the block, and the distributed sensors transfer their sensed data to the block as well. Therefore, the global state of the smart factory can be extracted from the massive real-time system information. Based on the powerful computing ability, the block processes these big data in time to coordinate the behaviors of the distributed smart objects, and to feedback performance indicators to the self-organized network. Therefore, this global optimization can assist the autonomous agents to achieve a better performance.

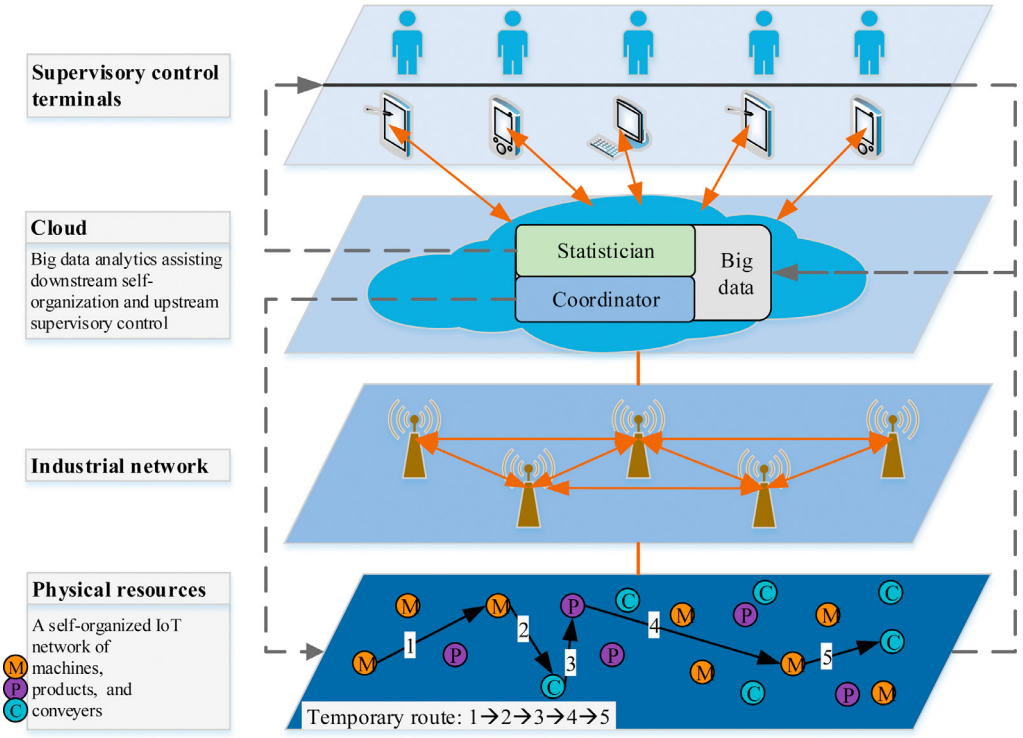

Fig. 1. Framework of the smart factory of Industry 4.0.

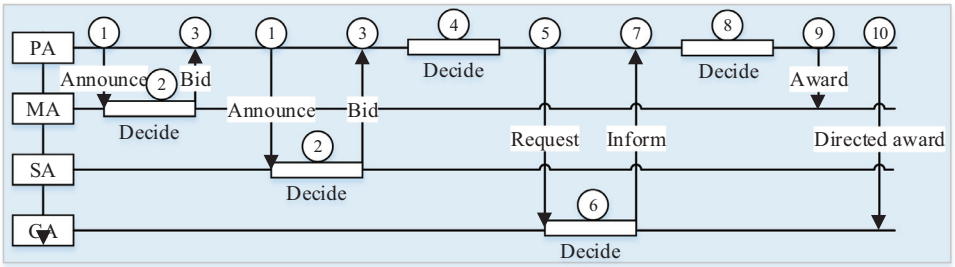

Fig. 2. Negotiation process among agents.

The big data analytics can also provide various statistical results to the users for the purpose of supervisory control and the users can adjust system configuration through the field or remote terminals.

3. Self-organized manufacturing based on distributed decision making and intelligent negotiation of smart agents

This study classifies physical shop-floor objects into four types of agents. The Machining Agent (MA) represents machines that perform machining or testing operations; the Conveying Agent (CA) represents the devices (such as conveyers, robots or AGVs) that move the products; the Product Agent (PA) represents products that are processed by the system; and the Supplementary Agent (SA) represents the buffers that temporarily hold PA agents. In this section, a negotiation mechanism that leads agents to cooperate with each other is developed. Figure 2 shows the overall negotiation process. For each operation of the sequence, the PA agent first negotiates with the MA and SA agents to determine one of them to perform or buffer the operation. Then, it negotiates with the CA agents to determine some of them to form a route that transports the PA agent from the current position to the destination. The negation mechanism is based on contract net protocol (CNP) [25]: the PA agent acts as the manager to initiate the tasks for operations while the other types of agents act as the contractors to compete for the tasks.

3.1. Overall negotiation process among agents

As Fig. 2 shows, the process consists of ten steps. The PA agent acts as the manager to initiate a task announcement message to the MA and SA agents (step 1) that will estimate the requirements against their capabilities (step 2,

Table 1
Manager' decisions based on received messages.

No.	Received messages	Decisions
1	No bid or immediate response messages are received.	The manager may initiate negotiations again. However, this situation most likely indicates that no agents support the operation or the network malfunctions. Therefore, the coordinator should be informed or a local warning indicator should be triggered.
2	Only the immediate response messages are received.	Right now, all the contractors are busy. The negotiation should be started again after a predefined delay.
3	One or more bid messages are received.	Continue the negotiation process until the task is dispatched.

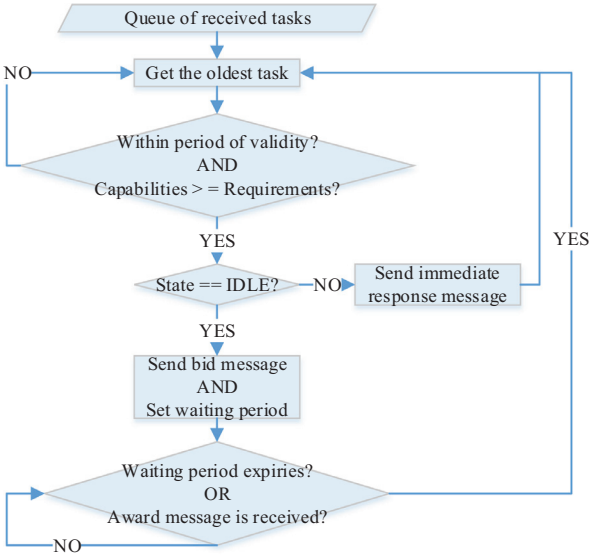

Fig. 3. Contractor's decision-making process on bidding.

discussed in Section 3.2). If the requirements can be satisfied, the bid or immediate response messages, depending on whether the contractors are idle or not, will be sent out to the manager (step 3). Then, the manager has to make decisions based on the received messages, as shown in Table 1 (step 4).

Conveying route should be determined for each bidder through the negotiation with the CA agents (steps 5 to 7 discussed in Section 3.3). The manager should rank the bidders' characteristics according to some predefined rules (step 8 discussed in Section 3.4). The highest ranked bidder will be selected. The manager will inform the winner via an award message (step 9) and will also pose a directed award message to the first CA agent of the route to start transporting the PA agent (step 10).

3.2. Contractor's decision-making process on bidding

The contractors described in this subsection involve the MA and SA agents. A contractor may receive multiple task announcement messages from different managers. The message queue buffers the messages. The messages that enter the queue earlier are processed earlier, i.e., the contractor does not rank these messages. Figure 3 describes the contractor's decision-making process: the contractor bids for a task using bid message when it is capable and idle. Moreover, it informs the manager via immediate response message when it is capable but busy, in order to help the manager make the right decision.

3.3. Negotiation process to determine a conveying route

The CA agents are responsible for moving PA agents from one agent (MA, CA, and SA type) to another. A CA agent can interact with a group of other agents that form an accessibility set. Based on the operation range, different CA agents, in general, have different accessibility sets. However, different sets may intersect with each other and a CA agent may have other CA agents as its members; therefore, a chain of CA agents, forming a route, can be determined to transfer the PA agent from its current position to the destination (Fig. 4). Each CA agent knows its accessibility set, and each agent knows which sets it belongs to. The PA agent starts the searching process and the CA agents push forward the searching process until a valid route is found out. In this particular case, information exchange adopts the REQUEST-INFORMATION messages defined by the CNP.

3.4. Manager's ranking rules to select a bidder

The manager should select a contractor from two or more competitive bidders. Different methods based on single rule or weighted multiple rules have been proposed in the literature. According to the presented design, the bidders are ranked using layered multiple rules, i.e., the rules have different priorities and the high-priority rules are checked against before the ones with lower priority. Once a bidder gets higher score than the other bidders, it wins the task and the ranking process is aborted. Table 2 summarizes the rules. Although the manager's ranking process is based on local decision-making, we aim to approach the global optimization by introducing the reward mechanism, preferring the shortest conveying route, and saving more precise and faster resources to other operations.

4. Deadlock prevention by improving the agents' decision making and the coordinator's behavior

Deadlock problems that originated from computer engineering can also occur in manufacturing environment. It is a hot topic in the research of FMS, and PetriNet based methods have been widely used [26]. Smart production consists of autonomous agents instead of a central

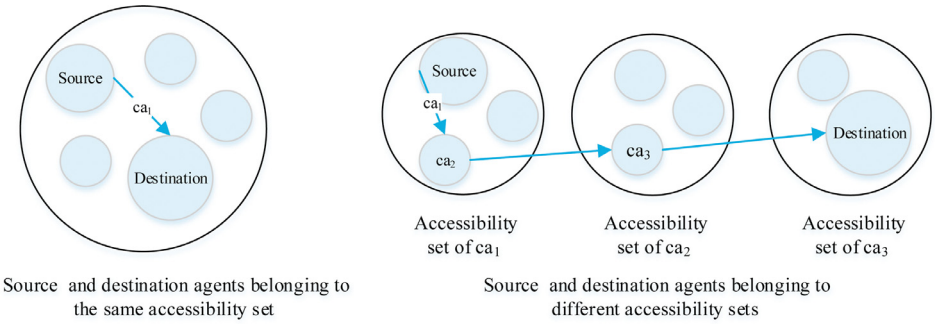

Fig. 4. Conveying route formed by a chain of CA agents.

Table 2
Rules for ranking bidders.

Priority (from high to low)	Description of rules	Comments
1	MA agents rank higher than SA agents.	MA agents really process the products while the SA agents just buffer products.
2	Bidders that have won the contracts for the same operation of other PA agents and have completed the operation successfully and timely rank higher than other bidders.	This reward mechanism utilizes agents' experience to simplify the selection process and to reduce the production dynamics.
3	Bidders with shorter conveying routes rank higher than others.	To reduce conveying time.
4	Bidders with lower machining or test precision rank higher than those with higher precision.	It is noted that all the bidders can satisfy the precision requirements. This rule saves more precise resources to other operations.
5	Bidders that can complete the bidding operation a little faster than the last operation, rank higher than those that are much faster.	This rule saves more efficient resources for others.

controller, making analysis and development of control algorithms very different from FMS. Therefore, novel deadlock prevention strategies should be investigated to address the features of smart production such as distribution, autonomy, and negotiation. This section describes the causes that lead to deadlocks and we propose four complementary strategies to prevent deadlocks. As the deadlock is a kind of system behavior, both the agents and the coordinator should be involved in the prevention process.

The strategies 1 and 2 use the functional redundancy of MA agents to break the loops. However, as the redundancy is generally not enough, loops may not be completely broken. Strategy 3 uses the SA agents to break the mutual exclusive use of the agents. Because the SA agents may also be not enough to completely break the loops, strategy 4 that uses congestion control is designed to prevent deadlocks. Strategy 4 does not depend on resources, thus it can effectively prevent any remaining deadlocks, but it is less efficient compared to the other strategies. Therefore, to balance resource demand and system efficiency, the four strategies should be used together. The strategy 1 should be applied first and then the strategy 2 should be applied. If loops still exist, strategy 3 can be applied if buffers are available. The strategy 4 should be used after strategy 3 or strategy 2 depending on whether strategy 3 has been used or not.

4.1. Conditions for deadlocks to occur

An MA agent has one or more machining or testing sub functions. Based on the number of sub functions that an MA agent owns, the MA agents can be classified into single-functional (MA) agent, with only one sub function, and multi-functional (MA) agent, which at least has two sub functions. A PA agent specifies a fixed sequence of operations, i.e., [1.Type1, 2.Type2, …, *m.Typen*]. The notation *m.Typen* denotes an individual operation, where *m* is the operation number and *Typen* is the operation type name. The operation numbers increase from 1 to *m*, denoting the operation sequence. If only one operation belongs to an operation type, we define the operation type as the single-occurrence (operation) type. Moreover, if two or more operations belong to the same operation type, we define the operation type as the multi -occurrence (operation) type.

The multi-functional agent can cause deadlocks, as shown in Fig. 5. The needed operation sequence is [1.A, 2.B, 3.C]; one MA agent ma_1 undertakes types A and C, while ma_2 undertakes type B. The multi-functional agent ma_1 leads to a loop generating a deadlock. The pa_1 enters the ma_1 for 1.A, and when finished, it proceeds to the ma_2 for 2.B. At this moment, the ma_1 is idle, thus the pa_2 enters ma_1 for its 1.A. When 2.B of pa_1 and 1.A of pa_2 are finished, the pa_1 needs ma_1 for its 3.C whereas the pa_2

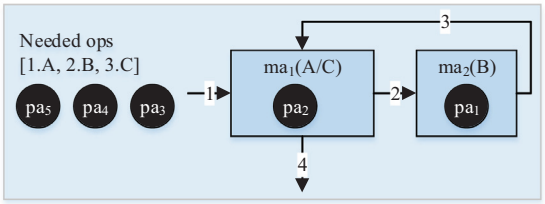

Fig. 5. Deadlocks caused by multi-functional agent.

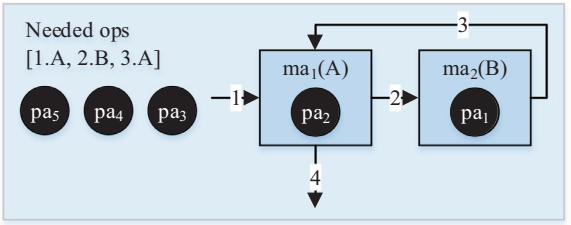

Fig. 6. Deadlocks caused by multi-occurrence type.

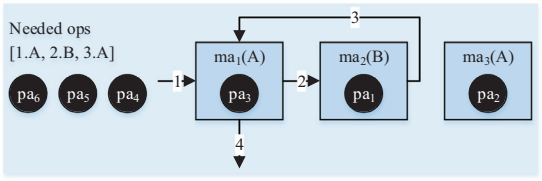

Fig. 7. Functional redundancy cannot always prevent deadlocks.

needs ma_2 for its 2.B, and then a deadlock occurs as the two PA agents request each other's occupied agents.

The multi-occurrence type can also cause deadlocks, as shown in Fig. 6. The needed operation sequence is [1.A, 2.B, 3.A]; the MA agent ma_1 undertakes type A, while ma_2 undertakes type B. The multi-occurrence type A leads to a loop around ma_1, generating a deadlock. The pa_1 enters ma_2 after ma_1 completes its 1.A. After a while, the pa_1 needs ma_1 again to perform 3.A. However, ma_1 is now occupied by pa_2 that wants ma_2 for its 2.B. Thus, pa_1 and pa_2 require each other's occupied agents and a deadlock occurs.

The functional redundancy may not resolve deadlocks as shown in Fig. 7. An additional MA agent ma_3 supporting the type A has been added to Fig. 6. The pa_1 enters the ma_1 for 1.A, and almost at the same time, the pa_2 enters the ma_3 for its 1.A. When pa_1 proceeds to ma_2 for 2.B, the pa_3 enters the ma_1 for its 1.A. All the three MA agents are now busy and a deadlock then occurs.

The multi-functional agents or the multi-occurrence types cause one agent to perform two or more operations. This actually enables loops of production route, which are essential constructs for deadlocks to occur because they enable a sequence of PA agents to request circularly each other's occupied MA agents. The loops are closed in agents that have multiple inward links. To facilitate discussion, in the following this kind of agents will be defined as critical agent**.** The critical agent has an equal number of inward and outward links, and the link number corresponds to the operation number. For example, the critical agent ma_1 in Fig. 7 has two inward links (numbered 1 and 3) and two outward links (numbered 2 and 4). The inward links indicate that the ma_1 tends to undertake the first and the third operation.

4.2. Strategy 1: Elimination of redundant sub functions of the multi-functional agents to break loops

A multi-functional agent supports multiple sub functions that correspond to multiple operation types. Therefore, the first strategy aims to reduce the number of sub functions of multi-functional agents using the functional redundancy (Fig. 8). In the route shown in Fig. 5, the multi-functional agent ma_1 causes a deadlock. Suppose that another multi-functional agent ma_3 that supports the operation types B and C is available. The redundancy is processed as follows:

(1) The sub function B of ma_3 is redundant with the single-functional agent ma_2; therefore, B is excluded from ma_3.
(2) The sub function C of ma_1 is redundant with that of ma_3; therefore, C is excluded from ma_1.
(3) After processing, ma_1 and ma_3 are dedicated to the operations 1.A and 3.C respectively. The loop is broken and deadlocks are prevented.

To sum up, the processing order of redundancy is as follows: 1) the sub functions that do not support the needed operation types of the PA agent are prohibited; 2) the sub functions that are redundant with the single-functional agents are prohibited; and 3) when multiple agents own the same sub functions, only one multi-functional agent is selected to keep this sub function. Figure 9 shows the algorithm of the process. The Unmarked Index (UMI) is defined as the number of allowed sub functions. The higher UMI agent tends to cause more complex loops and the agent that is processed first has more chance to reduce its UMI. Therefore, this algorithm first processes the agents with higher UMIs.

After applying this algorithm, the UMI for each agent is finally determined. According to the UMI, the multi-functional agents can be classified into three categories, as summarized in Table 3. Note that the UMI k ($k \geq 1$) agents only use the sub functions that are allowed; and the allowed sub functions are different from each other and from the functions of the single-functional agents, i.e., the sub function redundancy is completely eliminated.

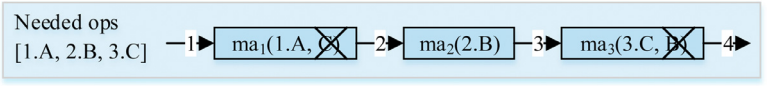

Fig. 8. Breaking loops arising from the multi-functional agent.

```
// Coordinator's roles:
N = the highest UMI;
Defines the variable Temp for temporarily storing agent;

For k = N : 1
  while(true)
    Gets an agent, and saves it to the Temp;  //the Temp now represents the agent
    IF UMI(Temp) == k  //check the UMI of Temp
      IF There is one sub function of Temp redundant with that of another agent
        Forbids the sub function;
        UMI(Temp) = k-1; //adjust the UMI
      END
    END
    IF  Every agent is checked;
      break;  //exit the while loop
    END
  END
END
// MA agents' roles:
//the MA agents only use the sub functions that are not forbidden
```

Fig. 9. Algorithm for eliminating redundant sub functions.

Table 3
Agent classification based on UMI.

No.	Agent classification	Effect
1	UMI 0 agents	All the sub functions are prohibited; therefore, the agents are free, i.e., not dedicated for any operations.
2	UMI 1 agents	Only one sub function is allowed; therefore, the agents operate like single-functional agents.
3	UMI k (k≥2) agents	Each of the agents exclusively undertakes at least two sub functions, becoming a critical agent.

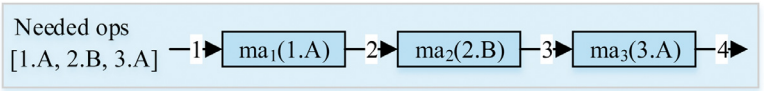

Fig. 10. Breaking loops arising from the multi-occurrence type.

4.3. Strategy 2: Allocation of redundant MA agents to the operations of multi-occurrence types to break loops

After applying Strategy 1, an operation type may be performed by either only one UMI k ($k \geq 1$) multi-functional agent, or a group of single-functional agents. There may be some UMI 0 agents that can undertake the operation type, thus they can be specified as dedicated agents for the operation type. For a multi-occurrence type, we select to equally assign the available single-functional agents and the dedicated UMI 0 agents to different operations of the same operation type (Fig. 10). For example, the multi-occurrence type A shown in Fig. 7 causes a deadlock. However, the allocation of ma_1 to operation 1.A and ma_3 to operation 3.A, breaks the loop and prevents the deadlocks.

Figure 11 shows the algorithm. If the number of agents is enough for a multi-occurrence type, each operation can have a dedicated agent to process it, which can prevent all the deadlocks. Otherwise, one or more agents should undertake at least two operations of the multi-occurrence type. Thus, deadlocks still exist, but the loops are decomposed to multiple small ones, leading to simpler structures.

4.4. Strategy 3: Applying SA agents to break the mutual exclusive use of the critical agents

After applying the strategies 1 and 2, the loops that cannot be broken and the critical agents responsible for the loops can be finally determined. The multiple PA agents will compete for the critical agents that are exclusive resources, and deadlocks might occur. This section describes how the SA agents can be used to break the mutual exclusive use of the critical agents so as to prevent deadlocks.

Strategy 3 applies following rules: 1) each inward link (except one with the lowest number) of the critical agent is assigned to a dedicated slot, and the slot inherits the number of the link; and 2) the PA agents in the higher-numbered slots have higher priority, and the PA agents from the lowest-numbered link have the lowest priority. Note that the SA agents that provide the dedicated slots compete for the corresponding operations with the critical agents, so when the critical agent is busy, the PA agents will enter the slots. Figure 12 summarizes the algorithm:

```
// Coordinator's roles:
Compute sthe number of the operations of the multi-occurrence type
and saves it to the variable OpsCnt;

Computes the number of the agents that support the operation type
and saves it to the variable AgentCnt;

IF  AgentCnt >= OpsCnt
    For each operation, specifies an agent;
ELSE
    Defines Q, R as integer variables;
    Q = OpsCnt / AgentCnt;    //get the quotient
    R = OpsCnt % AgentCnt;  //get the remainder
    For the first R agents, each undertakes Q + 1 operations;
    For the other AgentCnt - R agents, each undertakes Q operations;
END
// MA agents' roles:
//the MA agents only bid for the specified operations
```

Fig. 11. Algorithm for allocating redundant agents to the operations of multi-occurrence type.

```
// Coordinator's roles:
Lists inward links of the critical agent in ascending order: {k[1], k[2], ···, k[n]};
For each inward link except the k[1], specifies a slot:{s[2], ···, s[n]}
// Critical agents' roles:
IF  State == IDLE  //the critical agent is free
  FOR i = n : 2
    IF s[i] == NOT EMPTY
        Accepts the agent from the s[i];
        return; //abort the algorithm, and the following sentences are not executed
    END
  END

  IF There is PA agent from k[1] requesting the operation
    Bids for  the operation;
  END
END
```

Fig. 12. Algorithm for using slots to prevent deadlocks.

Figure 13 illustrates the process: the needed operations are [1.A, 2.B, 3.A, 4.C]. The MA agents ma_1 and ma_2 are responsible for the operation types A and B respectively. The critical agent ma_1 has two inward links. Suppose that an idle slot (no matter where it is and which SA agent it belongs to) is specified to be used by 3.A, as shown in Fig. 13(a). At time t_1, pa_1 and pa_2 cause a deadlock, as shown in Fig. 13(b). However, the deadlock can be prevented with the slot that is specified for 3.A. This situation is illustrated in Fig. 13(c), where the pa_1 enters the slot so that the pa_2 can enter the ma_2 for its 2.B. Then the ma_1 selects the pa_1 other than pa_3 that waits in inward link 1 based on the priority rule. After the pa_1 has its 3.A completed, it exits from the loop to outward link 4, indicating that all the deadlocks related to the ma_1 are prevented.

4.5. Strategy 4: Applying congestion control to prevent deadlocks

If the number of SA agents is not enough, deadlocks cannot be completely prevented. Thus, the strategy 4 applies congestion control to prevent deadlocks for the critical agents that have no SA agents to assist them according to these rules: 1) the inward link with higher operation number is assigned to the higher priority; and 2) the critical agent does not accept PA agents (even it is free) from the lower priority inward link if there are PA agents that tend to come from the higher priority inward links. The algorithm is shown in Fig. 14.

Figure 15 describes how the strategy 4 can prevent deadlocks when the SA agents are not available (i.e., the strategy 3 is not applied). The inward link 3 has the higher priority than the inward link 1. At the time t_1 (Fig. 15(a)), the ma_1 can accept pa_1 from the link 1, because no agents will come from the link 3. Then the pa_1 move forward to ma_2 for its $2.op_2$, as shown in Fig. 15(b). At this time, the ma_1 does not accept a new agent from the inward link 1, because the pa_1 will come from the link 3. Finally, the pa_1 goes back to ma_1 for its $3.op_1$ and then it exits from the loop to outward link 4 (Fig. 15(c)), indicating that all the deadlocks related to the ma_1 are prevented.

5. Simulation

To verify the proposed framework and its associated dynamic negotiation mechanism and deadlock prevention strategies, a simulation program has been developed using Microsoft VS integrated development environment (IDE). Five NF5270M3 servers (Chinese Inspur Corp.) have been interconnected through Ethernet links. A group of virtual machines (VMs) has been created, and each VM is set as an agent. An additional VM has been set as the

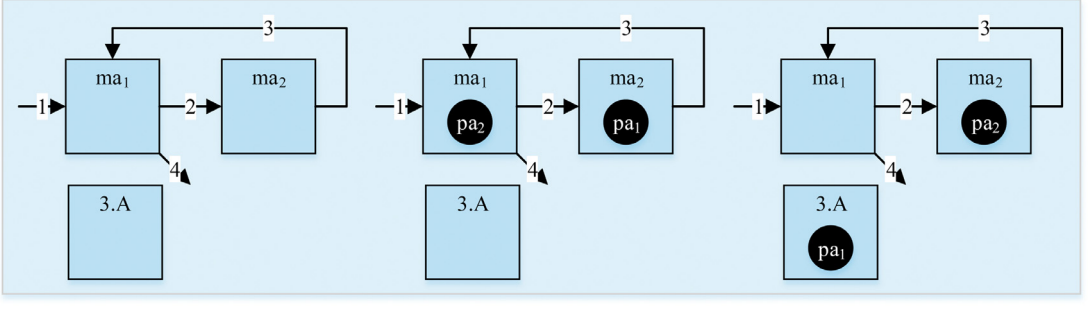

Fig. 13. Snapshots of the loop and its associated slot during the time.

```
// the Coordinator's roles:
Lists inward links of the critical agent in ascending order: {k[1], k[2], ···, k[n]};
// the critical agents' roles:
IF State == IDLE //the critical agent is free
  IF There is a PA agent requesting the operation k[i] //the operation k[i]-1 has been finished
    FOR j = n : i+1
      IF the operation k[j]-1 is being processed
        return; //abort the algorithm, and the following sentences are not executed
      END
    END
    Bids for the operation; //now, the operation k[i] has the highest priority
  END
END
```

Fig. 14. Algorithm for preventing deadlocks using congestion control.

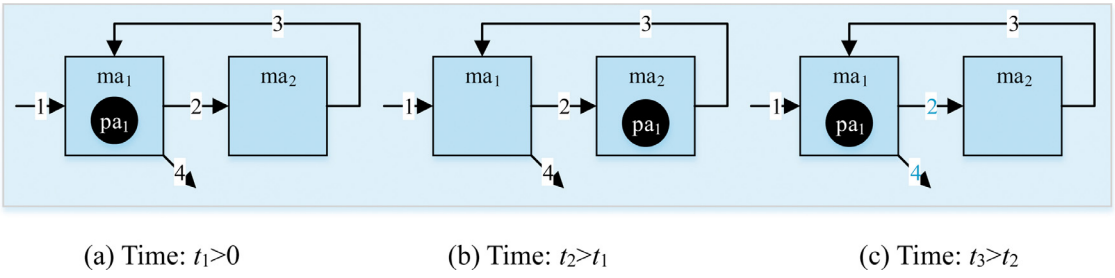

Fig. 15. Snapshots of the loop during the production.

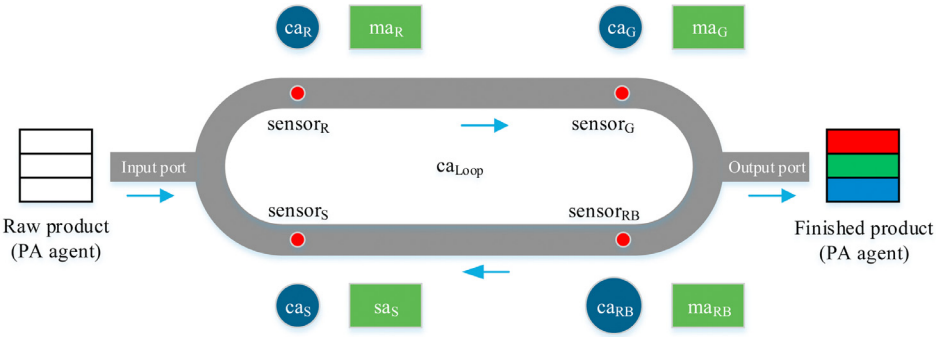

Fig. 16. System layout for simulation of tinting process.

coordinator, and a computer has been linked to the server network for the purpose of supervisory control.

5.1. Simulation layout

A tinting process has been simulated. The rectangular surface of the raw product that represents the PA agent is divided into small rectangles, each of which should be painted using one of the three different colours: red, green, and blue (corresponding to operations R, G, and B respectively). Figure 16 shows the layout of the system. There are three MA agents: ma_R and ma_G are single-functional agents for operations R and G respectively, whereas ma_{RB} is the multi-functional agent for two operations R and B. Moreover, there is an SA agent called sa_S, which has only one slot. Four CA agents (ca_R, ca_G, ca_{RB}, and ca_S respectively) represent robotic arms. These robotic arms interchange PA agents between the circular conveyor belt (implemented as a CA agent called ca_{Loop}) and the MA agents as well as the SA agent.

The ma_R, ma_G, and ma_{RB} are set to paint a rectangle in 20 s. The loading/unloading cycle of robotic arms is set to 5 s. The ca_{Loop} transports a PA agent between two adjacent operation points (denoted as $sensor_R$, $sensor_G$, $sensor_{RB}$, and $sensor_S$ respectively) in 200 s. The input port lies between $sensor_R$ and $sensor_S$, whereas the output port is between $sensor_G$ and $sensor_{RB}$.

5.2. Simulation results and discussion

The processing of two different types of products is simulated. One product type shows how the multi-functional agent causes deadlocks, while the second one illustrates how the multi-occurrence type causes deadlocks. A batch is composed of twenty products of the same type. Table 4 summarizes the results. The notation "√" in

Table 4
Simulation plan and results.

Product type	Batch	Applied strategies				Agent sequence	Result	Efficiency (s)
		1	2	3	4			
[1.R, 2.G, 3.B]	1	×	×	×	×	[ma_R, ma_G] [ma_{RB}]	×	––
	2	√	×	×	×	[ma_R, ma_G, ma_{RB}]	√	5660
[1.R, 2.G, 3.R]	3	×	×	×	×	Same as batch 1	×	––
	4	√	×	×	×	Same as batch 1	×	––
	5	√	√	×	×	Same as batch 2	√	5660
	6*	√	√	√	×	[ma_R, ma_G, sa_S, ma_R]	√	9490
	7*,$	√	√	×	√	[ma_R, ma_G, ma_R]	√	20005

* suppose that the ma_{RB} malfunctions.
$ suppose that the sa_S malfunctions.

"Applied strategies" column means that the strategy is applied, whereas the notation "×" means the strategy is not applied. The notation "√" in "Result" column means that the batch can be successfully processed, whereas the notation "×" means the batch cannot be successfully processed. The efficiency is evaluated as the total processing time of the batch, expressed in seconds.

The PA agents should be processed by a sequence of the MA and SA agents as shown in the "Agent sequence" column of Table 4. These sequences are determined dynamically through negotiation.

For the product type [1.R, 2.G, 3.B], the multi-functional agent ma_{RB} will cause a loop, so the deadlock will occur without applying any deadlock prevention strategies (as shown in batch 1). After applying strategy 1, the ma_{RB} is specified to the B operation thus the loop is broken preventing the deadlock (as shown in batch 2).

For the product type [1.R, 2.G, 3.R], the multi-occurrence operation type R will cause a loop, thus the deadlock will occur without applying any deadlock prevention strategies (as shown in batch 3). The strategy 1 solely cannot break the loop either (as shown in batch 4), but it contributes to identify the ma_{RB} as a UMI 0 agent. In this case, the strategy 2 can be applied to break the loop where the ma_R and ma_{RB} are responsible for the operations 1.R and 3.R respectively (as shown in batch 5). However, if the ma_{RB} malfunctions (strategies 1 and 2 are not effective in this case), the strategies 3 or 4 should be applied instead to prevent the deadlocks, as shown in batch 6 and batch 7 respectively. The sole slot of sa_S is specified to the 3.R in the strategy 3, and the ma_R is responsible for the congestion control in strategy 4.

The efficiency of the batches 2, 5, 6, and 7 indicates that the strategy 1 and strategy 2 are more efficient than the strategy 3, while the strategy 4 is the least efficient. However, the strategies 1 and 2 need more resources than strategy 3, and strategy 4 is the most efficient in terms of resources consumption. Therefore, there is a positive correlation between resources and efficiency and the four strategies should be used together to achieve the balance between these two factors.

The simulation results indicate that the implementation of smart factory can prevent deadlocks caused by multi-functional MA agents or multi-occurrence operation types. Moreover, as the negotiation mechanism enables the operations of the smart factory to be reconfigured dynamically, the smart factory can process various product types provided that the operations belong to the available operation types R, G, and B. The system is also robust to disturbance, e.g., when the ma_{RB} and/or sa_S agent malfunction. Dynamic negotiation and the deadlock prevention strategies let the system achieve flexibility, agility, and fault-tolerance ability.

6. Conclusions

The emerging information technologies such as IoT, big data, and cloud computing, together with artificial intelligence technologies, help to implement the smart factory of Industry 4.0. The smart machines, conveyers, and products communicate and negotiate with each other to reconfigure themselves for flexible production of multiple types of products. The industrial network collects massive data from smart objects and transfers them to the cloud. This enables system-wide feedback and coordination based on big data analytics to optimize system performance. The above self-organized reconfiguration and big-data-based feedback and coordination define the framework and operational mechanism of the smart factory.

In this paper, distributed self-decision making and intelligent negotiation mechanisms are carefully designed to implement self-organized manufacturing system. Moreover, the deadlock prevention strategies are implemented based on the integrated effort of autonomous agents and the central coordinator. Based on the model presented in this paper, we are going to build up a smart factory prototype, and develop algorithms for further optimizing system performance.

Acknowledgments

This work was supported in part by the National Key Technology R&D Program of China (Grant no. 2015BAF20B01), the Fundamental Research Funds for the Central Universities (Grant nos. 2015ZZ079, 2014ZM0014, 2014ZM0017, 2013KJ034, and 2100219043), the Strategic Emerging Industry's Core Technology R&D Program of Guangdong Province (Grant nos. 2012A090100012 and 2012A010702004), the National Science Foundation of China (Grant nos. 61472283, 61262013, and 61103185), the Science and Technology Planning Project of Guangzhou City (Grant no. 201508030007), and the Fok Ying-Tong Education Foundation, China (Grant no. 142006).

References

[1] E. Alkaya, M. Bogurcu, F. Ulutas, G.N. Demirer, Adaptation to climate change in industry: improving resource efficiency through sustainable production applications, Water Environment Research 87 (1) (2015) 14–25.

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

[3] W. Shen, Q. Hao, H.J. Yoon, D.H. Norrie, Applications of agent-based systems in intelligent manufacturing: An updated review, Advanced Engineering Informatics 20 (4) (2006) 415–431.

[4] F. Tao, Y. Zuo, L.D. Xu, L. Zhang, IoT based intelligent perception and access of manufacturing resource towards cloud manufacturing, IEEE Transactions on Industrial Informatics 10 (2) (2014) 1547–1557.

[5] Q. Jing, A. Vasilakos, J. Wan, J. Lu, D. Qiu, Security of the internet of things: perspectives and challenges, Wireless Networks 20 (8) (2014) 2481–2501.

[6] F. Chen, P. Deng, J. Wan, D. Zhang, A. Vasilakos, X. Rong, Data mining for the internet of things: literature review and challenges, International Journal of Distributed Sensor Networks (2015) Article ID 431047, 14 pages, 2015.

[7] M. Qiu, X. Chun, Z. Shao, Q. Zhuge, M. Liu, E. Sha, Efficient algorithm of energy minimization for heterogeneous wireless sensor network, Proceedings of the Embedded and Ubiquitous Computing, Springer Berlin Heidelberg, 2006, pp. 25–34.

[8] M. Qiu, E. Sha, Energy-aware online algorithm to satisfy sampling rates with guaranteed probability for sensor applications, Proceedings of the High Performance Computing and Communications, Springer Berlin Heidelberg, 2007, pp. 156–167.

[9] M. Chen, S. Mao, Y. Liu, Big data: a survey, Mobile Networks and Applications 19 (2) (2014) 171–209.

[10] X. Xu, From cloud computing to cloud manufacturing, Robotics and computer-integrated manufacturing 28 (1) (2012) 75–86.

[11] Q. Liu, J. Wan, K. Zhou, Cloud manufacturing service system for industrial-cluster-oriented application, Journal of Internet Technology 15 (4) (2014) 373–380.

[12] J. Wan, D. Zhang, Y. Sun, K. Lin, C. Zou, H. Cai, VCMIA: A novel architecture for integrating vehicular cyber-physical systems and mobile cloud computing, Mobile Networks and Applications 19 (2) (2014) 153–160.

[13] J. Wan, D. Li, H. Yan, P. Zhang, Fuzzy feedback scheduling algorithm based on central processing unit utilization for a software-based computer numerical control system, Proceedings of the the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture 224 (7) (2010) 1133–1143.

[14] F. Soliman, M.A. Youssef, Internet-based e-commerce and its impact on manufacturing and business operations, Industrial Management & Data Systems 103 (8) (2003) 546–552.

[15] "Recommendations for implementing the strategic initiative INDUSTRIE 4.0", April 2013. Available at: http://www.acatech.de/fileadmin/user_upload/Baumstruktur_nach_Website/Acatech/root/de/Material_fuer_Sonderseiten/Industrie_4.0/Final_report__Industrie_4.0_accessible.pdf .

[16] The Industrial Internet Consortium: A Global Nonprofit Partnership Of Industry, Government And Academia, March 2014. Available at: http://www.iiconsortium.org/about-us.htm .

[17] Premier of the State Council of China, K.Q. Li, "Report on the work of the government," delivered at the third session of the 12th National People's Congress, March 5, 2015.

[18] M. Riedl, H. Zipper, M. Meier, C. Diedrich, Cyber-physical systems alter automation architectures, Annual Reviews in Control 38 (1) (2014) 123–133.

[19] J. Wan, H. Yan, Q. Liu, K. Zhou, R. Lu, D. Li, Enabling cyber-physical systems with machine-to-machine technologies, International Journal of Ad Hoc and Ubiquitous Computing 13 (3/4) (2013) 187–196.

[20] J. Wan, D. Zhang, S. Zhao, L. Yang, J. Lloret, Context-aware vehicular cyber-physical systems with cloud support: architecture, challenges, and solutions, IEEE Communications Magazine 52 (8) (2014) 106–113.

[21] E.M. Frazzon, J. Hartmann, T. Makuschewitz, B. Scholz-Reiter, Towards socio-cyber-physical systems in production networks, Procedia CIRP (7) (2013) 49–54.

[22] O.O. Balogun, K. Popplewell, Towards the integration of flexible manufacturing system scheduling, International Journal of Production Research 37 (15) (1999) 3399–3428.

[23] W. Shen, Q. Hao, H.J. Yoon, D.H. Norrie, Applications of agent-based systems in intelligent manufacturing: An updated review, Advanced Engineering Informatics 20 (4) (2006) 415–431.

[24] S. Wang, J. Wan, D. Li, and C. Zhang, "Implementing Smart Factory of Industrie 4.0: An Outlook," International Journal of Distributed Sensor Networks, in press.

[25] R.G. Smith, The contract net protocol: high-level communication and control in a distributed problem solver, IEEE Transactions on Computers C-29 (12) (1980) 1104–1113.

[26] Z.W. Li, M.C. Zhou, N.Q. Wu, A survey and comparison of petri net-based deadlock prevention policies for flexible manufacturing systems, IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 38 (2) (2008) 173–188.