Scientific Research Publishing

Journal of Software Engineering and Applications, 2017, 10, 257-272
http://www.scirp.org/journal/jsea
ISSN Online: 1945-3124
ISSN Print: 1945-3116

Improve the Performance of a Complex FMS with a Hybrid Machine Learning Algorithm

Hang Li

Institute of Automation and Information System, Technische Universität München, Garching near Munich, Germany
Email: hang.li@tum.de

How to cite this paper: Li, H. (2017) Improve the Performance of a Complex FMS with a Hybrid Machine Learning Algorithm. *Journal of Software Engineering and Applications*, **10**, 257-272.
https://doi.org/10.4236/jsea.2017.103015

Received: January 26, 2017
Accepted: March 10, 2017
Published: March 13, 2017

Abstract

Modern manufacturing systems are expected to undertake multiple tasks, flexible for extensive customization, and that trends make production systems become more and more complicated. The advantage of a complex production system is a capability to fulfill more intensive goods production and to adapt to various parameters in different conditions. The disadvantage of a complex system, on the other hand, with the pace of the increase of complexity, lies in the control difficulties rising dramatically. Moreover, classical methods are reluctant to control a complex system, and searching for the appropriate control policy tends to become more complicated. Thanks to the development of machine learning technology, this problem is provided with more possibilities for the solutions. In this paper, a hybrid machine learning algorithm, integrating genetic algorithm and reinforcement learning algorithm, is proposed to cope with the accuracy of a control policy and system optimization issue in the simulation of a complex manufacturing system. The objective of this paper is to cut down the makespan and the due date in the manufacturing system. Three use cases, based on the different recipe of the product, are employed to validate the algorithm, and the results prove the applicability of the hybrid algorithm. Besides that, some additionally obtained results are beneficial to find out a solution for the complex system optimization and manufacturing system structure transformation.

Keywords

Complex System, Flexible Manufacturing System (FMS), Machine Learning, System Optimization

1. Introduction

Over the past few years, industrial manufacturing is confronted with extensive changes. From local economy towards a globalized and fully competitive econ-

omy, markets require highly qualified and customized products at lower costs and with shorter life cycles [1]. To dispose of these challenges, the performance of their production system has to be improved by the manufacturing enterprises. Since last decades, new concepts for manufacturing system have been developed. An agile manufacturing system brings to production much greater concurrency and integration of activities [2]. Sustainable manufacturing requires a holistic view spanning, not only a product, and production processes involved in its manufacturing, but also the whole supply chain, including the manufacturing systems across multiple product life cycles [3]. Furthermore, a coupled cyber-physical system scheme of predictive manufacturing system is developed to integrate, manage and analyse machinery or process data to operate more efficiently during life cycle by Lee *et al.* [4].

Among all these concepts, production process tends to be characterized by modularity, decentralization, autonomy, scalability, reusability, adaptability and reconfigurability. Moreover, a critical issue in production is the system control approach. For a complex system, traditional control methods are reluctant to the adaptation of more input variables, state parameters, and more flexible production requirements.

One regular practice is a job shop scheduling or a job shop problem (JSP). The objective of that is to minimize the makespan with the given n jobs and m identical machines in a factory. It is recognized as an NP-hard problem in the mathematical domain [5]. With classical control policy and present programmable logic controller (PLC), it is hardly possible when the topological structure of a manufacturing system is quite complicated. Due to the rapid development of artificial intelligence (AI), significant results have been obtained with advanced algorithms such as simulated annealing [6], evolutionary algorithm [7], and hybrid AI algorithm [8]. These kinds of probability-based or evolution-based local search algorithms make it possible to obtain the optimal solution, although the solution space is quite large. However, one disadvantage of the heuristic algorithm and the evolutionary algorithm is that too many parameters influence the convergent speed of a training model based on manufacturing process effects. If to be more precise, they are not convergent at all, e.g. the convergent rate of a genetic algorithm (GA) subjects to the encoding of chromosomes, selection of initial population, etc. Jimenez *et al.* employed GA on the Pollux architecture in a job shop problem to improve total job makespan, however, the complementary statistical studies needed to be generalized [9]. Moreover, owing to the specific system structure, there is no suboptimal solution interval in the global solution space. Under some circumstances, the GA cannot dispose control policies for a complex system. However, integration with another algorithm may improve the convergent situation. That will be discussed in Section 3 and Section 4 of this paper.

An inferior encoding method generates an enormous amount of invalid solutions and enlarges searching space, so a reasonable method for encoding is the precondition for GA. Binary encoding and Gray encoding are suitable for a se-

ries of real parameters e.g. five DeJong test functions [10]. With weighted coding, the knapsack problem has been solved successfully [11]. Ordinal representation [12] meets requirements of the Traveling Salesman Problem (TSP) that a salesperson must pass through every city once and only once. Matrix encoding can cope with the JSP scheduling and present better results [13]. Nevertheless, for a particular practical problem, researchers cannot always find an encoding plan easily, e.g. Grefenstette *et al.* came up with an ordinal representation [14], compared with the GA that was described in the 1960s, and the TSP that was formulated in 1930s. Initial parents of the population are selected after encoding for a practical problem. During selection of offspring generations, elitist selection [15] and gradient acceleration [16] are introduced to speed up the crossover process. If the gradient of solution space or some elite chromosomes is found, then evolution direction is known. Correspondingly, initial efforts turn to how to allocate the gradient. In a specific engineering practice, the gradient of solution space is usually obtained from accumulated experience after years of a manufacturing system analysis.

Another part of the hybrid algorithm is reinforcement learning. Reinforcement learning (RL) is another sound algorithm based on studies of a system's structure. RL algorithms are a series of learning policies that make programs improve their performance by receiving rewards or punishments from the environment [17]. It is learning of a mapping from situations by actions to maximize a scalar reward or reinforcement signal [18]. After establishing initial state-ac- tion pairs and system dynamics, solution space is defined. Then the system calculates Bellman optimization function based on a trial and error process with reward or penalty. Since a reward function affects the convergence speed or even makes values of Bellman function diverge, the reward function searching procedure must be prior to a system design process or decided in advance according to some previous experience. The RL algorithms are rarely implemented in engineering practice mainly because of the difficulty in determining a sound reward function.

In this paper, a hybrid machine learning algorithm is represented to improve a flexible manufacturing system. An approach based on both system analyses and the results from messy GA is proposed. It presents a way to making a decision on a better reward function for RL algorithm. All this work is executed in the simulation environment of CoDeSys under IEC 61131-3. Based on obtained experience, an idea to optimize flexible system structure is discussed.

The rest of the paper is organized as follows. Section 2 introduces the flexible manufacturing system and formulates the problem. The control policy is discussed in Section 3. Simulated results are present and discussed in Section 4. Then some additional beneficial results are presented in Section 5. Section 6 concludes the paper and proposes avenues for future work.

2. Problem Formulation

Nowadays, a flexible manufacturing system becomes more and more completed and flexible for massive customization. It is designed with more branches be-

tween workstations, which means it can provide more options for products to be transport and flexibility for manufacturing different kinds of goods. A demonstrator of such a system in the laboratory of the chair of Automation and Information Systems, Technical University of Munich, is shown in Figure 1.

Some fundamental properties of the manufacturing system can be concluded as follows:

- There are two filling machines mounted on a workstation respectively at the left side of the system, each with two kinds of pellets, which means it can provide different recipes. So it is a flexible manufacturing system, and it can be used to produce various types of products.
- There are ten conveyor switches which connect workstations and inventory, and each of them leads to the different directions in the system. That means that there are several routes for the production process for one kind of a product.
- The inventory for raw product and inventory for final product are located next to each other and share one robot.

There are five materials in the experiment, namely red pellets, green pellets, blue pellets, black pellets, and water. Among them, the racks of red pellets and green pellets are mounted on workstation 1; the racks of blue pellets and black pellets are installed on workstation 2. Referring to water, it is a dependent variable. Thus we just consider the pellets. It is easy to find out that there are at most $2^4 - 1 = 15$ products. Here, we use $\{R, G, B, S\}$ to represent red, green, blue and blackpellets, w to represent workstations, z to count the number of either productor workstation, r_p to represent the recipes of the products, and $t_{w,pi}$ to present the processing time of one product by $i-th$ workstation.

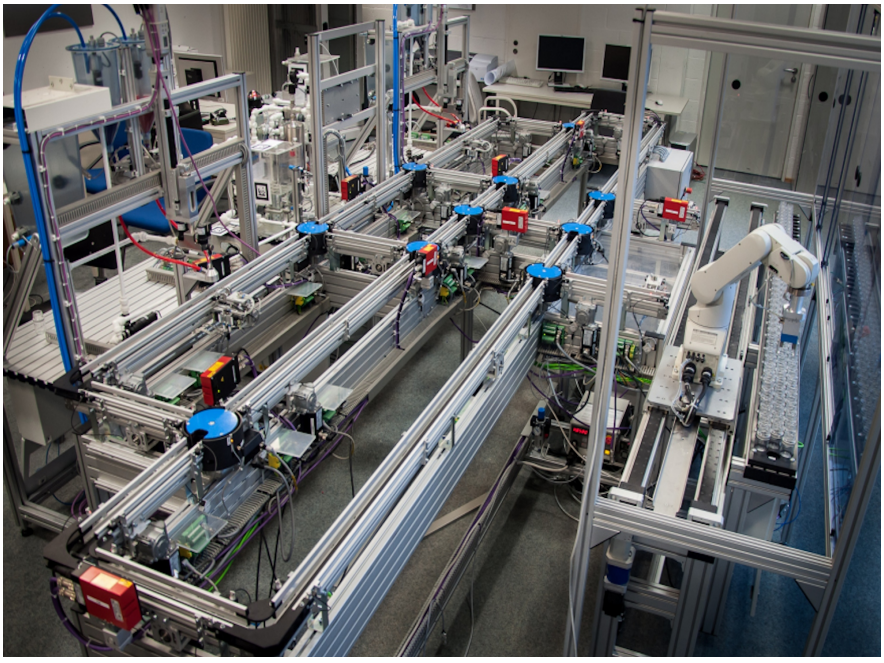

Figure 1. A demonstrator for complex manufacturing system in the chair of Automation and Information Systems, Technical University of Munich.

According to the position of raw materials on workstations, we will check the recipe of a product and define sub use cases.

2.1. Use Case 1

If the recipes of products are following:

$$r_p = \{R, G, R \cup G\} \tag{1}$$

That means a raw product must go by the workstation 1, but not the workstation 2. That is defined as a use case 1. Based on the above recipes, the use case 1 consists of 3 sub-scenarios.

2.2. Use Case 2

If the recipes of products are following:

$$r_p = \{B, S, B \cup S\} \tag{2}$$

That means a raw product must go by the workstation 2, but not the workstation 1. That is defined as a use case 2. Based on the above recipes, the use case 2 consists of 3 sub-scenarios.

2.3. Use Case 3

If the recipes of products are following:

$$r_p = \{\{R \cup G | w_1\} \cup \{B \cup S | w_2\}\} \tag{3}$$

That means a raw product must go by the workstation 2, but not the workstation 1. That is defined as a use case 3. According to the above recipes, the use case 3 consists of 9 sub-scenarios.

Here, we need some further discussion on the mentioned use cases.

1) $\forall w_i \in W$, $(r_{p1} \cap r_{p2}) \cup (t_{w,p_1} \cap t_{w,p2}) = \varnothing$, e.g. $r_{p1} = \{R \cup B\}$, $r_{p2} = \{S\}$, that means a filling process of one product does not interfere the other. It is defined as the use cases 1 and 2.

2) $\exists w_i \in W$, $(r_{p1} \cap r_{p2}) \cup (t_{w,p_1} \cap t_{w,p2}) \neq \varnothing$, e.g. $r_{p1} = \{R \cup S\}$, $r_{p2} = \{G\}$, that means a filling process of one product will interfere the other one. It is defined as the use case 3.

3) If there are more than two products, that means $z > |w_i|$, with $|w_i|$ denoting the number of available workstations, a filling process will interfere others. It is also defined as the use case 3.

Besides, we have to consider the following two scenes:

1) When a raw product is a feed by the entrance of the system, it has several routes to a workstation, and after the filling process, it has random ways to the exit. Different route plan consumes different makespan, and it may interrupt the following product and generates "traffic jam."

2) When a final product appears by the exit of the manufacturing system, the robot has two options, to feed a raw product to the system first or to pelletize the completed product. Different choices result in different tardiness time and completion time.

These two scenarios are referred to the behavior of the feeding/palletizing robot, but one selection of the robot has a global influence on a manufacturing system. So they are viewed as a part of the three use cases defined above. Due to the random characteristics of the process, e.g. the various completion time by different routes, the whole manufacturing process is a stochastic process. Thus, the goal of this paper is to find out how to plan a reasonable route for the global process and to determine the potential relationship between feeding interval and completion time with the application of a hybrid machine learning algorithm.

3. Methodology of Control Policy for a Manufacturing System

After illustrating and formulating the target system, the next step is to find a reasonable control method to implement. Figure 1 shows that the flexible manufacturing system can be used for different products and can provide more than one option for the product, which will make the control process more difficult or even conflict. However, on the other hand, its property has also transformed the working process from a deterministic process to a stochastic process, so the machine learning algorithms can be employed to cope with it.

3.1. System Structure Analyses

Before the introduction of the algorithm implementation, we need to have an overview of a representative complex manufacturing system. Essential elements of the system consist of the robot, workstations, and conveyor belts (CB) and conveyor switches (CS). Conveyor belts transport products to the targeted workstation and conveyor switches change routes for the process and make it possible for one product to bypass an unnecessary workstation. Under the normal circumstance, one conveyor switch cannot work independently but must cooperate with conveyor belts. It connects three belts in the manufacturing system so that the system can decide not only which product will be delivered first, but also the coordination of a process when several belts are taking one product to the conveyor switch simultaneously. So the fundamental research unit consisting of one conveyor switch and three conveyor belts is defined as a trident node. In other words, transporting tasks in the flexible manufacturing system are undertaken by Trident nodes. The decomposition of the fabrication system to Trident nodes is shown in Figure 2. Furthermore, the details of the system analyses and dynamics can be seen in the previous work of the author in [19].

One important target of the system is to decide a better transport order and to coordinate behaviors of conveyor belts. So the key component in the node is the conveyor switch. Since one node consists of four elements, all the conveyor switches are numbered by 4n. On the global level, a product will go from the feeding part to the palletizing part. According to the system structure in Figure 2, for the n-th node, the product comes from the right side in most of the cases. So a conveyor belt on the right side in the node will be numbered by $4n + 1$. Further, vertical conveyor belts are numbered by $4n + 2$, and a conveyor belt on

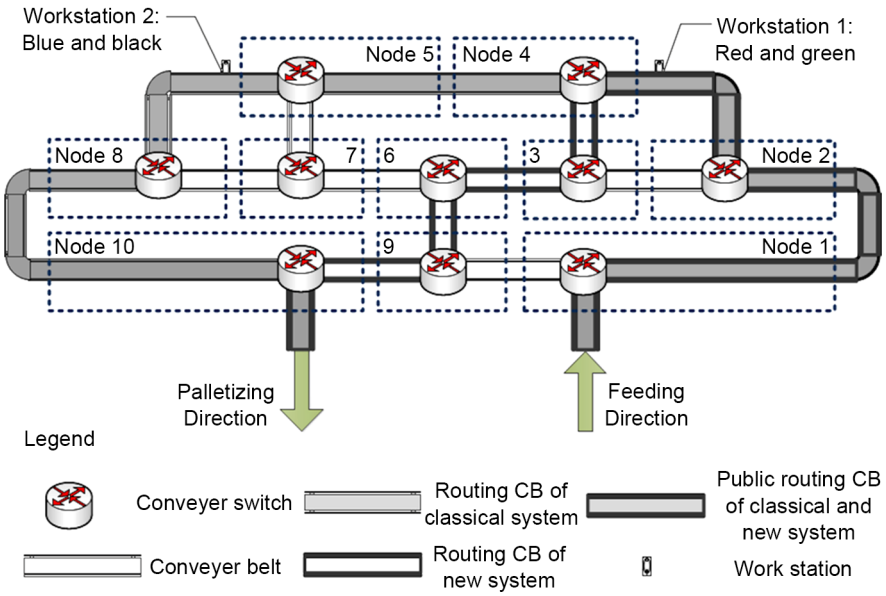

Figure 2. The node definition of the manufacturing system.

the left end of one conveyor switch will be counted by 4n + 3. Besides that, a public conveyor belt in adjacent nodes has two identifiers.

The control system tries to find the time-efficient option for one product; then it attempts to find the time-efficient option for the whole manufacturing process. On the other hand, algorithm recognizes the influence of each option or each node simultaneously. The flowchart of the hybrid algorithm is displayed in **Figure 3**.

3.2. Implementation of GA

According to the above analyses, a messy GA constitutes the basis of this hybrid algorithm. A standard procedure for the implementation process of GA includes encoding, operation, and selection.

3.2.1. Encoding

In this paper, the system is encoded by the sequence of the nodes' number with genetic algorithms. A raw product may pass the different amount of nodes, so the gene code is a messy digital number. It was first mentioned by D. Goldberg [20], but the messy encoding method in this paper differs from it.

A "tour" is defined as a product traveling from the feeding entrance to the palletizing exit. A "legal tour" means a pathway that passes by the corresponding filling machine according to the recipes. On the contrary, an "illegal tour" doesn't pass by the corresponding filling machine. For example, 1243690 is a legal tour for the recipe "red pellets", and 1236780 is an illegal tour for the recipe "black pellets". In the coded number, "0" represents node 10 to avoid conflict with node 1.

3.2.2. Gene Length

Since an encoding number implies the passing order of nodes, the gene length is

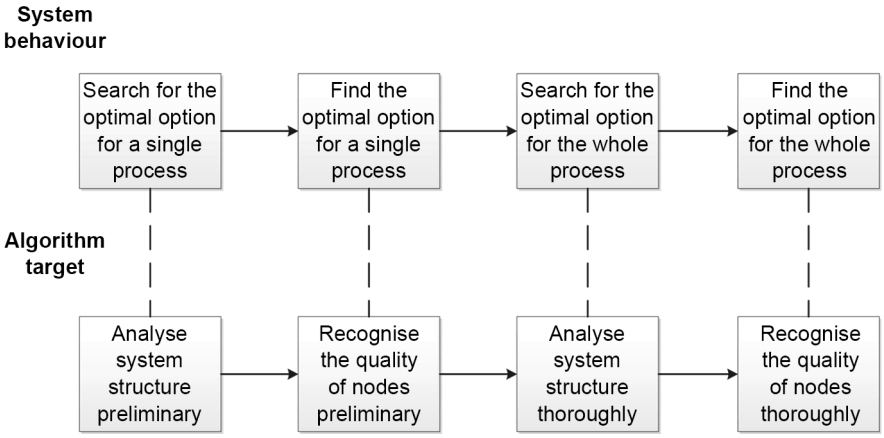

Figure 3. The flowchart of the hybrid algorithm, system behaviour and algorithm target in parallel.

naturally more than 6. That means a product must pass through at least six nodes from the feeding point to the palletizing point via some filling station. Intuitively, the gene length can be a relatively huge number because a product may go around some cycles e.g. "3243", "345783". If a solution is a very long string, it will consume more time. Thus it can be considered as not an optimal solution. On the other hand, because of constraints of a crossover operator and mutation operator, the gene length cannot be set too short either. In this paper, the gene length shall be fixed to no more than 16.

3.2.3. Crossover Operator

During the crossover operation, the same node(s) except node 10, node 1, node involved in a filling process and the adjacent node to 10 and 1 in common between two parents is searched at first. For instance, the following two parents are legal tours for the use case 1.

Parent 1: [1] [2] [4] [3] [6] [7] [8] [0]

Parent 2: [1] [9] [6] [3] [2] [4] [5] [8] [0]

This means two parental solutions pass through common node 1, 2, 3, 4, 6, 8, 10. Node 3 is involved in a filling process for the use case 1 and node 8 is the adjacent node to 1 in common. In that way, node 2, 4, 6 are potential crossover points. The system will choose one of them randomly and then will build up an offspring. If one child represents an illegal tour, the system will select another rest crossover point to generate new offspring. Suppose node 6, which is underlined in parents, is a crossover point, in this case, we get offspring like following,

Offspring 1': [1] [9] [6] [7] [8] [0]

Offspring 2: [1] [2] [4] [3] [6] [3] [2] [4] [5] [8] [0]

The route of offspring one does not include the filling station. Using node□; 4, as a crossover point, the system keeps offspring2 and then generates a new random child,

Offspring 1: [1] [2] [4] [5] [8] [0]

The crossover rate is set as 1 in this paper.

3.2.4. Mutation Operator

The mutation is necessary for helping this algorithm to jump out of a premature searching space. Inspired by cycles in the system mentioned in 3.2.2, we employ them to execute a mutation operator. They increase the gene length, but not always increase time consumption. So a cycle is applied here as modifications to help the system to get rid of a pseudo-optimal solution space. One of the node 3, 4, 5, 7 will be selected to add a cycle. Suppose a parent as following:

Parent: [1][2][4][5][8][0]

One possible mutation might be,

Offspring: [1][2][4][3][2][4][5][8][0]

Node 4 which is underlined in the parent's chromosome is randomly chosen as a mutation point in the instance. The mutation rate is set based on the parent. If chromosomes of offspring are identical to two parents, then the mutation rate is set as 1. Else, if potential chromosomes of offspring are different from parents, the mutation rate is set as 0.05 at the global level.

3.2.5. Selection of the Initial Parents

In the mutation process, an extra circle is added. Similarly, a circle in a parent can be also subtracted. However, that will make an intergenerational transmission of a mess. To avoid that kind of confusion, non-repeat chromosome routes are provided with higher priority. However, it does not mean that non-repeat chromosome routes are elite routes. This is another necessity of the mutation process. 15 initial parents in the whole population are chosen to generate and iterate the later population.

- If the non-repeat chromosome population in a use case is more than 15, then choose 15 of them randomly.
- If the non-repeat chromosome population in a use case is less than 10, the system will accept all of them first and then substitute the vacancy with repeating chromosome randomly.

Based on the analysis in the previous section, a repeating chromosome offspring can be generated by the non-repeat parents. It is essential because the optimal solution may exit in repeat chromosome routes.

4. Simulation Results

Because control units on real plants are PLC-mounted, the simulation is executed on CoDeSysV3.5 SP5 Patch 3. CoDesys is a developing environment based on the IEC-61131 standard. Physical parameters are following,

- Speed of conveyor belt - v_b = 300 mm/s;
- The robot transporting time - t_{rt} = 2 s;
- The moving speed of robot - v_{rm} = 300 mm/s;
- The filling time at each workstation - t_f = 2 s.

Another fundamental property should be pointed that one product can pass a workstation several times, but it can pick up pellets by one workstation only once in the simulation. The behaviors of the system in this paper are simulated

with the proposed hybrid algorithm. Owing to the characteristic of a stochastic process, one node in the system can decide the transporting sequence and direction, and the robot can decide to feed a raw product first or to palletize a finished product. The goal of the simulation is to recreate the decision and learning process of the system to find out whether the hybrid algorithm fits the circumstance. It is evident that the production effectivity is decided by robot transporting time, moving time on the slide way of the robot, and time-consuming within the manufacturing system. On the other hand, the makespan in the production system is denoted by one selected route in the stochastic process. Because of its own characteristics of the system, there is no intervention process in the use case 1 and the use case 2. So targets of the use case 1 and the use case 2 are to find the optimal routes. The simulation results are displayed in Figure 4 and Figure 5 respectively.

From Figure 4 and Figure 5, it is evident that optimal solutions are non-repeat chromosome sequence. The optimal solution for the use case 3 is displayed in Figure 6.

4.1. Lessons Learned

Figure 6 shows that the optimal solution found by the GA for the use case 3 is a repeat chromosome. This means that a product may hinder another product. The optimal solution for the entire process is not a simple superposition of the optimal solution, but a combination of a set of simpler solutions. Thus, this genetic algorithm cannot dispose of the use case 3 with present parameters and operators. Here, we use one reinforcement learning algorithm to cope with this use case. From the implementation of GA in the use case 1 and the use case 2, it can be found that the weight of each node is different. For example, in the use case 1, there are three kinds of nodes. A product with the corresponding recipe must pass through node 1, 2, 4, 10, which means they are essential nodes and if

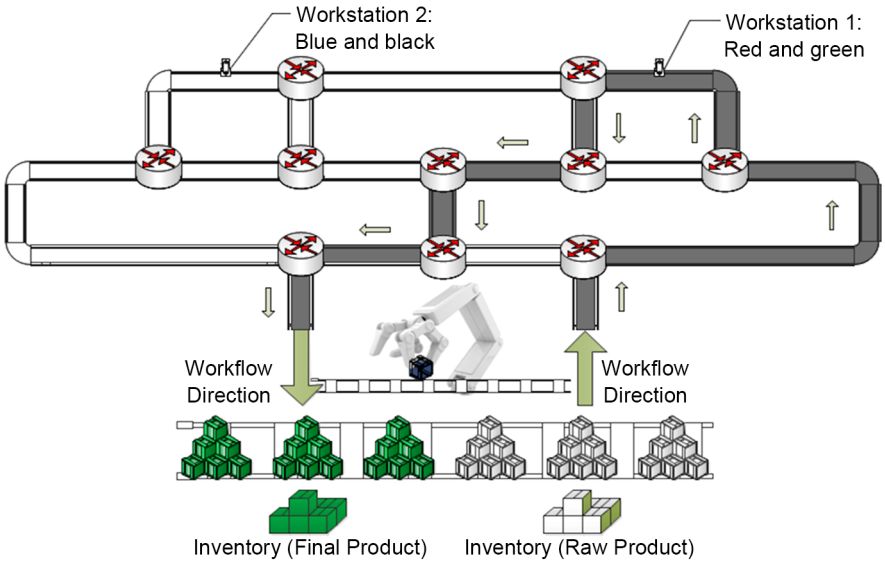

Figure 4. The optimal solution for the use case 1: (G2-2): 1243690.

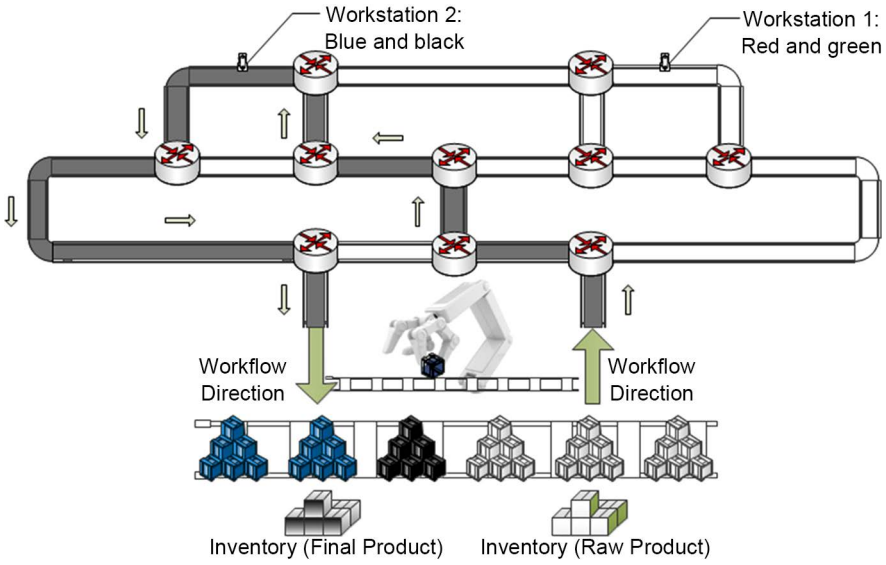

Figure 5. The optimal solution for the use case 2 (G2-4): 1967580.

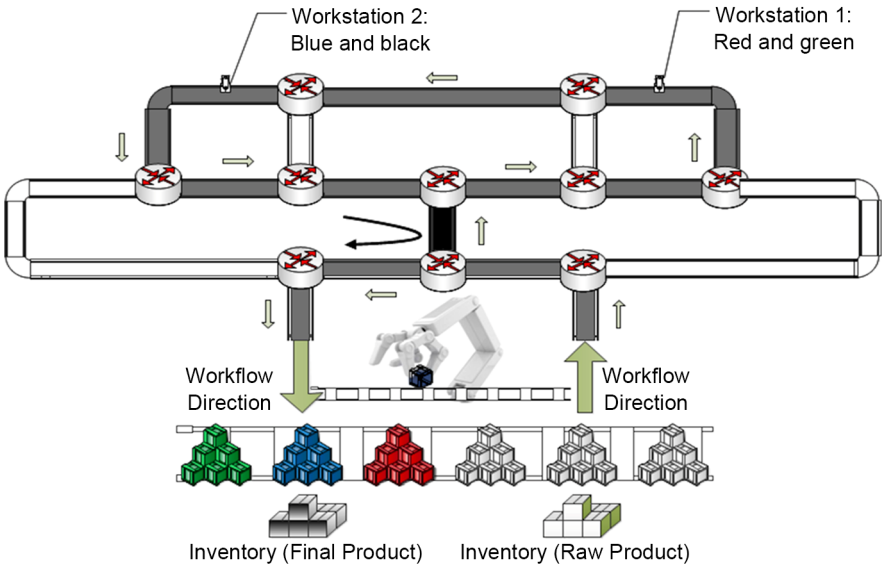

Figure 6. The optimal solution for the use case 3.

those four nodes appear in one solution, it does not affect the performance of a solution. On the other hand, rest nodes are non-essential nodes. Nonetheless, they will affect the performance. The influence of these nodes may be beneficial, or harmful. In fact, a raw product needs only red pellets or green pellets in use case 2, which means it is time-saving to avoid node 5, 7, 8 in the route plan. Thus, node 5, 7, 8 are defined as the negative nodes or negative chromosomes. Finally, we cannot judge rest nodes 2, 3, 6, 9 on the surface. So they are defined as neutral nodes or chromosomes.

There are also some other facts we need to consider. For example, there are several routes for a product to go through node 8 to node 10. One is from node 8 to node 10; another might be from node eight via node 7, 6, 9 to node 10. That means adjacent nodes are not independent.

Based on experience gained before, there ward function can be set as follows:

$$r = 2 * \sum_0^n Z(\mathrm{Pn}) + 0 * \sum_0^n Z(\mathrm{Nn}) - \sum_0^n Z(\mathrm{Hn}) - \sum_0^n Z(n_i n_j) \quad (4)$$

Here, Pn represents the potential positive nodes; Hn represents the harmful nodes; Nn represents the neutral nodes, and $Z(\ )$ counts the passing number of nodes.

There are 63 raw products waiting in line in total based on the physical structure displayed on the right side in Figure 1. The process from the first raw product is moving to the entrance conveyor belt to the 63rd product getting off the exit conveyor belt, is defined as one episode.

4.2. Implementation of the Reinforcement Learning Algorithm

After the reward function is determined, the reinforcement learning algorithms need to be introduced briefly. A typical decision starts at time t_0, with the initial state given by s_0. At any time, possible actions are based on the current state, and it is depicted as $a_t \in \Gamma(s_t)$ [21]. The value of the state-action pair (s,a) under the policy π, denoted by $V^\pi(s,a)$, represents the expected return when starting in state s, taking action a and following policy π thereafter:

$$V^\pi(s,a) = F^\pi\{E|s,a\} \quad (5)$$

where $F^\pi\{\cdot\}$ denotes time consumption under the stochastic dynamics f, given that the controller applies policy π. The optimal action value function V^* is defined as the maximum Bellman equation among all potential policies:

$$V^*(s,a) = \max V^\pi(s,a) \quad (6)$$

Once V^* is obtained, an optimal policy (*i.e.* one that minimizes time consumption) can be decided by an optimization over the action argument:

$$\pi^*(s) = \arg\max V^*(s,a). \quad (7)$$

The Q-learning algorithm estimates Q* from the interaction between actuators and environment, depending on both the previous state and the selected action. Thus Q is updated. The foundation of the algorithm is a simple value iteration update proposed in [22]. Moreover, it updates the following equation to search the optimal action:

$$Q_{t+1}(s_t,a_t) = Q_t(s_t,a_t) + \alpha\left[r_{t+1} + \gamma \max Q_t(s_{t+1},a_t) - Q_t(s_t,a_t)\right] \quad (8)$$

Here, r_{t+1} represents reward values obtained by one controller during interacting with environment; γ represents the discount rate and α is the learning rate. In this experiment, they are both set to 0.9.

Obviously, the system transporters fulfill the following conditions, and then it has been proved that the control policy converges to Q^* when $t \to \infty$ [23].

- Explicit, distinctive values of the Q-function are stored and updated for each state-action pair
- The sum of the squares of α is finite, whereas the sum of α is infinite.
- The controller keeps trying all actions in all states with nonzero probability.

The completion time of 50 episodes is shown in Figure 7.

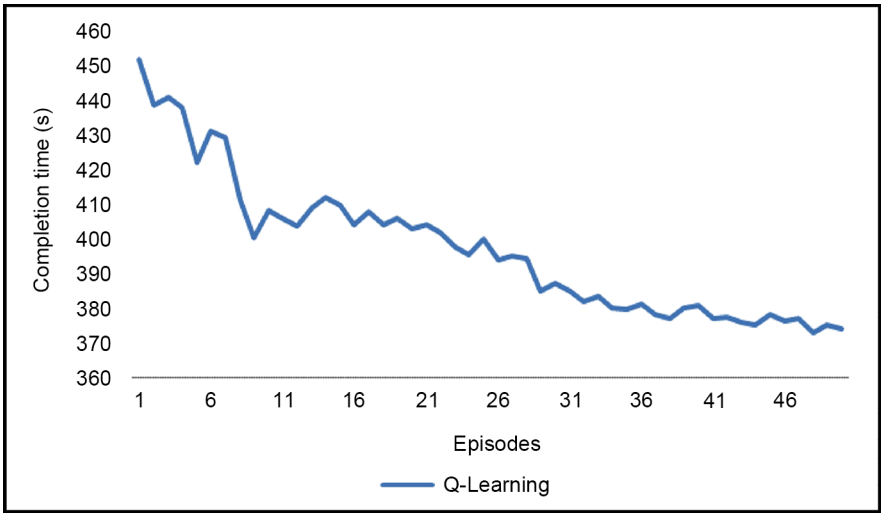

Figure 7. Time consumption for 50 tests in the simulation.

From the simulated 50 trials, the completion time tends to decrease. At the very beginning, it shrinks quickly, and then slows down. The curve is not smooth because the range of the tour time is discontinuous. Furthermore, the curve fluctuates a lot because of the system structure and the relationship between nodes and routes. The control system will traverse as many initial populations as possible at the very beginning. The influence of a local decision is comprehensive and in-depth. One previous decision affects the tours of several following products, thus affects the total completion time. Once an option of a previous product is selected, the tour range of the following product shrinks. The sharp time decline occurs when the algorithm recognizes one crucial node, while the subsequent rise is always tiny because the algorithm tries to promote the system performance in the neighborhood. After 30 tests, the curve tends to be mild. Therefore, it can be convinced that the system has completed the training process.

5. Further Discussion

5.1. The Influence of Transporting Robot on the System Performance

There is another part of the system, *i.e.* the feeding and palletizing robot. Though it is defined as a subsidiary device, it will also influence the performance of the system. For example, if it chooses to continue to feed a raw product to the system when a finished product appears on the exit conveyor belt, the palletizing process must be delayed, and vice versa. The mean tardiness time is employed as an indicator to elaborate that influence. The tardiness time is defined as the time delay from the appearance of a raw product at the head of the scheduling list to its first appearance at the entrance conveyor belt. The average tardiness time is displayed in Figure 8.

It is obvious that test 23 has a significant shorter completion time than test 3 by the evaluation of completion time. Synthesizing the results from Figure 7 and

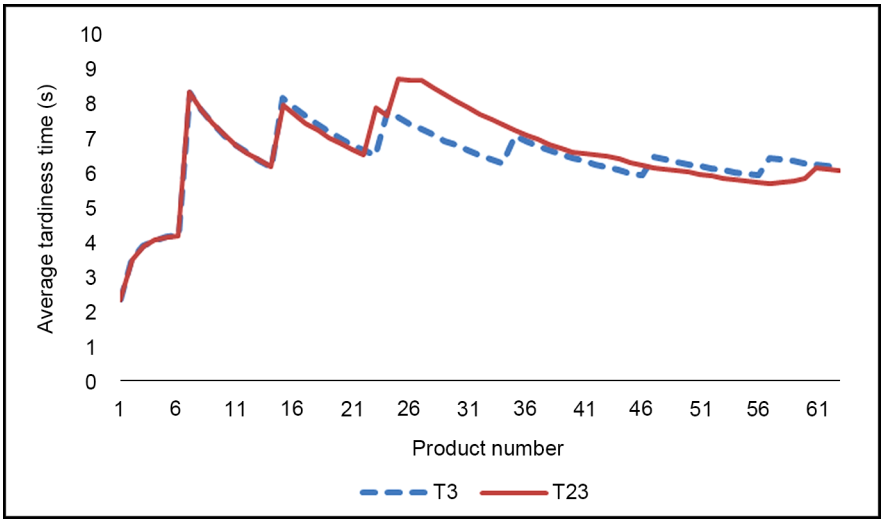

Figure 8. A comparison example of average tardiness time in the 3rd test and 23rd test.

Figure 8, we can find the flatter the curve of average tardiness, the shorter the time consumption. For the manufacturing system, it needs to find the best way (use case 1 and use case 2), and arrange the combination of routes (use case 3). And for the feeding and palletising subsystem, it needs to decline average tardiness time. When these two points are completed, the system can execute with the optimal performance.

Figure 8 indicates the inherent direction of the system optimization. The oscillation of average tardiness time in the early stage of the test, e.g. the transportation process of the first 15 product, is ascribed to choices of the robot to fetch a raw product first or a completed product. The more number of oscillations signifies the more alternatives between feeding and palletising selection of the robot. If we have more positions for both raw product and finished product in the inventory, we will get a smoother curve of average tardiness time.

Comparison of test 3 and test 23 shows that if the average tardiness time is viewed as a sequence, the limit of the sequence is decided by the physical structure of the system, but not the number of the product.

5.2. System Structure

Based on the analysis and simulation results of different nodes in the system, it is found that the topological structure has a significant influence. A flexible manufacturing system can be controlled and improved more conveniently by a corresponding machine learning algorithm due to the more reasonable nodes topological structure in the design process. In return, an intelligent machine learning algorithm can enhance the design of a flexible manufacturing system.

It is evident that the best position for a unique workstation is by node nine if there is only one kind of a product. At that moment, the complexity of the system is reduced to a simple manufacturing system. Even though work station one is fixed by the conveyor belt between node 2 and node 4, the system behaviour (e.g. completion time, average tardiness time) will be totally changed.

6. Conclusions

In this paper, the model of a multi-branches complex manufacturing system is elaborated and formulated. Then a hybrid algorithm integrating messy GA and RL algorithm is proposed to control the fabrication process based on system structural analyses and recipe specification. Messy GA is employed to represent route options, and RL algorithm is used to evaluate and to improve system performance. Next, implementation of the hybrid control algorithm is simulated on IEC 61131 environment. Results of the simulation prove that the algorithm can significantly cut down the time consumption.

Besides that, the result can be beneficial for the system design in the topological structure level. The position of workstations and nodes can be better planned to cut down the time consumption on the route planning or to avoid conflicts between two or more products when they need to pass through one node.

Further research should focus on two prospects. First, it is necessary to simulate more complicated scenarios in a production session which can put forward the control policy for the complex manufacturing system. For instance, some products only pass workstation 1, and some products pass both workstations in one process. Besides, the production order is uncertain, and needs to be scheduled by the control system. Second, it will be beneficial to find the optimal position of a workstation without changing the structure of a system.

References

[1] Leitão, P. (2009) Agent-Based Distributed Manufacturing Control: A State-of-the-Art Survey. *Engineering Applications of Artificial Intelligence*, **22**, 979-991. https://doi.org/10.1016/j.engappai.2008.09.005

[2] Gould, P. (1997) What Is Agility? [Agile Manufacturing]. *Manufacturing Engineer*, **76**, 28-31. https://doi.org/10.1049/me:19970113

[3] Jayal, A.D., Badurdeen, F., Dillon, O.W. and Jawahir, I.S. (2010) Sustainable Manufacturing: Modeling and Optimization Challenges at the Product, Process and System Levels. *CIRP Journal of Manufacturing Science and Technology*, **2**, 144-152. https://doi.org/10.1016/j.cirpj.2010.03.006

[4] Lee, J., Lapira, E., Bagheri, B. and Kao, H.A. (2013) Recent Advances and Trends in Predictive Manufacturing Systems in Big Data Environment. *Manufacturing Letters*, **1**, 38-41. https://doi.org/10.1016/j.mfglet.2013.09.005

[5] Garey, M.R., Johnson, D.S. and Sethi, R. (1976) The Complexity of Flowshop and Jobshop Scheduling. *Mathematics of Operations Research*, **1**, 117-129. https://doi.org/10.1287/moor.1.2.117

[6] Van Laarhoven, P.J., Aarts, E.H. and Lenstra, J.K. (1992) Job Shop Scheduling by Simulated Annealing. *Operations Research*, **40**, 113-125. https://doi.org/10.1287/opre.40.1.113

[7] Chen, J.C., Wu, C.C., Chen, C.W. and Chen, K.H. (2012) Flexible Job Shop Scheduling with Parallel Machines Using Genetic Algorithm and Grouping Genetic Algorithm. *Expert Systems with Applications*, **39**, 10016-10021. https://doi.org/10.1016/j.eswa.2012.01.211

[8] Karthikeyan, S., Asokan, P., Nickolas, S. and Page, T. (2015) A Hybrid Discrete Firefly Algorithm for Solving Multi-Objective Flexible Job Shop Scheduling Prob-

lems. *International Journal of Bio-Inspired Computation*, **7**, 386-401. https://doi.org/10.1504/IJBIC.2015.073165

[9] Jimenez, J.F., Bekrar, A., Zambrano-Rey, G., Trentesaux, D. and Leitão, P. (2016) Pollux: A Dynamic Hybrid Control Architecture for Flexible Job Shop Systems. *International Journal of Production Research*, 1-19. https://doi.org/10.1080/00207543.2016.1218087

[10] Schaffer, R.A.C.J.D. and Eshelman, L.J. (2016) Gray and Binary Coding for Genetic Algorithms. *Machine Learning Proceedings*, Ithaca, 26-27 June 1989, 375-378.

[11] Yuan, Q. and Yang, Z. (2013) A Weight-Coded Evolutionary Algorithm for the Multidimensional Knapsack Problem. arXiv:1302.5374

[12] Sastry, K., Goldberg, D.E. and Kendall, G. (2014) Genetic Algorithms. In: Burke, E.K. and Kendall, G., Eds., *Search Methodologies*, Springer, Berlin, 93-117. https://doi.org/10.1007/978-1-4614-6940-7_4

[13] Zhang, G., Gao, L. and Shi, Y. (2011) An Effective Genetic Algorithm for the Flexible Job-Shop Scheduling Problem. *Expert Systems with Applications*, **38**, 3563-3573. https://doi.org/10.1016/j.eswa.2010.08.145

[14] Grefenstette, J., Gopal, R., Rosmaita, B. and Van Gucht, D. (1985) Genetic Algorithms for the Traveling Salesman Problem. *Proceedings of the* 1*st International Conference on Genetic Algorithms and their Applications*, Pittsburg, 24-26 July 1985, 160-168.

[15] Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T.A.M.T. (2002) A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, **6**, 182-197. https://doi.org/10.1109/4235.996017

[16] Ting, T.O., Wong, K.P. and Chung, C.Y. (2008) Hybrid Constrained Genetic Algorithm/Particle Swarm Optimisation Load Flow Algorithm. *IET Generation, Transmission & Distribution*, **2**, 800-812. https://doi.org/10.1049/iet-gtd:20070224

[17] Li, H. (2015) The Implementation of Reinforcement Learning Algorithms on the Elevator Control System. 20t*h Conference on Emerging Technologies & Factory Automation*, Luxembourg, 8-11 September 2015, 1-4. https://doi.org/10.1109/etfa.2015.7301554

[18] Sutton, R.S. (1992) Introduction: The Challenge of Reinforcement Learning. In: Sutton, R.S., Ed., *Reinforcement Learning*, Springer, Berlin, 1-3. https://doi.org/10.1007/978-1-4615-3618-5_1

[19] Li, H. (2016) An Approach to Improve Flexible Manufacturing Systems with Machine Learning Algorithms. 42*nd Annual Conference of the IEEE Industrial Electronics Society*, Florence, 23-26 October 2016, 54-59.

[20] Goldberg, D., Deb, K. and Korb, B. (1989) Messy Genetic Algorithms: Motivation, Analysis, and First Results. *Complex Systems*, **3**, 493-530.

[21] Watkins, C.J. and Dayan, P. (1992) Q-Learning. *Machine Learning*, **8**, 279-292. https://doi.org/10.1007/BF00992698

[22] Bradtke, S.J. and Duff, M.O. (1995) Reinforcement Learning Methods for Continuous-Time Markov Decision Problems. *Advances in Neural Information Processing Systems*, **7**, 393-400.

[23] Jaakkola, T., Tommi Jordan, M.I. and Singh, S.P. (1994) On the Convergence of Stochastic Iterative Dynamic Programming Algorithms. *Neural Computation*, **6**, 1185-1201. https://doi.org/10.1162/neco.1994.6.6.1185