

A reinforcement learning based approach for a multiple-load carrier scheduling problem

Ci Chen · Beixin Xia · Bing-hai Zhou · Lifeng Xi

Received: 3 July 2013 / Accepted: 22 November 2013 / Published online: 19 December 2013
© Springer Science+Business Media New York 2013

Abstract This paper studies the problem of scheduling a multiple-load carrier which is used to deliver parts to line-side buffers of a general assembly (GA) line. In order to maximize the reward of the GA line, both the throughput of the GA line and the material handling distance are considered as scheduling criteria. After formulating the scheduling problem as a reinforcement learning (RL) problem by defining state features, actions and the reward function, we develop a $Q(\lambda)$ RL algorithm based scheduling approach. To improve performance, forecasted information such as quantities of parts required in a look-ahead horizon is used when we define state features and actions in formulation. Other than applying traditional material handling request generating policy, we use a look-ahead based request generating policy with which material handling requests are generated based not only on current buffer information but also on future part requirement information. Moreover, by utilizing a heuristic dispatching algorithm, the approach is able to handle future requests as well as existing ones. To evaluate the performance

of the approach, we conduct simulation experiments to compare the proposed approach with other approaches. Numerical results demonstrate that the policies obtained by the RL approach outperform other approaches.

Keywords Materials handling · Multi-criteria decision making · Reinforcement learning · Multiple-load carrier scheduling · Look-ahead scheduling

Introduction

A general assembly (GA) line is a system where parts are assembled into products. As an example, the GA line shown in Fig. 1 consists of a series of six assembly workstations. Before the GA line, there are incoming semi-products from upstream manufacturing processes. A semi-product will be first loaded to Workstation 1 of the GA line and then to Workstation 2 and go on. At each workstation, operators pick up certain types of parts from line-side buffers at the station and assemble them into the semi-product. The semi-product is then transferred to the next workstation or to the warehouse when it becomes a final-product after the assembly operation in Workstation 6 is finished. Because capacities of line-side buffers are limited, a dolly train is used by the material handling system (MHS) of the GA line to deliver parts from a stocking area to the line-side buffers. The dolly train is a multiple-load carrier that can carry multiple containers of parts at a time. To maximize the profit of the GA line, the MHS has to replenish parts in time to make sure that there are enough parts for assembly. Otherwise the throughput of the GA line cannot be maximized. Furthermore, because material handling cost plays an important part of manufacturing cost (Joe et al. 2012), the MHS has to minimize material handling cost. Therefore, an efficient material handling

C. Chen · B. Xia (✉) · L. Xi
Department of Industrial Engineering and Logistics Management,
Shanghai Jiao Tong University, No. 800 Dongchuan Road,
Shanghai 200240, China
e-mail: bxxia1984@163.com

C. Chen
e-mail: cichen@outlook.com

L. Xi
e-mail: lfxi@sjtu.edu.cn

B. Xia
School of Mechatronics Engineering and Automation,
Shanghai University, 99 Shangda Road, Shanghai 200444, China

B. Zhou
Department of Industrial Engineering, Shanghai Tongji University,
Shanghai, China
e-mail: bhzhou@tongji.edu.cn

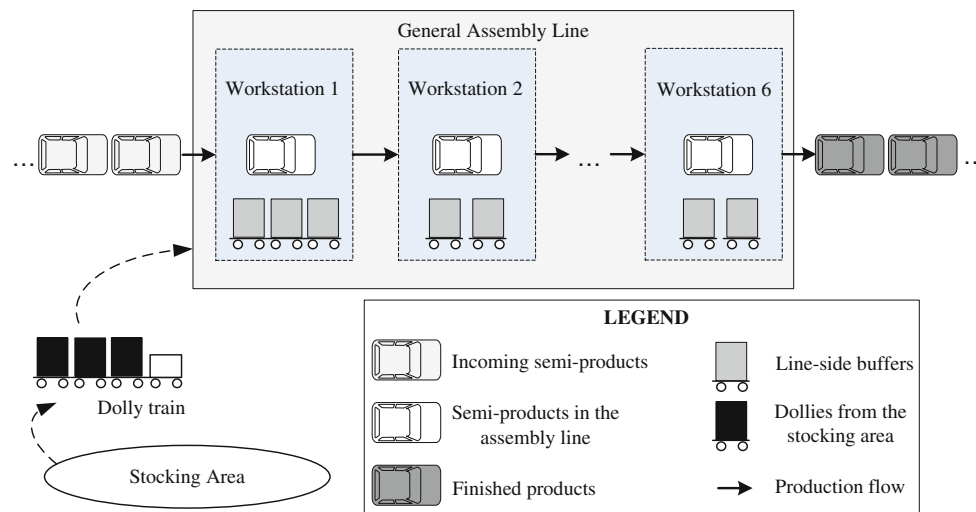


Fig. 1 A general assembly line

scheduling approach is very important for the GA line. In this paper, we study the problem of scheduling the dolly train in a real-time fashion. Because we aim at optimizing the profit of the GA line, both the throughput of the GA line and the material handling distance are considered.

However, the material handling scheduling problem considered here is very difficult. Firstly, according to our previous study (Chen et al. 2011), it's very difficult to optimize both criteria at the same time even if it's not impossible. Higher throughput usually means longer material handling distance while shorter distance often indicates lower throughput. Secondly, due to the fluctuation and variation of market requirement and production plans, the consumption rates of parts are not constant from a short-term perspective. Therefore we should find a policy that is able to schedule according to the real-time status of the GA line. Thirdly, information necessary for scheduling of the entire scheduling period is not available before scheduling in practice. Actually, at each decision epoch, the scheduling system can only access to information within a short period of time. Hence, we need to make decisions based on local information despite that we focus on long-term performance. As a result, it's important but difficult to evaluate the performance of a decision from a long-term perspective. Furthermore, scheduling multiple-load carriers is considered to be far more complicated than scheduling unit-load carriers (Berman et al. 2009).

The importance and the difficulties of material handling scheduling problems have attracted a lot of attentions. In the literature, different scheduling approaches have been developed. For example, Ozden (1988) use a simple pick-all-send-nearest rule for a carrier scheduling problem. There are also other researches study simple scheduling approaches which consider only a few attributes of system status (e.g., Occena and Yokota 1993; Nayyar and Khator 1993).

Sinriech and Kotlarski (2002) categorize multiple-load carrier scheduling rules into three categories, namely idle carrier dispatching rules, load pick-up rules and load drop-off rules.

To further improve the performance of MHSs, more complex methods have been proposed (e.g. Potvin et al. 1992; Chen et al. 2010; Dang et al. 2013). Such approaches include fuzzy logic based methods (Kim and Hwang 2001), genetic algorithm based methods (Orides et al. 2006), particle swarm optimization based methods (Belmecheri et al. 2013), neural network based methods (Min and Yih 2003; Wang and Chen 2012), hybrid meta-heuristic methods (Vahdani et al. 2012), etc. Vis (2006), Le-Anh and De Koster (2006) and Sarin et al. (2010) provide comprehensive reviews on material handling scheduling problems.

Recently, because of RL methods' ability in finding optimal/near-optimal policies in dynamic environments, RL algorithms have been applied to production scheduling problems (e.g., Wang and Usher 2005; Gabel and Riedmiller 2011; Zhang et al. 2012). Li et al. (2002) apply an RL algorithm based on Markov games to determine an agent's best response to others' response in an automated-guided vehicle dispatching problem. Jeon et al. (2011) consider the problem of determining shortest-time routes for vehicles in port terminals. They use Q-learning methods to estimate waiting times of vehicles during travelling. However, previous studies have not thoroughly explored the application of RL algorithms to material handling scheduling problems, especially to the multiple-load carrier scheduling problem considered in this paper.

Furthermore, most of the studies in the literature focus on manufacturing systems where MHSs are used to transfer materials between workstations. In these type of manufacturing systems, a material handling request is generated

whenever a workstation completes a load (see Ho et al. 2012 for an example). It's quite obvious to do so because after a workstation completes a load, the load should be delivered to another workstation for processing. On the other hand, for the GA line we study here, when to generate material handling requests is a question need to be answered first by scheduling approaches. In automotive industry, the reorder-point (ROP) policy is usually applied in practice. With the ROP policy, a request for the i th type of part, P_i , will be generated when the inventory level of P_i 's buffer is no more than its reorder point, i.e., $buf_i \leq RP_i$. However, using the ROP policy in a dynamic environment could cause problems sometimes. For instance, if $buf_i = 0$ but P_i is not required in the near future, requesting for P_i is not necessary and could lead to extra cost (such as holding cost).

Although it's impossible to obtain information of the entire scheduling period before scheduling in practice, information in a look-ahead horizon is usually available even in a stochastic environment. To improve performance of MHSs in such dynamic and stochastic manufacturing environments, it's necessary to develop look-ahead methods that consider forecasted information (Le-Anh et al. 2010). De Koster et al. (2004) show that using look-ahead information leads to significant improvement in the performance for a unit-load carrier scheduling problem. On the other hand, Grunow et al. (2004) develop an algorithm which considers orders initiated in a look-ahead time window for a multiple-load AGV dispatching problem. However, because these studies pay no attention to the problem of generating material handling requests, we need to further investigate how to incorporate look-ahead information in material handling request generating policies.

Moreover, in most studies, MHSs are only allowed to handle existing requests. This is because every request corresponds to a load completed by a workstation in those studies. And it's impossible to move a load that is not yet output from a workstation. However, there is no such restriction for the case we study here. The dolly train is allowed to replenish P_i even though there is no request for P_i yet. Actually, it's beneficial to do so sometimes. For instance, consider two types of parts, P_1 and P_2 , whose buffer locations are close to each other. Assume that there is a request for P_1 but no request for P_2 at time t_0 . Also assume that buf_2 is very close to RP_2 , which means a request for P_2 will soon be generated shortly after t_0 . Then it's better to replenish P_1 together with P_2 . Otherwise the dolly train has to start a new trip for P_2 after P_1 is replenished. Hence, it's necessary to develop an approach to consider both existing requests and future requests.

To address the issues above, we first propose a new method to generate material handling requests based on forecasted information in a look-ahead horizon. We also propose a heuristic dispatching algorithm that is able to consider future requests as well as existing requests. Using the new request

generating policy and the heuristic algorithm, we formulate the material handling scheduling problem as a reinforcement learning (RL) problem by defining state features, actions and reward function. Finally, we apply a $Q(\lambda)$ algorithm to find the optimal policy of the material handling scheduling problem.

The remainder of the paper is organized as follows. The next section describes the scheduling problem investigated in this paper and formulates it as an RL problem. After problem formulation, the RL based scheduling approach is proposed. Next, simulation experiments are conducted to evaluate the performance of the proposed approach. The last section draws conclusions and highlights areas for further research.

Problem formulation

Notations

The following notations will be used throughout this paper:

| | |
|--------------|--|
| N_w | the number of workstations in the GA line; |
| N_p | the number of part types will be assembled into semi-products; also the number line-side buffers; |
| N_A | the number of product models; |
| N_c | the capacity of a dolly train, i.e., the maximum number of containers a dolly train can handle at the same time; |
| CT | the cycle time of the GA line; |
| P_i | the i th type of parts; |
| SPQ_i | standard pack quantity of P_i , i.e., the quantity of P_i that will be loaded to one container in the stocking area; also the capacity of P_i 's line-side buffer; |
| buf_i | the inventory level of P_i 's line-side buffer; |
| ULL_i | the threshold that determines whether extra unloading time is needed to unload P_i |
| t_{load} | loading time per container; |
| t_{unload} | unloading time per container; |
| t_{extra} | extra unloading time per container |
| d_i | the distance between the stocking area and P_i 's line-side buffer; |
| T | the length of the entire scheduling period; |
| M | the throughput of the GA line in T ; |
| w_M | the coefficient of M ; |
| D | the material handling distance in T ; |
| w_D | the coefficient of D ; |
| v | the velocity of a dolly train; |
| l_i | the maximum number of assembly orders that P_i 's line-side buffer can support without replenishment; |

| | | | |
|---------------------------|---|-----------------|---|
| $W(i)$ | the workstation that P_i is assembled into semi-products; | G | the number of RBFs; |
| $I(W, t)$ | the no. of the semi-product that is the first to enter workstation W after time t ; | θ_g^a | the weights of $\phi_g(\mathbf{s})$; |
| $B(j)$ | the model of the j th semi-product; | \mathbf{C}_g | the center of $\phi_g(\mathbf{s})$; |
| $H(B, i)$ | the quantity of P_i required by a semi-product whose model is B ; | σ_g | the width of $\phi_g(\mathbf{s})$; |
| $J(t)$ | the no. of the latest semi-product arrived in the look-ahead period $[t, t + L]$; | Θ^a | $\Theta^a = (\theta_1^a, \theta_2^a, \dots, \theta_G^a)^T$ is the vector of weights; |
| u_i | the quantity of P_i required by a semi-product on average; | Φ | $\Phi = (\phi_1(\mathbf{s}), \phi_2(\mathbf{s}), \dots, \phi_G(\mathbf{s}))^T$ is the vector of RBFs; |
| RP_i | P_i 's reorder point defined in the ROP policy; | $\mathbf{Y}(a)$ | $\mathbf{Y}(a) = (y_1^a, y_2^a, \dots, y_G^a)^T$ is a vector of the eligibility traces for action a ; |
| RP_i^L | P_i 's reorder point defined in the look-ahead based reorder point policy; | | |
| n_{req} | the number of material handling requests generated; | | |
| t_{next} | the time left for the GA line to proceed to the next assembly cycle; $X = \{x_1, x_2, \dots, x_n\}$ is a drop-off sequence; x_k ($k = 1, 2, \dots, n$) is the type of parts in the k th container to deliver; | | |
| n | the number of loads to handle in a drop-off sequence; | | |
| t_{total} | the amount of time to execute a drop-off sequence; | | |
| t_{stop} | the amount of time that the GA line is not working during the execution of a drop-off sequence; | | |
| t_a | $t_a = t_{total} - t_{stop}$ is the actual working time of the GA line during the execution of a drop-off sequence; | | |
| SMD_i | the <i>standard material handling distance to deliver a unit quantity of P_i</i> ; | | |
| ARQ_i | the <i>actual replenished quantity of P_i</i> ; | | |
| $d_{eff}(X)$ | the effective material handling distance during the execution of X ; | | |
| $d_{total}(X)$ | the total material handling distance to execute X ; | | |
| $d_{saved}(X)$ | the total saved distance to execute X ; | | |
| ASR | the average standard reward to assemble a unit product; | | |
| ρ | the standard reward rate; | | |
| $SR(X)$ | the standard reward received during the execution of X ; | | |
| $ER(X)$ | the expected reward received during the execution of X ; | | |
| $\tilde{r}(X)$ | the expected reward rate during the execution of X ; | | |
| \mathbf{s}, \mathbf{s}' | state vectors | | |
| t_{wait} | the maximum amount of time that the dolly train should wait in the stocking area until the next decision epoch; | | |
| N_a | the number of actions; | | |
| $\phi_g(\mathbf{s})$ | the g th Gaussian radial basis function (RBF); | | |

Problem description

The multiple-load carrier scheduling problem considered in this paper can be described as follows. The GA line is a serial line consisting of N_w workstations. The workstations are synchronized because of the usage of a synchronized conveyor transferring semi-products between workstations. Therefore, the cycle time CT is equal to the longest cycle time of the workstations; and the entire GA line will stop working whenever a workstation stops. During assembly, parts will be assembled into semi-products in these workstations. The GA line employs a dolly train whose capacity is N_c to replenish parts. To schedule the dolly train, the MHS usually needs to (1) generate material handling requests based on a request generating policy (e.g., the ROP policy); (2) decide whether the dolly train should deliver parts or wait in the stocking area when it is idle; (3) select parts to deliver if “deliver” decision has been made; and (4) determine the drop-off sequence for the parts selected. Therefore, the multiple-load carrier scheduling problem involves four sub-problems. For clarity and simplicity, the following assumptions are made:

- (1) If the dolly train is idle, it must stay in the stocking area unless there are one or more material handling requests;
- (2) If the GA line is stopped because there is no enough parts for assembly, the dolly train is not allowed to be idle in the stocking area;
- (3) If the dolly train is busy, it must finish tasks currently assigned to it before it can be dispatched to handle new tasks;
- (4) Drop-off sequence of parts cannot be changed once it's determined;
- (5) The dolly train's velocity v is constant;
- (6) The types of parts in different containers carried by the dolly train at the same time must be different;
- (7) Different types of parts cannot be mixed in the same container;
- (8) Breakdown and maintenance of the dolly train are not considered;
- (9) If the GA line is *not* stop and $buf_i > ULL_i$, it is not allowed to unload P_i ;

- (10) It takes t_{unload} time units to unload P_i if $buf_i \leq ULL_i$. Otherwise it takes $t_{unload} + t_{extra}$ time units, because more efforts are required in such cases;
- (11) buf_i is equal to SPQ_i right after P_i is replenished;
- (12) The distance between the unloading point of P_i and that of P_j is given by $|d_i - d_j|$;
- (13) The first workstation (Workstation 1) will never starve;
- (14) The last workstation (Workstation N_w) will never be blocked;
- (15) The semi-products arrive to the GA line are numbered sequentially starting from 1; and
- (16) The information of semi-products that will arrive to the GA line in the next L time units is assumed to be available.

The objective of the problem is to maximize f_T , the total reward of a given scheduling period T , i.e.,

$$f_T = w_M M - w_D D. \tag{1}$$

Look-ahead based request generating policy

In this section, we first discuss the problem of generating material handling requests because it is the basis to make other decisions.

As described in Introduction, the MHS will monitor the inventory levels when the ROP policy is used. It will generate a request for P_i if the following condition is met and there is no request for P_i yet.

$$buf_i \leq RP_i, \quad \forall i \in \{1, 2, \dots, N_p\}. \tag{2}$$

When assembly orders arrive in a way that models of products distribute evenly over time, larger buf_i means longer remaining life of P_i 's buffer, i.e., P_i 's buffer can support more assembly cycles without replenishment. However, this is usually not the case in a dynamic environment where consumption rates of parts fluctuate frequently from a short-term perspective. Therefore, it's possible to improve the performance of the scheduling system by using a better request generating policy in such an environment.

Since buf_i is not a good indicator of the remaining life of P_i 's buffer in a dynamic environment, we directly monitor the remaining lives of buffers instead of the inventory levels, i.e., the inequities (2) become

$$l_i \leq RP_i^L, \quad \forall i \in \{1, 2, \dots, N_p\}. \tag{3}$$

Then the request generating policy used in this paper can be described as: generate a request for P_i if condition (3) is satisfied and there is no request for P_i yet.

Unlike the inventory level, however, the remaining life of a buffer cannot be directly observed. In order to determine l_i for each P_i , the following algorithm (Algorithm 1) is used.

Algorithm 1.

```

For  $i = 1$  To  $N_p$ 
  1. Initialize  $b_i = buf_i, l_i = 0, j = I(W(i), t)$ 
  2. While  $j \leq J(t)$  AND  $b_i \geq 0$ 
    a.  $b_i = b_i - H_A(B(j), i)$ 
    b.  $j = j + 1$ 
    c.  $l_i = l_i + 1$ 
  End While
  3. If  $b_i \geq 0$  //the buffer is enough for all of the incoming orders
     $l_i = l_i + \lceil (b_i + 1) / u_i \rceil$ 
  End If
End For
    
```

Because the information of semi-products that arrive in the near future is required to calculate l_i s in Algorithm 1, the request generating policy used in this paper is a look-ahead request generating policy (LRGP).

In order to analyze the computational complexity of Algorithm 1, let $n_{orders} = J(t) - I(N_w, t)$ be the number of semi-products currently in the GA line plus the number of semi-products that will arrive in the following L time units. Then it takes $O(n_{orders})$ efforts in the Step 2. And since we have N_p types of parts, it takes $O(n_{orders}N_p)$ efforts to compute all the l_i s.

A heuristic dispatching algorithm

After a “deliver” decision has been made by the MHS, dispatching approaches are used to select parts to deliver and determine their drop-off sequence. When requests correspond to outputs from workstations, it's impossible to handle requests not yet generated because the corresponding loads don't exist. Therefore, dispatching approaches used for such environments consider only existing requests. However, this is not the case for the problem we study in this paper. The MHS is allowed to replenish parts even though there are no requests for them. And as discussed in the Introduction, it could be beneficial to do so.

Because material handling requests are generated based on remaining lives of buffers, those types of parts that are requested should be given higher priorities than others that are not. Otherwise the GA line might stop working because the parts requested replenishment might become out of stock. Assume that there exists a load pick-up rule $RPICK$ and a load drop-off rule $RDROP$ to select requests and determine an initial drop-off sequence. Then the dispatching problem can be simplified as selecting types of parts that are not yet requested and then inserting them to the initial drop-off sequence.

To implement the above idea, we first define the *expected reward rate* \tilde{r} for a drop-off sequence.

Let $X = \{x_1, x_2, \dots, x_n\}$ be a drop-off sequence. And “executing” X means delivering all the loads in X . Let SMD_i , which is given by Eq. (4), be the *standard material handling distance to deliver a unit quantity of P_i* ; and ARQ_i , which is given by Eq. (5), be the *actual replenished quantity of P_i* . Then $d_{eff}(X)$ is defined by Eq. (6).

$$SMD_i = 2d_i / SPQ_i \quad (4)$$

$$ARQ_i = SPQ_i - buf_i \quad (5)$$

$$\begin{aligned} d_{eff}(X) &= \sum_{k=1}^n SMD_{x_k} ARQ_{x_k} \\ &= \sum_{k=1}^n \frac{2d_{x_k}}{SPQ_{x_k}} (SPQ_{x_k} - buf_{x_k}) \end{aligned} \quad (6)$$

It is easy to see that the $d_{eff}(X)$ equals to the *total standard material handling distance* to deliver the same amount of parts as X does. Then $d_{saved}(X)$ is defined by Eq. (7).

$$d_{saved}(X) = d_{eff}(X) - d_{total}(X). \quad (7)$$

For two different sequences X and X' whose loads are the same, i.e., $\forall x_k \in X \Leftrightarrow x_k \in X'$, X is considered to be more efficient than X' in terms of material handling distance if $d_{saved}(X) > d_{saved}(X')$, because the distance required to replenish the same amount of parts is relatively shorter by executing X instead of X' .

According to Eqs. (1) and (4), the *average standard reward* to assemble a unit product is $ASR = w_M - w_D \sum_{i=1}^{N_p} u_i SMD_i$. Therefore we have $\rho = ASR/CT$. And the *standard reward* received during the execution of X is given by

$$SR(X) = \rho t_{total}. \quad (8)$$

Then the *expected reward* $ER(X)$ is defined as

$$\begin{aligned} ER(X) &= SR(X) - \rho t_{stop} + w_D d_{saved}(X) \\ &= \rho t_a + w_D d_{saved}(X), \end{aligned} \quad (9)$$

Finally, the expected reward rate $\tilde{r}(X)$ is given by

$$\tilde{r}(X) = ER(X) / t_{total}. \quad (10)$$

A drop-off sequence X can be considered to be more efficient with a greater $\tilde{r}(X)$ because the GA line receives more reward per unit time in such case. Based on this idea, we propose the following heuristic algorithm (Algorithm 2) to pick up loads and determine their drop-off sequence.

Algorithm 2.

1. If $n_{req} = 0$
 Terminate the algorithm
 Elseif $n_{req} < N_c$
 Select all the requests
 Else
 Select N_c requests according to the load pick-up rule R_{PICK}
 End If
2. Apply the load drop-off rule R_{DROP} to determine a drop-off sequence X .
3. If $|X| = N_c$ Then
 Go to Step 5
 End If
4. Repeat
 - a. $n = |X|$, $X^* = X = \{x_1^*, x_2^*, \dots, x_n^*\}$
 - b. For each $P_i \notin X^* // P_i$ is the type of part to be inserted to X^*
 For $k = 1$ To $n + 1$ // the position of P_i to be inserted
 $X' = \{x_1^*, x_2^*, \dots, x_{k-1}^*, P_i, x_k^*, \dots, x_n^*\}$ // a new candidate sequence
 If $\tilde{r}(X') > \tilde{r}(X)$ Then // the new candidate is better
 $X = X'$
 End If
 End For
 End For
5. Until $|X| = N_c$ OR $X = X^*$
5. Use X as the drop-off sequence.

With the above algorithm, the dolly train will always pick up as many requests as possible. If its capacity is not exceeded, it will also select other types of part not yet selected when the expected reward rate can be improved to do so (i.e., a greater $\tilde{r}(X)$ is achieved).

Note that R_{PICK} and R_{DROP} are not specified in Algorithm 2. This allows us to specify suitable rules in different situations. After all, there is no such rule that is globally optimal (Montazeri and Wassenhove 1990). By specifying different load pick-up rules and load drop-off rules in the above algorithm, we can also define different actions for the RL problem. Because the shortest-slack-time-first-picked (SSFP) rule and the shortest-distance-first-picked (SDFP) rule outperform other load pick-up rules and the shortest-distance-first-delivered (SDFD) rule outperforms other load drop-off rules according to Chen et al. (2011), we will use the SSFP rule and the SDFP rule as load pick-up rules and the SDFD rule as the load drop-off rule to define the actions.

The worst-case computational complexity of Algorithm 2 is analyzed as follows. Because that the exact load pick-up and load drop-off rules are specified later, we focus on analyzing Step 4, the process to generate new drop-off sequences. In the inner loop of Step 4.b, we have $n + 1 \leq N_c$ because the capacity of the dolly train is N_c . Therefore it takes $O(N_c N_p)$ efforts in Step 4.b. After Step 4.b, X either remains the same or is replaced by a new candidate sequence X' which satisfies $|X'| = |X| + 1$. The outer loop of Step 4 will stop in the former case and will continue if $|X| < N_c$ in the latter one. Hence, it takes $O(N_c)$ efforts in the outer loop of Step 4. Therefore the worst-case computational complexity of Step 4 is $O(N_c^2 N_p)$.

Reinforcement learning formulation

In order to apply RL algorithms to scheduling problems, we need to formulate the scheduling problems as RL problems. There are three major steps in the formulation, including defining state features, constructing actions and defining the reward function (Zhang et al. 2011).

State features

The objective of an RL algorithm is to find an optimal policy that is able to choose optimal actions for any given state. Theoretically, anything that is variable and related to the GA system (either the GA line itself or the MHS), such as status of workstations, status of the dolly train, information of semi-products that are being assembled currently, incoming assembly orders, etc., can be considered as states. However, for a complex manufacturing system, it's impossible to take all kinds of such information into consideration and it would make the RL problem way too complex. Therefore, we need to carefully define state features concisely such that the state space is compressed and it's easier for learning. More importantly, they should be able to represent the major characteristics of the GA system that impact the scheduling of MHS considerably. In this paper, we define the state features as follows:

State feature 1 (s_1). Let s_1 indicate if the GA line is stopped and define

$$s_1 = \begin{cases} 0 & \text{if the GA line is working} \\ 1 & \text{if the GA line is stopped} \end{cases} \quad (11)$$

State feature 2 (s_2). Let t_{next} denote the time left for the GA line to proceed to the next assembly cycle and define

$$s_2 = t_{next}/CT. \quad (12)$$

State feature 3 ($s_{3,i}, i \in \{1, 2, \dots, N_p\}$). $s_{3,i}$ is defined as

$$s_{3,i} = buf_i/SPQ_i. \quad (13)$$

State feature 4 ($s_{4,i}, i \in \{1, 2, \dots, N_p\}$). Let l_i^{MAX} be the maximum possible value of l_i which is given by

$$l_i^{MAX} = \left\lceil \frac{L}{CT} + W(i) - 1 + \frac{(SPQ_i + 1)}{u_i} \right\rceil. \quad (14)$$

Then $s_{4,i}$ is defined as

$$s_{3,i} = l_i/l_i^{MAX}. \quad (15)$$

With the definition of the state features, the system state at time t can be represented as

$$s_t = (s_1, s_2, s_{3,i} (i = 1, 2, \dots, N_p), s_{4,i} (i = 1, 2, \dots, N_p)). \quad (16)$$

Note that actions are taken only at decision epochs. We call time t a decision epoch if one of the following conditions is met:

- (1) The dolly train becomes idle;
- (2) The GA line is stopped because a certain type of part becomes out of stock; and
- (3) A new material handling request is generated.

The states at decision epochs are also called decision states.

Actions

The actions of an RL problem are decisions made at decision epochs. There are a lot of decisions can be made at decision epochs for the multiple-load carrier scheduling problem we study here, such as waiting in the stocking area, dispatching the dolly train to deliver P_1 , dispatching the dolly train to deliver P_2 and P_3 with a drop-off sequence $X = \{2, 3\}$, dispatching the dolly train to deliver P_3 and P_4 with a drop-off sequence $X = \{4, 3\}$ and so on. It would be inefficient for learning if we define each possible decision as an action. Because the dolly train is either idle in the stocking area or dispatched to replenish parts, we define actions as follows:

Action 1 (a_1). Wait in the stocking area until one of the following conditions is met:

- (1) The dolly train has been idle for more than t_{wait} time units since the last Action 1 is taken; and
- (2) The system state transits to a new decision state.

Action 2 (a_2). Use Algorithm 2 to pick up loads and to determine the drop-off sequence with the SSFP rule as R_{PICK} and the SDFD as R_{DROP} , deliver the loads then come back to the stocking area.

Action 3 (a_3). Use Algorithm 2 to pick up loads and to determine the drop-off sequence with the SDFP rule as R_{PICK} and the SDFD as R_{DROP} , deliver the loads then come back to the stocking area.

After defining the actions, the action space can be represented as $\mathbf{A} = \{a_1, a_2, a_3\}$. And the number of actions is $N_a = |\mathbf{A}|$.

Reward function

Because the objective of the RL problem is to maximize the accumulated reward such that the reward of the GA line can be maximized, the definition of the reward function should be also related to both throughput and the material handling distance.

Consider the case of the state of the system at decision epoch t , s , transiting to a new state, s' , at the next decision

epoch t' after action $a_t \in \mathbf{A}$ is performed at t . Based on Eq. (9), the reward received in the period of $[t, t']$ is defined as

$$r(\mathbf{s}, a_t, \mathbf{s}') = \rho(t' - t - t_{stop}) + w_D d_{saved}. \tag{17}$$

The reinforcement learning based scheduling approach

In the previous section, we formulate the multiple-load carrier scheduling problem as an RL problem. To apply RL in scheduling, we need to solve the RL problem first. Because the time spent in state transition is part of the reward function, algorithms developed for problems in MDP context cannot be applied to the RL problem directly. In order to solve the RL problem, the discount factor γ is replaced by $e^{-\beta(t'-t)}$ and the reward received in the period of $[t, t']$ becomes

$$r(\mathbf{s}, a_t, \mathbf{s}') = \int_t^{t'} e^{-\beta(\tau-t)} \xi(\tau) d\tau, \tag{18}$$

where $\xi(\tau)$ ($\tau \in (t, t')$) is the reward rate function within $[t, t']$ which is given by

$$\xi(\tau) = [\rho(t' - \tau - t_{stop}) + w_D d_{saved}] / (t' - \tau). \tag{19}$$

Also note that all the state features are continuous except for s_1 . Thus the state space is infinite, which makes the use of tabular form RL algorithms impossible. In such cases, function approximators, such as linear approximator, neural networks approximator and kernelized approximator, etc. are usually used to approximate value functions. In this paper, we use a linear function with a gradient-descent method to approximate value function for each action a . Each approximator is a linear combination of a set of radial basis functions (RBFs), i.e.,

$$Q(\mathbf{s}, a) = \sum_{g=1}^G \theta_g^a \phi_g(\mathbf{s}), \tag{20}$$

where $\phi_g(\mathbf{s})$ ($g \in \{1, 2, \dots, G\}$) are Gaussian RBFs given by Eq. (21).

$$\phi_g(\mathbf{s}) = e^{-\|\mathbf{s}-\mathbf{C}_g\|^2/2\sigma_g^2} \tag{21}$$

According to Eqs. (20) and (21), we need to determine G , \mathbf{C}_g , σ_g and θ_g^a in order to approximate the Q-value function. In this paper, we set G to 20 by convenience and apply Hard K-Means (Duda and Hart 1973) based heuristic method to determine \mathbf{C}_g and σ_g . And then we use a gradient-descent method to update θ_g^a during learning. To solve the multiple-load carrier scheduling problem, a Q(λ) RL algorithm with function approximation is proposed as Algorithm 3.

Algorithm 3.

1. Initialize $\Theta^a = \mathbf{0}$ and $\mathbf{Y}(a) = \mathbf{0}$ for all $a \in \mathbf{A}$. Set $\mu, \gamma, \beta, \epsilon$ and λ to predetermined values.
2. Randomize system state and then start simulation until the system arrives at a decision epoch.
3. Let the current time be t and the system state be \mathbf{s} .
4. Choose $a^*(\mathbf{s}) = \arg \max_{a \in \mathbf{A}(\mathbf{s})} Q(\mathbf{s}, a)$ with probability of $1-\epsilon$ or a random action $a \in \mathbf{A}(\mathbf{s})$. Let a_t be the action selected.
5. Perform action a_t and simulate until the system arrives at the next decision epoch t' . Let the system state at t' be \mathbf{s}' . Observe d_{saved} and t_{stop} within the period of $[t, t']$.
6. If $a_t = a^*(\mathbf{s})$

$$\mathbf{Y}(a) = \lambda e^{-\beta(t'-t)} \mathbf{Y}(a)$$
 Else

$$\mathbf{Y}(a) = \mathbf{0}$$
 End If

$$\mathbf{Y}(a) = \mathbf{Y}(a) + \nabla_{\theta^a} Q(\mathbf{s}, a_t)$$
 Calculate $r(\mathbf{s}_t, a_t, \mathbf{s}_{t'})$ according to Eqs. (18) and (19).

$$\delta = r(\mathbf{s}_t, a_t, \mathbf{s}_{t'}) + e^{-\beta(t'-t)} \max_{a \in \mathbf{A}(\mathbf{s}')} Q(\mathbf{s}', a) - Q(\mathbf{s}, a_t)$$

$$\Theta^a = \Theta^a + \mu \delta \mathbf{Y}(a)$$
7. Set $t = t', \mathbf{s} = \mathbf{s}'$. If the stopping criterion is satisfied then terminate the algorithm, otherwise go to 4.

In the following we analyze the worst-case computational complexity of Algorithm 3. In Step 1, it takes $O(N_a G)$ efforts to initialize weights and eligibility traces. According to the computational complexity of Algorithm 1, it takes $O(n_{orders} N_p)$ efforts in Step 3 because we need to compute $l_i s$ to determine the system state \mathbf{s} . It takes $O(N_p N_a G)$ efforts to choose an action in Step 4. Similar to Step 3, Step 5 requires $O(n_{orders} N_p)$ efforts to determine the next system state \mathbf{s}' . In Step 6, it takes $O(N_p N_a G)$ efforts to update $\mathbf{Y}(a)$ and Θ^a and $O(n_{orders} N_p)$ efforts to calculate $r(\mathbf{s}_t, a_t, \mathbf{s}_{t'})$. Hence the worst-case computational complexity of an iteration of Algorithm 3 is $O(N_p(n_{orders} + N_a G))$.

Note that Algorithm 3 is executed before real-time scheduling. Once Algorithm 3 is terminated, Θ^a determined in Algorithm 3 can be used in multiple-load carrier scheduling. Let the system state be \mathbf{s} at an arbitrary decision epoch during scheduling. Then the scheduling system will take action $a^*(\mathbf{s}) = \arg \max_{a \in \mathbf{A}(\mathbf{s})} Q(\mathbf{s}, a)$ at the decision epoch. If $a^*(\mathbf{s}) = a_1$, the dolly train will wait until the next decision epoch. And if $a^*(\mathbf{s}) = a_2$, the dolly train will use Algorithm 2 with the SSFP rule as R_{PICK} and the SDFD as R_{DROP} to determine a drop-off sequence X . And then the dolly train will replenish parts according to X . Similarly, if $a^*(\mathbf{s}) = a_3$, the dolly train will also replenish parts. However, SDFP will be used as R_{PICK} in such cases.

Performance evaluation

To evaluate the performance of the proposed approach, we conduct simulation experiments to compare the approach with other scheduling approaches based on a numerical example.

Table 1 The layout of the GA line

| | P_1 | P_2 | P_3 | P_4 | P_5 | P_6 | P_7 | P_8 | P_9 | P_{10} | P_{11} | P_{12} | P_{13} | P_{14} |
|-----------------|-------|-------|-------|-------|-------|-------|-------|-------|-------|----------|----------|----------|----------|----------|
| Distance | 163 | 169 | 174 | 300 | 303 | 406 | 409 | 515 | 522 | 690 | 693 | 696 | 746 | 751 |
| Workstation no. | 1 | 1 | 1 | 2 | 2 | 3 | 3 | 4 | 4 | 5 | 5 | 5 | 6 | 6 |

“Workstation no.” is the no. of the workstation where the corresponding parts will be assembled (e.g., P_4 will be assembled to semi-products in Workstation 2)

Table 2 Bill of material and the SPQs

| | P_1 | P_2 | P_3 | P_4 | P_5 | P_6 | P_7 | P_8 | P_9 | P_{10} | P_{11} | P_{12} | P_{13} | P_{14} |
|-----|-------|-------|-------|-------|-------|-------|-------|-------|-------|----------|----------|----------|----------|----------|
| M1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| M2 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| M3 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| SPQ | 95 | 16 | 18 | 18 | 60 | 22 | 120 | 70 | 20 | 50 | 90 | 90 | 36 | 100 |

Experimental conditions

The GA line used for simulation experiments consists of six workstations. Operators will assemble 14 part types into three types of product models, M1, M2 and M3. The product mix of the three product models is (0.3, 0.3, 0.4) which means among all the products to be assembled, 30% are M1, 30% are M2 and 40% are M3. The cycle time of the GA line is 72 s. The capacity of the dolly train is three. The velocity of the dolly is 3.05 m/s. For loading and unloading operations, $t_{load} = 36.6$ s, $t_{unload} = 43.2$ s, and $t_{extra} = 60$ s. The look-ahead horizon is 1,080 seconds, i.e., $L = 1,080$ s. The coefficients of throughput and material handling distance, w_M and w_D , are 5,000 and 4.92, respectively. Table 1 gives the layout information of the GA line. Table 2 presents the bill of material (BOM) and the SPQs. The threshold for unloading is given by Eq. (22).

$$ULL_i = SPQ_i/10, \quad i = 1, 2, \dots, N_p \tag{22}$$

The scheduling approaches selected to compare with the RL approach include the minimum batch size (MBS-z, Neuts 1967) rule (including MBS-1, MBS-2 and MBS-3) based approaches, the next arrival control heuristic (NACH, Fowler et al. 1992) based approaches and the minimum cost rate (MCR, Weng and Leachman 1993) based approaches. These approaches use MBS-z rules, NACH algorithm or MCR algorithm to decide when to dispatch the dolly train to handle material handling requests. All of them use the SDFP rule to pick up loads and the SDFD rule to drop off loads. These approaches can use either ROP or LRGP to generate material handling requests. The reorder points of ROP and LRGP are given by Eqs. (23) and (24) respectively.

$$RP_i = \left\lceil \left[\left(\frac{1}{v} (d_i + 2 \times \max_j d_j) + \sigma \right) \times \frac{u_i}{CT} \right] \right\rceil, \quad i = 1, 2, \dots, N_p, \tag{23}$$

$$RP_i^L = \left\lceil \left[\left(\frac{1}{v} (d_i + 2 \times \max_j d_j) + \sigma^L \right) \times \frac{u_i}{CT} \right] \right\rceil, \quad i = 1, 2, \dots, N_p, \tag{24}$$

In Eqs. (23) and (24), σ and σ^L are parameters to change reorder point settings.

For the RL approach, we set $\mu = 0.001$ and $\beta = 0.00005$ using a heuristic method presented in Zhang et al. (2011). Because the cycle time is 72 s, t_{wait} should not be >72 . Otherwise the scheduling system is not able to react to changes of the GA line in time. And it is too frequent to make decisions with a too small t_{wait} . Hence, we set $t_{wait} = 20$ s after we search a suitable setting for t_{wait} within the interval [10, 72] through pilot simulation experiments.

Simulation results

Although Q-learning methods are proved to converge (Peng and Williams 1996), there is no convergence result for Q-learning methods with function approximators. Therefore, before we evaluate the performance of the proposed RL approach, we first study its convergence.

Let PD_k be the mean value of the square of the difference of all the Θ^a s between the $(k + 1)$ th iteration and the k th iteration, defined as Eq. (25) where $\phi_{g,k}^a$ denotes the value of ϕ_g^a in the k th iteration. Figure 2 presents how PD_k varies with the number of iterations. As Fig. 2 indicates, PD_k decreases sharply before 50,000 iterations. And then it decreases much more slowly after 100,000 iterations. Therefore Fig. 2 shows that the weights of RBFs become stable gradually and they converge asymptotically.

$$PD_k = \sum_{g=1}^G \sum_{a \in A} (\phi_{g,k+1}^a - \phi_{g,k}^a)^2 / N_a G \tag{25}$$

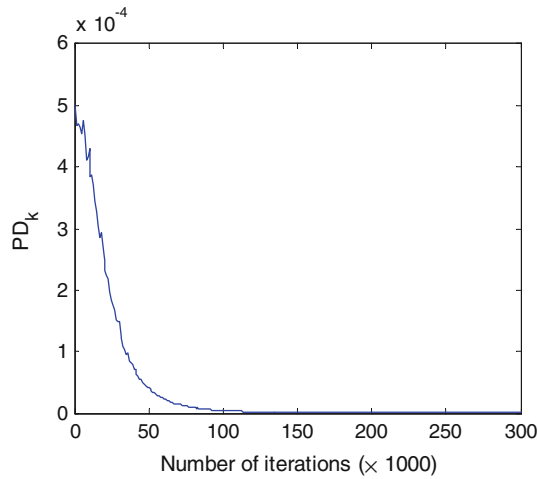


Fig. 2 Variation of PD_k with respect to the number of iterations

In order to evaluate the performance of the proposed RL approach, we conduct simulation experiments to compare throughput, material handling distance and the reward of different approaches. The simulation experiments consist of 50 tests, each of which lasts for 300h. And for each test, the warm-up period is 10h. In the experiments, both parameters for RP_i and RP_i^L , σ and σ^L , are set to 480. Note that Algorithm 3 is executed to solve the RL problem before real-time scheduling in the RL approach. The simulation results are shown in Table 3.

As shown in Table 3, there is no such approach that yields shortest distance and largest throughput at the same time, which agrees with Chen et al. (2011)’s observation. Among all the approaches, MBS-3(ROP), i.e. the MBS-3 approach using the ROP policy, is the best in terms of material handling distance while MBS-1(LRGP) is the worst. Their gap is 38.5%. This result indicates that if we want to minimize

the material handling distance, we should dispatch the dolly train to handle more requests at a time.

On the other hand, the proposed RL approach is the best approach in terms of throughput. Compared to the worst approach, MBS-3(LRGP), the RL approach improves the throughput by 23.1%. The RL approach is also the best approach that yields maximum reward. It’s 24.1% better than MBS-3(LRGP) which is the worst approach in terms of reward. The smallest gap between the RL approach and other approaches is 4.6%.

Because the cycle time is 72s, the maximum number of products the GA line can possibly assemble in 300h is $M^* = 300 \times 3,600/72 = 14,750$. Assuming that there is a hypothetical approach whose throughput is 14,750 and its distance is the same as the MBS-3(ROP) approach’s, then its reward would be 6.93×10^7 . Compared to the hypothetical approach’s performance, the RL approach’s performance is only 4.8% worse, which further indicates the effectiveness of the RL approach.

Considering that reorder point settings can affect the performance of scheduling, by changing σ and σ^L , we also conduct simulation experiments with different reorder point settings. In these experiments, all the parameters except for σ and σ^L are the same as the previous ones.

As shown in Table 4, the reorder point settings indeed affect the performance of different approaches. For example, the performance of MCR(LRGP) can be improved 5.7% by changing the σ^L from 60 to 540. However, the RL approach is still the best approach even with different reorder point settings. On average, the RL approach is 4.9% better than the second best approach.

Because prices of products and the cost per unit material handling distance might change over time due to the fluctuation of the dynamic environment, it’s very important to make sure that the proposed approach can perform well in

Table 3 Results of the case example

| Methods | Performance criteria | | | | | |
|-------------|----------------------------|----------|------------------------------|----------|--------------------------|----------|
| | Distance ($\times 10^6$) | Gap (%) | Throughput ($\times 10^4$) | Gap (%) | Reward ($\times 10^7$) | Gap (%) |
| MBS-1(ROP) | 1.27 | 29.2 | 1.37 | 3.9 | 6.24 | 5.7 |
| MBS-1(LRGP) | 1.46 | 38.5 | 1.34 | 6.6 | 5.96 | 10.6 |
| MBS-2(ROP) | 1.07 | 16.1 | 1.33 | 7.2 | 6.12 | 7.7 |
| MBS-2(LRGP) | 1.11 | 18.9 | 1.23 | 15.6 | 5.62 | 17.4 |
| MBS-3(ROP) | 0.90 | 0 | 1.27 | 11.9 | 5.92 | 11.3 |
| MBS-3(LRGP) | 9.65 | 6.7 | 1.16 | 23.1 | 5.32 | 24.1 |
| NACH(ROP) | 1.05 | 14.4 | 1.32 | 7.9 | 6.09 | 8.3 |
| NACH(LRGP) | 1.12 | 19.8 | 1.37 | 4.1 | 6.30 | 4.6 |
| MCR(ROP) | 0.91 | 1.5 | 1.28 | 11.0 | 5.97 | 10.4 |
| MCR(LRGP) | 0.93 | 3.4 | 1.31 | 8.7 | 6.10 | 8.2 |
| RL approach | 1.11 | 18.2 | 1.43 | 0 | 6.60 | 0 |

Gap (%) is calculated by $|\text{current value} - \text{best value}| / \text{current value} \times 100\%$. The best values in each column are highlighted in bold

Table 4 Results of experiments with different reorder point settings

| Methods | Reward ($\times 10^7$) | | | | | |
|-------------|--|-------------|-------------|-------------|-------------|-------------|
| | σ (For ROP) / σ^L (For LRGP) | | | | | |
| | 60 | 180 | 300 | 420 | 540 | 660 |
| MBS-1(ROP) | 6.23 | 6.23 | 6.26 | 6.22 | 6.25 | 6.18 |
| MBS-1(LRGP) | 5.71 | 5.83 | 5.89 | 5.94 | 5.97 | 5.98 |
| MBS-2(ROP) | 6.02 | 6.05 | 6.12 | 6.11 | 6.15 | 6.11 |
| MBS-2(LRGP) | 5.28 | 5.43 | 5.51 | 5.59 | 5.66 | 5.70 |
| MBS-3(ROP) | 5.70 | 5.78 | 5.87 | 5.89 | 6.00 | 5.95 |
| MBS-3(LRGP) | 5.06 | 5.17 | 5.23 | 5.30 | 5.37 | 5.41 |
| NACH(ROP) | 6.07 | 6.04 | 6.09 | 6.05 | 6.11 | 6.06 |
| NACH(LRGP) | 6.12 | 6.21 | 6.26 | 6.28 | 6.29 | 6.24 |
| MCR(ROP) | 5.73 | 5.78 | 5.85 | 5.88 | 5.95 | 5.94 |
| MCR(LRGP) | 5.80 | 5.93 | 6.00 | 6.08 | 6.12 | 6.12 |
| RL approach | 6.58 | 6.59 | 6.56 | 6.58 | 6.56 | 6.51 |

The best values in each column are highlighted in bold

Table 5 Results of experiments with different w_M

| Methods | Reward ($\times 10^7$) | | | |
|-------------|--------------------------|-------------|-------------|-------------|
| | $w_M=2,000$ | $w_M=3,000$ | $w_M=4,000$ | $w_M=6,000$ |
| MBS-1(ROP) | 2.12 | 3.49 | 4.87 | 7.61 |
| MBS-1(LRGP) | 1.95 | 3.29 | 4.63 | 7.30 |
| MBS-2(ROP) | 2.13 | 3.46 | 4.79 | 7.45 |
| MBS-2(LRGP) | 1.92 | 3.15 | 4.39 | 6.85 |
| MBS-3(ROP) | 2.10 | 3.38 | 4.65 | 7.20 |
| MBS-3(LRGP) | 1.84 | 3.00 | 4.16 | 6.47 |
| NACH(ROP) | 2.14 | 3.49 | 4.83 | 7.51 |
| NACH(LRGP) | 2.19 | 3.56 | 4.93 | 7.67 |
| MCR(ROP) | 2.12 | 3.40 | 4.69 | 7.26 |
| MCR(LRGP) | 2.18 | 3.51 | 4.84 | 7.49 |
| RL approach | 2.31 | 3.74 | 5.16 | 8.01 |

The best values in each column are highlighted in bold

such environments. Therefore, we conduct experiments with four different w_M values, 2,000, 3,000, 4,000 and 6,000. In all the experiments we use the same $w_D = 4.92$. And other parameters also remain the same.

Table 5 indicates that, the rewards of different approaches varies significantly because of the changes of w_M . Nevertheless, the proposed RL approach still outperforms other approaches in terms of reward for different settings of w_M s.

Note that offline RL is conducted before each setting of w_M when the RL approach is applied. In order to show that the RL approach can learn a good solution to adapt the environment, we compare throughput and *ADUP*, the average distance per unit product, for different w_M s. According to

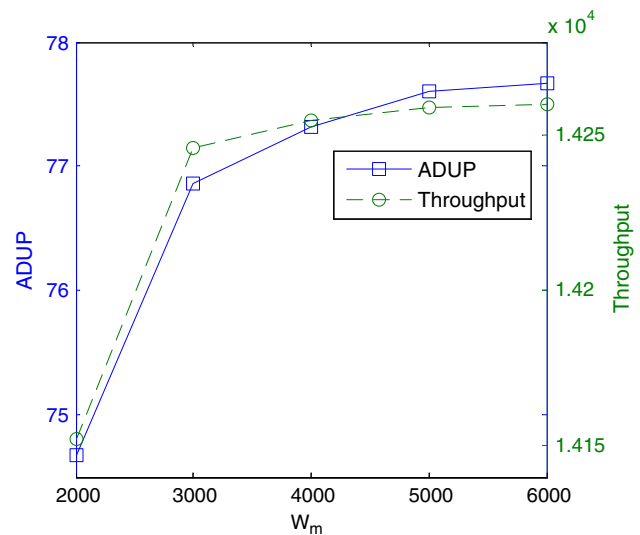


Fig. 3 Average distance per unit product and throughput under different settings of w_M

Eq. (26), a larger *ADUP* means more cost for a unit product assembled. And smaller *ADUP* indicates that less cost is required to assemble a unit product.

$$ADUP = D/M \tag{26}$$

Since w_M is the coefficient of throughput, when it increases, the importance of throughput also increases according to Eq. (1). And the importance of distance decreases relatively. In such cases, the GA line should focus more on throughput even though it has to pay more cost in material handling. As indicated by Fig. 3, both throughput and *ADUP* increase at the same time when w_M increases. It means that through RL, the RL approach adjusts its policy to achieve higher throughput with the price of higher unit cost when w_M increases.

We can also compare the RL approach with the hypothetical approach for different w_M s. The rewards of the hypothetical approach for the four different w_M values are 2.51×10^7 , 3.98×10^7 , 5.46×10^7 and 8.41×10^7 . Thus the gaps between the RL approach and the hypothetical approach are 8.0, 6.0, 5.5 and 4.7 %, respectively.

From the above simulation results, we can see that the proposed RL approach outperforms other approaches in terms of reward in all of the experiments. More importantly, Fig. 3 indicates that the RL approach can change its policy to adapt the environment through RL. By doing this, the RL approach performs well in all of the experiments. As indicated by the results of comparing the RL approach and the hypothetical approach, the performance of the RL approach is close to that of the hypothetical approach. Because it's almost impossible to find a realistic approach that can perform as well as the hypothetical approach does, the RL approach is suitable for

the multiple-load carrier scheduling problem studied in this paper.

Conclusion

In this paper we investigate the problem of scheduling a dolly train, a multiple-load carrier in a GA line. The objective of the scheduling problem is to maximize the reward of the GA line. Thus we consider two scheduling criteria, the throughput of the GA line and the material handling distance. Based on look-ahead information, we propose a look-ahead based material handling request generating policy. We also develop a heuristic dispatching algorithm to pick up loads and to determine the order of delivery. After modeling the scheduling problem as a RL problem with continuous time and space, we propose a $Q(\lambda)$ RL algorithm to solve the RL problem. In the RL algorithm, a linear function with a gradient-descent method is used to approximate Q-value functions.

To evaluate the performance of the proposed approach, we compare the performance of the proposed RL approach with that of other approaches. The results show that the proposed approach outperforms other approaches with respect to the reward of the GA line. Therefore, the proposed approach is an appropriate real-time multiple-load carrier scheduling approach in a dynamic manufacturing environment.

In this paper, we use a look-ahead based request generating policy other than the traditional ROP policy. However, because the scheduling system can always dispatch the dolly train to deliver parts regardless of the status of line-side buffers, it's not necessary to have a request generating policy in the GA line theoretically. As a matter of fact, a scheduling system without a request generating policy is identical to a scheduling system in which reorder points are set to some sufficiently large number. Therefore, it's very interesting to develop approaches without request generating policies. Furthermore, only three types of actions are defined in our RL formulation despite that there are a lot of possible actions can be defined. It's very important to discuss this issue in the future in order to improve the performance of the scheduling approach.

Acknowledgments The research is supported by the National Science Foundation of China under Grants No. 51075277, No. 51275558 and No. 61273035.

References

- Belmecheri, F., Prins, C., Yalaoui, F., & Amodeo, L. (2013). Particle swarm optimization algorithm for a vehicle routing problem with heterogeneous fleet, mixed backhauls, and time windows. *Journal of Intelligent Manufacturing*, 24(4), 775–789. doi:10.1007/s10845-012-0627-8.
- Berman, S., Schechtman, E., & Edan, Y. (2009). Evaluation of automatic guided vehicle systems. *Robotics and Computer-Integrated Manufacturing*, 25(3), 522–528. doi:10.1016/j.rcim.2008.02.009.
- Chen, C., Xi, L., Zhou, B., & Zhou, S. (2011). A multiple-criteria real-time scheduling approach for multiple-load carriers subject to LIFO-loading constraints. *International Journal of Production Research*, 49(16), 4787–4806. doi:10.1080/00207543.2010.510486.
- Chen, C., Zhou, B., & Xi, L. (2010). A support vector machine based scheduling approach for a material handling system. In: Presented at the natural computation (ICNC), 2010 sixth international conference on (Vol. 7, pp. 3768–3772).
- Dang, Q.-V., Nielsen, I., Steger-Jensen, K., & Madsen, O. (2013). Scheduling a single mobile robot for part-feeding tasks of production lines. *Journal of Intelligent Manufacturing*. doi:10.1007/s10845-013-0729-y.
- de Koster, R.(M.) B. M., Le-Anh, T., & van der Meer, J. R. (2004). Testing and classifying vehicle dispatching rules in three real-world settings. *Journal of Operations Management*, 22(4), 369–386. doi:10.1016/j.jom.2004.05.006.
- Duda, R. O., & Hart, P. E. (1973). *Pattern classification and scene analysis*. New York: Wiley.
- Fowler, J. W., Hogg, G. L., & Philips, D. T. (1992). Control of multi-product bulk service diffusion/oxidation processes. *IIE Transactions*, 24(4), 84–96. doi:10.1080/07408179208964236.
- Gabel, T., & Riedmiller, M. (2011). Distributed policy search reinforcement learning for job-shop scheduling tasks. *International Journal of Production Research*, 50(1), 41–61. doi:10.1080/00207543.2011.571443.
- Grunow, M., Günther, H.-O., & Lehmann, M. (2004). Dispatching multi-load AGVs in highly automated seaport container terminals. *OR Spectrum*, 26(2), 211–235. doi:10.1007/s00291-003-0147-1.
- Ho, Y.-C., Liu, H.-C., & Yih, Y. (2012). A multiple-attribute method for concurrently solving the pickup-dispatching problem and the load-selection problem of multiple-load AGVs. *Journal of Manufacturing Systems*, 31(3), 288–300. doi:10.1016/j.jmsy.2012.03.002.
- Jeon, S., Kim, K., & Kopfer, H. (2011). Routing automated guided vehicles in container terminals through the Q-learning technique. *Logistics Research*, 3(1), 19–27. doi:10.1007/s12159-010-0042-5.
- Joe, Y. Y., Gan, O. P., & Lewis, F. L. (2012). Multi-commodity flow dynamic resource assignment and matrix-based job dispatching for multi-relay transfer in complex material handling systems (MHS). *Journal of Intelligent Manufacturing*, 1–17. doi:10.1007/s10845-012-0713-y.
- Kim, D. B., & Hwang, H. (2001). A dispatching algorithm for multiple-load AGVS using a fuzzy decision-making method in a job shop environment. *Engineering Optimization*, 33(5), 523–547. doi:10.1080/03052150108940932.
- Le-Anh, T., & De Koster, M. B. M. (2006). A review of design and control of automated guided vehicle systems. *European Journal of Operational Research*, 171(1), 1–23. doi:10.1016/j.ejor.2005.01.036.
- Le-Anh, T., de Koster, R. B. M., & Yu, Y. (2010). Performance evaluation of dynamic scheduling approaches in vehicle-based internal transport systems. *International Journal of Production Research*, 48(24), 7219–7242. doi:10.1080/00207540903443279.
- Li, X., Tao Geng, YuPu Yang, & Xiaoming Xu. (2002). Multiagent AGVs dispatching system using multilevel decisions method. In Presented at the American control conference, 2002. Proceedings of the 2002, IEEE (Vol. 2, pp. 1135–1136 vol. 2). doi:10.1109/ACC.2002.1023172.
- Min, H.-S., & Yih, Y. (2003). Selection of dispatching rules on multiple dispatching decision points in real-time scheduling of a semiconductor wafer fabrication system. *International Journal of Production Research*, 41(16), 3921–3941.
- Montazeri, M., & Van Wassenhove, L. N. (1990). Analysis of scheduling rules for an FMS. *International journal of production research*, 28(4), 785.

- Nayyar, P., & Khator, S. K. (1993). Operational control of multi-load vehicles in an automated guided vehicle system. In Proceedings of the 15th annual conference on computers and industrial engineering (pp. 503–506). Blacksburg, Virginia, United States: Pergamon Press, Inc., Retrieved from <http://portal.acm.org/citation.cfm?id=186340>
- Neuts, M. F. (1967). A general class of bulk queues with Poisson input. *The Annals of Mathematical Statistics*, 38(3), 759–770.
- Occena, L. G., & Yokota, T. (1993). Analysis of the AGV loading capacity in a JIT environment. *Journal of Manufacturing Systems*, 12(1), 24.
- Orides, M., Castro, P. A. D., Kato, E. R. R., & Camargo, H. A. (2006). A genetic fuzzy system for defining a reactive dispatching rule for AGVs. In Systems, Man and Cybernetics, 2006. SMC '06. IEEE international conference on (Vol. 1, pp. 56–61). doi:10.1109/ICSMC.2006.384358.
- Ozden, M. (1988). A simulation study of multiple-load-carrying automated guided vehicles in a flexible manufacturing system. *International Journal of Production Research*, 26(8), 1353–1366. doi:10.1080/00207548808947950.
- Peng, J., & Williams, R. J. (1996). Incremental multi-step Q-learning. *Machine Learning*, 22(1–3), 283–290. doi:10.1023/A:1018076709321.
- Potvin, J.-Y., Shen, Y., & Rousseau, J.-M. (1992). Neural networks for automated vehicle dispatching. *Computers & Operations Research*, 19(3–4), 267–276. doi:10.1016/0305-0548(92)90048-A.
- Sarin, S. C., Varadarajan, A., & Wang, L. (2010). A survey of dispatching rules for operational control in wafer fabrication. *Production Planning & Control*, 22(1), 4–24. doi:10.1080/09537287.2010.490014.
- Sinriech, D., & Kotlarski, J. (2002). A dynamic scheduling algorithm for a multiple-load multiple-carrier system. *International Journal of Production Research*, 40(5), 1065–1080. doi:10.1080/00207540110105662.
- Vahdani, B., Tavakkoli-Moghaddam, R., Zandieh, M., & Razmi, J. (2012). Vehicle routing scheduling using an enhanced hybrid optimization approach. *Journal of Intelligent Manufacturing*, 23(3), 759–774. doi:10.1007/s10845-010-0427-y.
- Vis, I. F. A. (2006). Survey of research in the design and control of automated guided vehicle systems. *European Journal of Operational Research*, 170(3), 677–709. doi:10.1016/j.ejor.2004.09.020.
- Wang, C.-N., & Chen, L.-C. (2012). The heuristic preemptive dispatching method of material transportation system in 300 mm semiconductor fabrication. *Journal of Intelligent Manufacturing*, 23(5), 2047–2056. doi:10.1007/s10845-011-0531-7.
- Wang, Y. C., & Usher, J. M. (2005). Application of reinforcement learning for agent-based production scheduling. *Engineering Applications of Artificial Intelligence*, 18(1), 73–82. doi:10.1016/j.engappai.2004.08.018.
- Weng, W. W., & Leachman, R. C. (1993). An improved methodology for real-time production decisions at batch-process work stations. *IEEE Transactions on Semiconductor Manufacturing*, 6(3), 219–225. doi:10.1109/66.238169.
- Zhang, Z., Zheng, L., Hou, F., & Li, N. (2011). Semiconductor final test scheduling with Sarsa(λ , k) algorithm. *European Journal of Operational Research*, 215(2), 446–458. doi:10.1016/j.ejor.2011.05.052.
- Zhang, Z., Zheng, L., Li, N., Wang, W., Zhong, S., & Hu, K. (2012). Minimizing mean weighted tardiness in unrelated parallel machine scheduling with reinforcement learning. *Computers & Operations Research*, 39(7), 1315–1324. doi:10.1016/j.cor.2011.07.019.