Int J Adv Manuf Technol (2009) 45:603–615
DOI 10.1007/s00170-009-1996-9

ORIGINAL ARTICLE

Deadlock-free multi-attribute dispatching method for AGV systems

Xianping Guan · Xianzhong Dai

Received: 24 September 2008 / Accepted: 25 February 2009 / Published online: 17 March 2009
© Springer-Verlag London Limited 2009

Abstract This paper aims at developing a flexible, efficient, and deadlock-free dispatching method for automated guided vehicle systems. For this purpose, a deadlock-free multi-attribute dispatching method with dynamically adjustable weights (AWMA) is proposed. Traveling distance, input, and output buffer statuses are selected as dispatching attributes according to the efficiency and deadlock avoidance requirement. The weight for each attribute is dynamically adjusted according to the processing load and transportation load of the system. To ensure the system to be deadlock-free, a deadlock avoidance policy based on remaining capacity concept is introduced. It works by temporarily forbidding critical tasks according to the system state, which will otherwise cause system deadlock. The AWMA method is formed by integrating the deadlock avoidance policy into the multi-attribute dispatching procedure. To validate the effectiveness of the proposed method, several simulation experiments were carried out to compare three commonly used dispatching methods with the proposed one under different system settings. The simulation results indicate that the deadlock avoidance policy can guarantee the system to be deadlock-free and that the proposed method is efficient.

Keywords Automated guided vehicles · Dispatching · Multi-attribute · Deadlock avoidance · Remaining capacity

X. Guan (✉) · X. Dai
Key Laboratory of Measurement and Control of CSE
(School of Automation, Southeast University),
Ministry of Education, Southeast University,
Nanjing 210096, China
e-mail: gxp334@126.com

X. Dai
e-mail: xzdai@seu.edu

1 Introduction

Market demands are becoming highly specific and rapidly changing nowadays. To stay competitive, manufacturers are required to accommodate various products in a timely and cost-effective manner. A new manufacturing paradigm, reconfigurable manufacturing system (RMS), has been proposed to meet this requirement. To cope with fluctuant production, RMS undergoes frequent configuration changes and supervisory controller reconfigurations [1–3]. This leads to changes of material flow patterns as well and thus calls for flexible and efficient material handling system. Automated guided vehicle system (AGVS) is a kind of flexible and efficient material handling system with intelligent control capability, which satisfies the material handling requirement of RMS. As dispatching is one of the most important issues for AGVS management and control [4–6], this paper concentrates on dispatching method for AGVS under RMS job shop environment.

Dispatching methods take decisions on task allocations in real time. To make dispatching decision, the decision parameters should be specified. These parameters are often called attributes in the literature. Egbelu and Tanchoco [7] proposed several single attribute dispatching rules. As single attribute only reflects one aspect of the system, it cannot efficiently adapt to various system settings. To synthesize multiple aspects of the system, multi-attribute methods have been adopted [8–12]. Jeong and Randhawa [8] proposed several multi-attribute dispatching rules using three attributes: vehicle empty travel distance, remaining spaces in input buffers, and remaining spaces in outgoing buffers. When multi-attribute rules are used, it is difficult to specify the weights for the attributes. In [8], neural network learning method was used to decide the weights. Naso and Turchiano [9] utilized simulation and genetic algorithm to learn the weight of each attribute. These learning approaches have to be done off-line, so they are not responsive to requirement variety and system resource changes. On the other hand, suitable training data are difficult to acquire. Bilge et al. [10] used output buffer length and travel time to pick up as two attributes of dispatching rule, and the loads of the processing and transportation subsystem were used to adjust the weight of each attribute. However, the load of individual equipment is not different and deadlock is not considered. Guan and Dai [11] proposed a multi-attribute dispatching method with dynamically adjustable weights. Although this method is flexible and efficient, it does not ensure the system to be deadlock-free.

Block and deadlock may occur if the dispatching rules are not properly designed. In [6], several block and deadlock situations were listed. Liu and Hung [13] listed the four conditions for deadlock to occur, and they recognized two kinds of deadlock in AGVS: (1) deadlock caused by sharing guide path segments and (2) deadlock caused by finite queuing capacity. The first kind is called routing deadlock and the second dispatching deadlock in this paper. There are three commonly used deadlock handling methods: (1) prevention; (2) avoidance; and (3) detection and resolution. While handling deadlock, it is difficult to ensure the system to be deadlock-free and to keep the dispatching flexible and efficient at the same time. Generally, deadlock avoidance is more flexible than deadlock prevention and more efficient than deadlock detection and resolution. Therefore, deadlock avoidance method is adopted in this paper. Zone control methods are commonly used to avoid deadlock in automated guided vehicle (AGV) routing [14–19]. Yoo et al. [14] proposed graph theory-based method for deadlock avoidance of AGVS, but this method only dealt with AGVS routing deadlock and the given algorithm can only resolve deadlock caused by two AGVs. Srivastava et al. [15] developed an intelligent agent-based AGV controller for a flexible manufacturing system, and zone control method was employed to handle deadlock and collision of AGVS. Liu and Hung [13] proposed a deadlock-free dispatching method for multi-load AGV systems. However, this method does not consider the distance aspect and only suits for layout with a main loop. Moreover, this method does not ensure the system to be deadlock-free. Wu and Zhou [20] used colored resource-oriented Petri net to model both the AGV system and part processing system, which were integrated using macro transitions. Upon this integrated model, they proposed a maximally permissive deadlock avoidance policy. As analyzed in [14], Petri net modeling has the disadvantages of complexity and difficulty to adapt to system changes. Lehmann et al. [21] adopted resource assignment matrix to predict deadlock and avoided deadlock by adjusting assignment of some entities. When buffer capacity is considered, the assignment matrix is difficult to use. In a word, there is a lack of flexible, efficient, and deadlock-free dispatching method for AGVS.

To ensure the system to be deadlock-free and to allow the maximum flexibility of dispatching, a deadlock avoidance policy based on remaining capacity concept is proposed in this paper. This deadlock avoidance policy is integrated into the multi-attribute dispatching method proposed in [11]. Thus, the deadlock-free multi-attribute dispatching method with dynamic adjustable weights is formed. The method is expected to be flexible, efficient, and deadlock-free. To verify the expectation, a simulation system is designed and several experiments are carried out to compare three commonly used methods with the method proposed in this paper under various conditions.

This paper is organized as follows. Problem statement is given in Section 2 and the deadlock-free multi-attribute dispatching method with dynamically adjustable weights (AWMA) is described in the next section. In Section 4, several experiments are carried out to compare three commonly used dispatching methods with the one proposed in this paper under various system settings. Section 5 concludes this paper.

2 Problem statement

2.1 Description of the target system

Manufacturing system using AGVS as the transportation subsystem is generally composed of a processing subsystem and a transportation subsystem. The processing subsystem includes processing machines, input, and output buffers. The transportation subsystem includes several AGVs and a guide path network. As there are many kinds of such systems, this paper focuses on RMS systems under job shop environment. First, some assumptions are given for the target system:

1. Jobs are randomly generated according to the job mix and arrive according to Poisson process; each job enters the system through input workstation and leaves the system through output workstation after completing all operations.
2. Jobs in input buffers are input to processing machines according first come first input rule, and jobs in output buffers can be taken at any order as required.
3. A processing machine processes one job at a time, and processing operations cannot be interrupted.
4. The guide path is unidirectional.
5. One AGV carries one job at a time, and it travels alone the shortest path between pickup and drop down positions at constant speed.

6. Equipments failures are not considered.
7. Loading and unloading time for each job are the same for every workstation; when there is one AGV which is loading or unloading at a workstation, other AGVs with the same operation aim (loading or unloading) have to wait.
8. An AGV will stay at where it has finished its task if no more tasks are assigned to it, and it does not influence the travel of other AGVs.
9. Battery charge is not considered.
10. Change time for processing different jobs is assumed to be zero.

An example system is shown in Fig. 1. Generally, suppose there are N_M+2 workstations, WS, in the system, among which there is one input workstation 0 and one output workstation N_M+1; others are processing workstation. Each workstation m has one input buffer with capacity $C_I(m)$, one output buffer with capacity $C_O(m)$. Each processing workstation has a processing machine. The input buffer of the input workstation and the output buffer of the output workstation are not considered. There is one guide path network $G_g=(V_g,E_g)$ in the system. The node set V_g includes pickup nodes $P_m \in P$, drop down nodes $d_m \in D$, and other connecting nodes. The segments $e \in E_g$ on the network are unidirectional. For a given guide path network, the minimum distance from nodes i to node j is DS(i,j), which is obtained using Dijkstra algorithm. If the speed of AGV is v, then the travel time from node i to node j is $Tv(i,j)$=DS(i,j)/v. Suppose there are N_R AGVs in the system. AGVs travel on the segment of network following the direction shown by the arrows in the figure. The jobs arrive at the system through input workstation. And after completing required number of operations, they leave the system through output workstation. During a certain time period length, WT, there are N_J jobs waiting for processing. During a job j staying in the system, it needs transporting by AGV and processing by machine alternatively. After a sequence of such transporting task, T_{jk}, and processing task, O_{jk}, it leaves the system. Different jobs may have different task sequences, as shown in Fig. 2. For each transporting or processing task, one entity should be assigned to carry out the task. As it is assumed that the processing task has been assigned, this paper concentrated on the assignment of transporting task. The dispatching procedure can be utilized to accomplish this task. The dispatching procedure is discussed in more detail in the next sections.

2.2 Dispatching for AGV systems

To make dispatching decision, several issues must be considered.

1. When to make dispatching decision?

There are two events which cause the call of dispatching procedure: (1) the arrival of new transporting task and (2) an AGV has completed its current task. When a job arrives, or when a job completes its current processing operation, it needs to be transported to the next workstation for further processing or to leave the system. When an AGV unloads its current job, it is free to carry out another task. If one of the events mentioned above occurs, the dispatching procedure should be called.

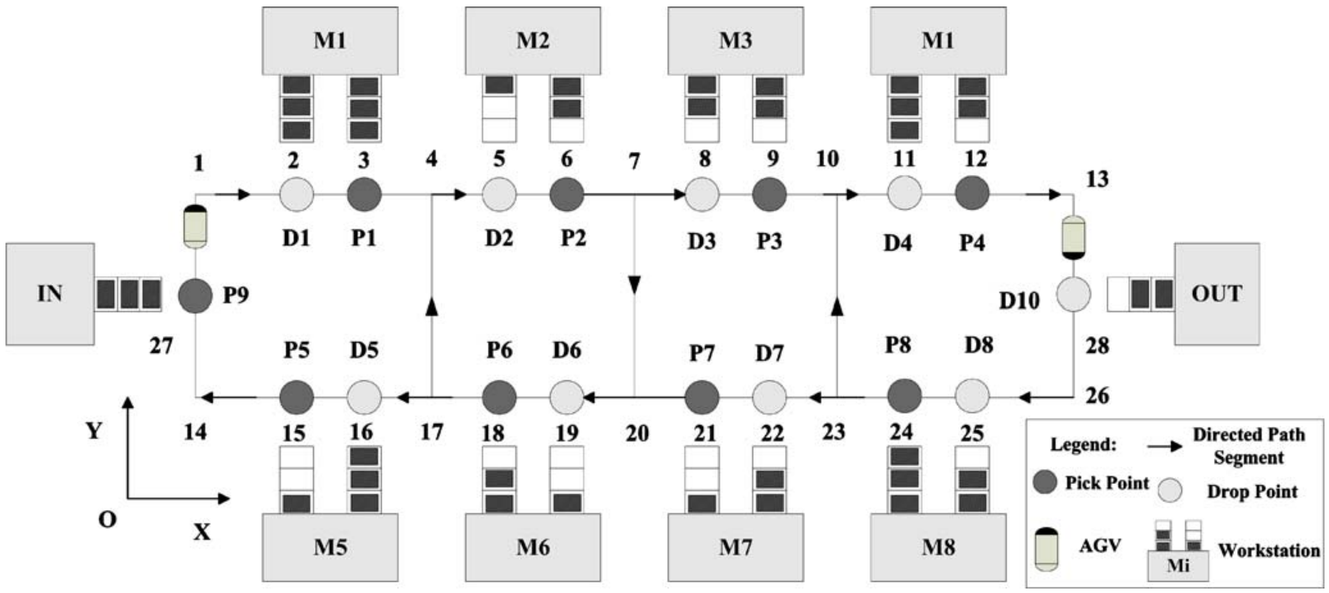

Fig. 1 The layout of the system

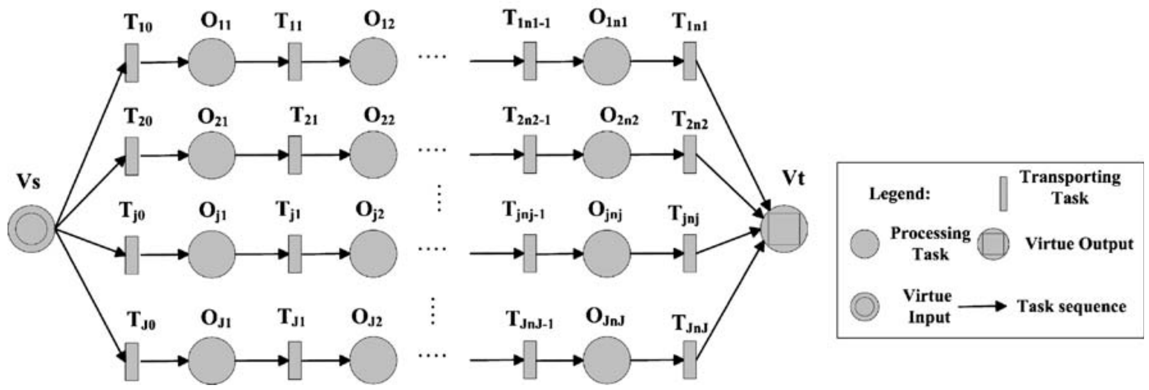

Fig. 2 The task sequences of the system

2. Which tasks and vehicles are involved?

Jobs that arrive at the input workstation and the ones that have finished their current processing operations and have been output to output buffers are considered as tasks waiting for dispatching. Idle AGVs are considered ready for receiving tasks.

3. How to decide task assignment?

For this issue, some values between tasks and AGVs should be computed. Such values are called utility values. This is critical for decision. Multi-attribute rule is used in this paper. After utility values calculation, task assignment decision should be made. Because deadlock will severely deteriorate the system performance, deadlock handling policy in required. So the general dispatching procedure is given in Fig. 3. The main models in this procedure, that is, utility value matrix calculation and task assignment, are given in the next section in more detail.

3 Deadlock-free multi-attribute dispatching method

The aim of this paper was to develop a flexible, efficient, and deadlock-free dispatching method for AGVS. Hence, a deadlock-free AWMA method is proposed. First, the utility values between tasks and AGVs are calculated. Multi-attribute dispatching rule is adopted to synthesize multiple aspects of the system. Traveling distance, input, and output buffer statuses are chosen as dispatching attributes. The weights of the attributes are dynamically adjusted according to the processing load and the transportation load of the system. Thus, the method can adapt to different system settings and task loads. Then, the deadlock-free dispatching procedure is given. To handle the deadlock situation, the remaining capacity concept is proposed. The deadlock avoidance policy works by temporarily forbidding those critical tasks that may lead to system deadlock. In addition, the deadlock-free property of this policy is proven.

3.1 Multi-attribute utility values calculation

To decide the utility values between tasks and AGVs, the attributes for dispatching must be chosen. As different attributes reflect different system conditions, combination of several attributes may give more comprehension of the system. Traveling distance, input, and output buffer statuses are chosen as dispatching attributes. To decide the weights of the attributes, system load condition is taken into account. As distance reflects transportation load, and input and output buffer statuses reflect processing load, so the

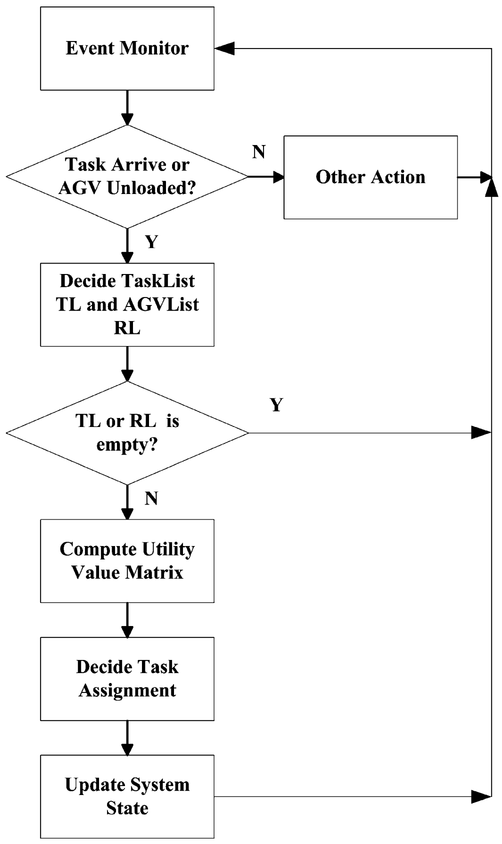
Event Monitor
Task Arrive or AGV Unloaded?
N
Other Action
Y
Decide TaskList TL and AGVList RL
TL or RL is empty?
Y
N
Compute Utility Value Matrix
Decide Task Assignment
Update System State

Fig. 3 The dispatching procedure

weight of distance is mainly influenced by transportation load, and the weights of input and output buffer statuses are mainly determined by processing load. The utility value between a task i and an AGV j can be calculated formally as:

$$f(i,j) = W_D f_D + W_I f_I + W_O f_O \tag{1}$$

where f_D, f_I, f_O are the values for the three attributes and W_D, W_I, W_O are weights for the attributes. The parameters of this formula need to be calculated. The value of each attribute is calculated according to the status of the related entity and the weight of each attribute according to the system load and load of each entity.

To make decision, some parameters of the system are computed.

The total processing load is:

$$\mathrm{NP} = \sum_{j=1}^{N_J} \sum_{k=1}^{n_j} \lambda q_j P_{jk} \mathrm{WT} \tag{2}$$

where λ is the job arrive rate, q_j is the mix of job type j, P_{jk} is the processing time of operation O_{jk}, and WT is the planned run time length.

The total transportation load is:

$$\mathrm{NT} = K_R \left(\sum_{j=1}^{N_J} \sum_{k=0}^{n_j} \lambda q_j (t_{jk} + t_l + t_u) \mathrm{WT} \right) \tag{3}$$

where K_R is AGV travel factor, which reflects AGV's loaded and empty travel rate, t_{jk} is the time needed for transporting task T_{jk}, with $t_{jk} = Tv(p_{Mjk}, d_{Mj(k+1)})$, t_l is loading time for a task, t_u is unloading time, N_J is the number of jobs to the system, n_j is the number of operations of job j.

The total processing ability of the system is:

$$\mathrm{CM} = N_M \times \mathrm{WT} \tag{4}$$

where N_M is the number of processing machines in the system.

The total transportation ability of the system is:

$$\mathrm{CV} = N_R \times \mathrm{WT} \tag{5}$$

where N_R is the number of AGVs in the system.

So the processing load factor of the system is:

$$L_p = \mathrm{NP}/\mathrm{CM} = \sum_{j=1}^{N_J} \sum_{k=1}^{n_j} \lambda q_j P_{jk} / N_M, \tag{6}$$

and the transportation load factor of the system is:

$$L_t = \mathrm{NT}/\mathrm{CV} = K_R \sum_{j=1}^{N_J} \sum_{k=0}^{n_j} \lambda q_j (t_{jk} + t_l + t_u) / N_R. \tag{7}$$

The processing load for each workstation can be computed as follows:

$$l_m = \sum_{j=1}^{N_J} \sum_{k=1}^{n_j} \lambda q_j P_{jk} X_{jkm} \tag{8}$$

with $X_{jkm} = \begin{cases} 1, & \text{if}(M_{jk} = m) \\ 0, & \text{otherwise} \end{cases}$.

So the load factor for each workstation can be defined as:

$$K_m = \frac{l_m}{L_P}. \tag{9}$$

The load factors of input and output workstations are defined as 1.

The transporting tasks can be denoted as $T_{jk}=(M_p,M_d)$, with M_p as the source workstation and M_d as the destination workstation. The attribute value for input buffer for workstation m can be given as:

$$f_{in}(m) = \frac{c_I(m)}{C_I(m)} \tag{10}$$

where $C_I(m)$ is the input buffer capacity of the workstation m and $c_I(m)$ is the current number of jobs in the input buffer of workstation m.

The attribute value for output buffer is given as:

$$f_{out}(m) = \frac{c_O(m)}{C_O(m)} \tag{11}$$

where $C_O(m)$ is the output buffer capacity of the workstation m and $c_O(m)$ is the current number of jobs in the output buffer of workstation m.

To avoid deadlock situations, emphasis is given to the full output buffers and empty input buffers, so the attribute value for input buffer is adjusted as:

$$f_I = \begin{cases} F_i(1 - f_{in}(M_d)), & \text{if } c_I(M_d) = 0 \\ 1 - f_{in}(M_d), & \text{others} \end{cases} \tag{12}$$

and the one for output buffer as:

$$f_O = \begin{cases} F_o(f_{out}(M_p)), & \text{if } c_O(M_p) = C_O(M_p) \\ f_{out}(M_p), & \text{others} \end{cases} \tag{13}$$

with F_i, F_o as constants greater than 1 and is often set $F_o > F_i$. The weight for each attribute should be adaptive to the system load to make load balance for the system resources. So the distance weight is set as $W'_D=L_t$, the input buffer weight as $W'_I=L_p$, and the output buffer weight as $W'_O=L_p$. These weights can be normalized as:

$$W_D = \frac{W'_D}{W'_D + W'_I + W'_O} \tag{14}$$

$$W_{\mathrm{I}} = \frac{W'_{\mathrm{I}}}{W'_{\mathrm{D}} + W'_{\mathrm{I}} + W'_{\mathrm{O}}} \tag{15}$$

$$W_{\mathrm{O}} = \frac{W'_{\mathrm{O}}}{W'_{\mathrm{D}} + W'_{\mathrm{I}} + W'_{\mathrm{O}}}. \tag{16}$$

So the value for task T_{jk} can be given as:

$$f_{jk} = W_{\mathrm{I}} f_{\mathrm{I}} K_{M_{\mathrm{d}}} + W_{\mathrm{O}} f_{\mathrm{O}} K_{M_{\mathrm{p}}} \tag{17}$$

where $K_{M\mathrm{d}}$ and $K_{M\mathrm{P}}$ are calculated according to Eq. 9.

For task T_{jk} to AGV R located at position P_{R}, the distance value is:

$$f_{\mathrm{D}} = \frac{\mathrm{MD} - Tv(p_{\mathrm{R}}, p_{M_{\mathrm{p}}})}{\mathrm{MD}} \tag{18}$$

where MD=max($Tv(i,j)$) is the maximum travel time between nodes in the path network. So the utility value between task T_{jk} and AGV R is:

$$f\left(T_{jk}, R\right) = f_{jk} + W_{\mathrm{D}} f_{\mathrm{D}}. \tag{19}$$

3.2 Deadlock-free multi-attribute dispatching procedure

To make deadlock-free dispatching, several definitions are given.

Number of input AGVs, $L_{\mathrm{I}}(m)$

The number of input AGVs of a workstation m is defined as the number of AGVs that have been assigned tasks, and the destination workstation of the tasks is m.

Number of output AGVs, $L_{\mathrm{O}}(m)$

The number of output AGVs of a workstation m is defined as the number of AGVs that have been assigned tasks, and the source workstation of the tasks is m.

Remaining capacity, RM(m)

The remaining capacity of workstation m is defined as the number of AGVs that workstation m can accommodate additionally, which can be calculated as:

$$\begin{aligned}\mathrm{RM}(m) = C_{\mathrm{I}}(m) + C_{\mathrm{O}}(m) + 1 - c_{\mathrm{I}}(m) - c_{\mathrm{O}}(m)\\ - pr(m) - L_{\mathrm{I}}(m) + L_{\mathrm{O}}(m)\end{aligned} \tag{20}$$

with $pr(m)$=1 if there is one job on the processing machine of workstation m, otherwise $pr(m)$=0.

Blocked AGV

An AGV is called blocked if the AGV has accepted a task and the remaining capacity of the destination workstation of the task is non-positive.

Active AGV

An AGV is called active if it is not blocked.

Number of blocked AGVs, N_{B}

The number of blocked AGVs in the system.

As Moorthy stated, "Deadlock in a broad sense is a situation in which at least a part of the system stalls" [19], so system deadlock can be given as follows:

System deadlock

When all entities in the system stall and cannot progress with their process, the system is in deadlock.

Deadlock is a definite state. On the other hand, only under certain constraints, the states that will inevitably lead to deadlock can be confirmed. In Section 2, several assumptions of the system have been given. Here, further assumptions about dispatching conditions are listed: Only idle AGVs can receive tasks; once tasks are assigned, they cannot be cancelled or reassigned; path block or deadlock is not considered. Then, the deadlock discussion in this paper is under all above conditions.

Under such conditions, once task assignment is made, the next system state is definite. Whether the system gets into deadlock or not can be decided after a task assignment and cannot change if there is no further task assignment. That is, only task assignment can change the state of deadlock. Thus, the deadlock state can be extended to the whole time length after the previous task assignment and before the next task assignment. Thus, different states can be divided into different task assignment states.

Extended system state, q

System state during the whole time length between two consequence task assignments is regarded as an extended system state.

So a system state is called deadlocked if the system gets into deadlock at the end of the extended system state. If deadlocked, then no further task assignment can be made. Hereafter, when system state is discussed, it refers to extended system state. Under certain system state, the number of blocked AGVs will not change. Only task assignment can change system state and, thus, the number of blocked AGVs in the system.

According to the definitions, it is easy to get the following lemma.

Lemma 1

The system gets into deadlock if and only if N_{B}=N_{R}.

Proof 1

As the AGVs are the only active part in the system, so if all the AGVs stall, then no job is able to progress with its tasks. The whole system will eventually stall. According to the definition of blocked AGV, a blocked AGV cannot unload its accepted job and it needs other active AGVs to activate it; otherwise, it remains blocked. So if N_{B}=N_R, then no AGV can get active and no further task assignment can be made, so eventually, all the AGVs will stall. Then, the system gets into deadlock.

Proof 2

If the system is in deadlock, then all the AGVs must be blocked. Otherwise, if there is at least one active AGV, it

Table 1 The buffer capacity of each workstation

Workstation number	Input buffer capacity	Output buffer capacity
WS1	3	3
WS2	3	3
WS3	3	4
WS4	4	3
WS5	3	3
WS6	4	4
WS7	3	4
WS8	4	3

will eventually turn idle and can receive other task. So the system is not deadlocked under this state; this contradicts with the condition that the system is in deadlock. So if the system is deadlocked, $N_B=N_R$.■

According to Lemma 1, the following definition is given.

Deadlock of AGV dispatching

The deadlock of AGV dispatching is defined as the situation when all AGVs are blocked and no further task assignment can be made from then on.

To ensure the system to be deadlock-free, deadlock avoidance policy is required.

Suppose the task list waiting for assignment is TL={1,2,…,N_T} and the available AGV list is RL={1,2,…,N_V}, A is the assignment matrix, with each element as:

$$a(i,j)=\begin{cases}1, & \text{if task } T_i \text{ is assigned to AGV } R_j\\ 0, & \text{others}\end{cases}, T_i\in \text{TL},\ R_j\in \text{RL}. \qquad (21)$$

Initially, every element in matrix A is set to 0. Then, according to Eq. 19, the utility value for each task–AGV pair $f(i,j)$ is calculated.

So to avoid deadlock, what is only needed is to make sure there is at least one active AGV in the system. When $N_B=N_R-1$, the dispatching procedure should make sure that the only active AGV will not get blocked, or if it gets blocked, it can make another blocked AGV active. Then, the deadlock avoidance policy can be formed. If the number of blocked AGVs $N_B=N_R-1$, some tasks should temporarily be forbidden; otherwise, deadlock may occur.

So when $N_B=N_R-1$, the forbidden task set is defined as:

$$\text{FL}=\{T_i|L_I(T_i(p))=0,\text{RM}(T_i(d))\le 0,T_i\in \text{TL}\}.$$

Then the available task set is given as: AL=TL\FL.

Task–AGV pair with max utility value is selected and fixed:

$$(k,r)=Arg\max_{i,j}f(i,j),T_i\in \text{AL},R_j\in \text{RL}. \qquad (22)$$

Then set $a(k,r)=1$, and TL=TL\{T_k}, RL=RL\{R_r}, then the available task set AL can be updated. If both AL and RL are not empty, the procedure is repeated. The dispatching procedure with deadlock avoidance policy is given as follows.

Dispatching procedure with deadlock avoidance policy:

```
Input: TL,RL,$N_B$,RM,$L_O$,$L_I$;
Output: $A$, updated TL,RL,$N_B$,RM,$L_O$,$L_I$;
Step 1:  Calculate utility value matrix $f$ using Eq. 19.
Set $a(i,j)=0$ for every element of $A$.
Step 2:
    if  $N_B=N_R-1$
        $FL=\{T_i|L_I(T_i(p))=0,RM(T_i(d))\le 0,T_i\in TL\}$
        $AL=TL\backslash FL$
    else
        AL=TL
    endif
Step 3:
    while  $AL\ne\Phi \& RL\ne\Phi$
        $(k,r)=\text{Arg}\max_{i,j}f(i,j),T_i\in \text{AL},R_j\in \text{RL}$
        $a(k,r)=1$
        $L_O(T_k(p))=L_O(T_k(p))+1$
      if  $\text{RM}(T_k(p))<0$
          $N_B=N_B-1$
```

Table 2 The process information of job set 1

Job type	Operation sequence and process time (s)	Mix (%)
A	1(180)5(120)4(240)6(180) 2(240)7(180)	20
B	7(180)5(240)4(120)8(180)6(180) 2(120)	20
C	3(120)2(240)5(180)6(120)8(120)	20
D	4(180)3(240)8(120)5(120) 1(180)	20
E	3(180)1(240)5(120) 6(240)7(180)	20

Table 3 The process information of job set 2

Job type	Operation sequence and process time (s)	Mix (%)
F	1(270)5(180)4(360)6(270) 2(360)7(270)	20
G	7(270)5(360)4(180)8(270)6(270) 2(180)	20
H	3(180)2(360)5(270)6(180)8(180)	20
I	4(270)3(360)8(180)5(180) 1(270)	20
J	3(270)1(360)5(180) 6(360)7(270)	20

```
        endif
          $\mathrm{RM}(T_k(p)) = \mathrm{RM}(T_k(p)) + 1$
          $L_1(T_k(d)) = L_1(T_k(d)) + 1$
          $\mathrm{RM}(T_k(d)) = \mathrm{RM}(T_k(d)) - 1$
          if  $\mathrm{RM}(T_k(d)) < 0$
              $N_\mathrm{B} = N_\mathrm{B} + 1$
        endif
            $\mathrm{TL} = \mathrm{TL}\backslash\{T_k\}, \mathrm{RL} = \mathrm{RL}\backslash\{R_r\}$
        if  $N_\mathrm{B} = N_\mathrm{R} - 1$
            $\mathrm{FL} = \{T_1 | L_1(T_i(p)) == 0 \,\&\, \mathrm{RM}(T_i(d)) \leq 0, T_i \in \mathrm{TL}\}$
            $\mathrm{AL} = \mathrm{TL}\backslash\mathrm{FL}$
        else
        AL=TL
        endif
      endwhile
```

Step 4: Return (A,TL,RL,N_B,RM,L_O,L_I).

The deadlock avoidance policy is relatively independent of the dispatching method. So it can be adapted to other dispatching methods.

As task forbidden policy is employed, cares should be taken to the situation when all tasks are forbidden and no task can be assigned any more. So to be deadlock-free, there must be more than one AGV in the system. This condition can be easily satisfied because generally, several AGVs are used in the system. Then, the following theorem can be given.

Theorem 1

If N_R>1, the dispatching procedure with deadlock avoidance policy is deadlock-free.

Proof 3

First, it can be proven that all AGVs will not be blocked. Then, it can be proven that all tasks will not be forbidden. Obviously, if there are two or more active AGVs in the system, then no task will be forbidden and the system is deadlock-free. So only the situation when there is just one active AGV in the system needs to be considered.

Suppose the only active AGV is R_j. For any task T_i=(M_p, M_d), with M_p=T_i(p), M_p=T_i(d). According to the value of L_I(M_P) and RM(M_d). There are four possible cases:

case 1: $L_\mathrm{I}(M_\mathrm{p})>0$ and $\mathrm{RM}(M_\mathrm{d})>0$,
case 2: $L_\mathrm{I}(M_\mathrm{p})>0$ and $\mathrm{RM}(M_\mathrm{d})>0$,
case 3: $L_\mathrm{I}(M_\mathrm{p})>0$ and $\mathrm{RM}(M_\mathrm{d})>0$,
case 4: $L_\mathrm{I}(M_\mathrm{p})>0$ and $\mathrm{RM}(M_\mathrm{d})>0$.

In case 1, the source workstation has input AGV, and the remaining capacity of the destination workstation is greater than 0. If task T_i is assigned to R_j, then R_j will not blocked; according to Lemma 1, the system is not in deadlock.

In case 2, the source workstation has input AGV, and the remaining capacity of the destination workstation is less than or equal to 0. If task T_i is assigned to R_j, then R_j will get blocked. As L_I(M_p)>0, and R_j will load T_i on M_P, so one space can be made for input AGVs of workstation M_p to unload; thus, one of the input AGVs, say R_k, will turn to active, so there is still one active AGV in the system; according to Lemma 1, the system is not in deadlock.

In case 3, the source workstation has no input AGV, and the remaining capacity of the destination workstation is greater than 0. If task T_i is assigned to R_j, then R_j will not blocked; according to Lemma 1, the system is not in deadlock.

In case 4, the source workstation has no input AGV, and the remaining capacity of the destination workstation is less than or equal to 0. If task T_i is assigned to R_j, then R_j will get blocked. And because L_I(M_p)=0, no blocked AGV can be turned to active. Then, all AGVs will get blocked; system deadlock occurs. But under this case, task T_i will be forbidden and T_i will not be assigned to AGV R_j. So R_j will not get blocked.■

Proof 4

On the other side, the situation that all tasks are forbidden will not last forever. As N_R>1, then if there is only one active AGV, there must be one or more blocked AGVs. Then, the output tasks of the workstation where the blocked AGV located will not be forbidden.

From Proofs 3 and 4, by using the deadlock avoidance policy, the system keeps one or more active AGVs and not all tasks will be forbidden forever; thus, the system will not get into deadlock. So the dispatching procedure is deadlock-free.■

Table 4 The two settings for the simulation cases with heavy load

Number	Number of AGVs	Job set	Job arrive rate (p/h)	AGV speed (m/s)
1	9	2	100	6
2	9	1	100	7

As remaining capacity concept is introduced, the deadlock avoidance policy differs from previous policies. From the dispatching procedure, it can be seen that critical tasks are forbidden only when there is just one active AGV in the system, so it is simple and flexible. Thus, the proposed method is expected to be flexible, efficient, and deadlock-free. To verify this expectation, simulation experiments are carried out in the next section.

4 Simulation experiments

To test the dispatching method proposed in this paper, a hypothetical system is designed and three commonly used dispatching methods are compared with the proposed method under different system settings.

The system consists of eight processing workstations, one input workstation, one output workstation, several AGVs, and a unidirectional guide path network. The layout of the system is shown in Fig. 1. The buffer capacities of the workstations are listed in Table 1, and the job types and processing sequences are listed in Tables 2 and 3. The loading and unloading time are both set as 30 s. The three commonly used dispatching methods are:

Shortest travel time first (STTF);
Modified maximum output queue first (MOQF);
Fixed weight multi-attribute (FWMA).

The STTF method is simple single-attribute method. The utility value between task i and AGV j is calculated as: $f(i,j)=f_D$. The MOQF method used here is tailored using maximum output queue first rule first and shortest travel time first rule next if there is a tie. The utility value of this method is:

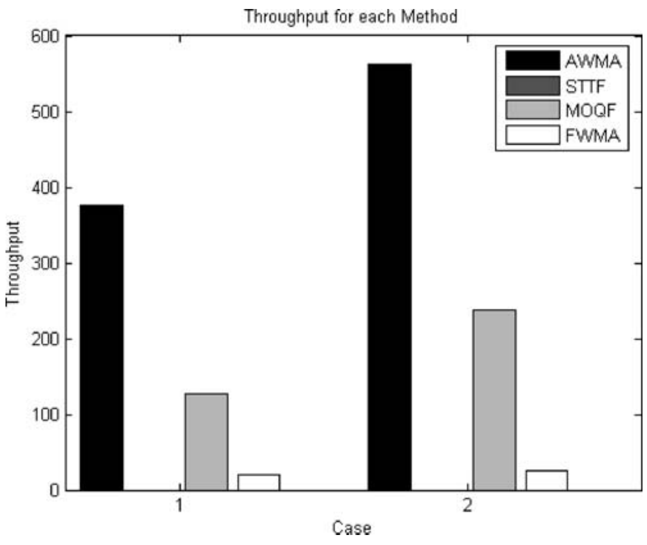

Fig. 4 Throughput for the heavy load cases without deadlock avoidance policy

Table 5 Simulation result for the heavy load cases

Setting	Round	Deadlock	Deadlock time	Completed jobs
AWMA				
1	1	Y	9,261	8
	2	Y	156,193	633
	3	Y	83,418	316
	4	N	–	704
	5	Y	148,357	586
	6	Y	11,395	13
	7	N	–	714
	8	Y	78,783	300
	9	Y	8,687	5
	10	Y	127,024	481
2	1	N	–	1052
	2	Y	46,679	237
	3	Y	52,408	295
	4	N	–	1,051
	5	Y	138,649	817
	6	N	–	1,032
	7	Y	10,024	30
	8	Y	22,260	104
	9	Y	120,474	713
	10	Y	50,654	285
STTF				
1	1	Y	3,057	0
	2	Y	3,287	0
	3	Y	2,981	0
	4	Y	2,639	0
	5	Y	2,296	0
	6	Y	2,389	0
	7	Y	2,066	0
	8	Y	1,908	0
	9	Y	2,364	0
	10	Y	2,206	0
2	1	Y	2,214	0
	2	Y	2,116	0
	3	Y	1,984	0
	4	Y	1,916	0
	5	Y	1,933	0
	6	Y	1,850	0
	7	Y	1,867	0
	8	Y	1,449	0
	9	Y	1,971	0
	10	Y	1,811	0
MOQF				
1	1	Y	20,880	62
	2	Y	11,860	30
	3	Y	5,018	0
	4	Y	28,067	95
	5	Y	21,663	73
	6	Y	92,436	378
	7	Y	12,664	35
	8	Y	23,368	81

Table 5 (continued)

Setting	Round	Deadlock	Deadlock time	Completed jobs
	9	Y	126,756	520
	10	Y	5,421	1
2	1	Y	43,689	254
	2	Y	16,952	89
	3	Y	2,804	0
	4	Y	40,968	234
	5	Y	62,331	370
	6	Y	27,430	159
	7	Y	10,398	38
	8	Y	73,887	453
	9	Y	109,406	663
	10	Y	22,010	117
FWMA				
1	1	Y	12,749	30
	2	Y	11,589	17
	3	Y	8,371	12
	4	Y	12,163	23
	5	Y	10,799	19
	6	Y	12,022	19
	7	Y	11,035	20
	8	Y	11,884	19
	9	Y	11,002	21
	10	Y	10,957	17
2	1	Y	12,980	46
	2	Y	6,946	17
	3	Y	7,349	15
	4	Y	10,434	33
	5	Y	7,493	16
	6	Y	9,011	17
	7	Y	9,055	28
	8	Y	10,743	33
	9	Y	8,607	23
	10	Y	7,785	22

$f(i,j) = Mf_O + f_D$, where M is a large enough number. In the FWMA method, three attributes are adopted: travel distance, output buffer status, and input buffer status, that is: $f(i,j) = W_D f_D + W_O f_O + W_I f_I$. Here, the weights are set as $W_D = W_O = W_I = 1/3$, and the AWMA method is used as described in Section 3. First, experiments without deadlock avoidance policy are carried out to find out under what conditions the dispatching methods will get into deadlock. Then, different dispatching methods with deadlock avoidance policy are compared to find out the efficiency of each method.

4.1 Dispatching without deadlock avoidance policy

The results in [11] show that under normal load conditions, the AWMA method does not lead to deadlock. But under some extreme conditions, the AWMA will get into deadlock. That is, when AGV speed is high, the number of AGV is medium and the job arriving rate is high; the method leads to system deadlock. So the deadlock avoidance policy is necessary.

After several tries, two cases where the AWMA may get into deadlock are selected, which are listed in Table 4. In these cases, the AGV speed is high, the number of AGVs is medium, and the processing load is relatively heavy. Such cases are called heavy load cases. Every case is carried out with ten simulation runs for each method, each with simulation length of 48 h. In the AWMA method, some simulation parameters are set as: F_o=3,F_i=2,K_R=1.6. The simulation result is shown in Fig. 4 for these heavy load cases. Detailed simulation result is given in Table 5.

The simulation results show that under some extreme conditions, the AWMA method may get into deadlock. But compared to other methods, the AWMA method gets into deadlock much later, and it completes more jobs under the same settings. Under heavy load, the other methods cannot

Table 6 The 12 settings for the simulation cases with normal load

Number	Number of AGVs	Job set	Job arrive rate (p/h)	AGV speed (m/s)
1	5	1	12	1
2	6	1	12	1
3	7	1	12	1
4	8	1	16	1
5	9	1	16	1
6	10	1	16	1
7	6	2	12	1
8	7	2	12	1
9	8	2	12	1
10	8	2	16	1
11	9	2	16	1
12	10	2	16	1

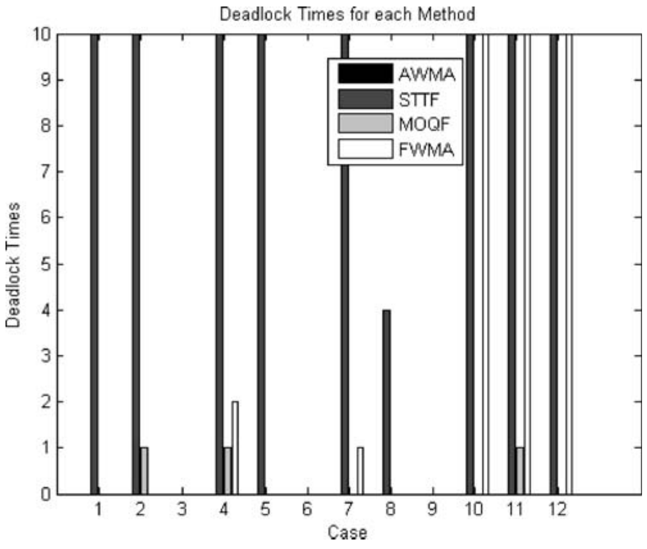

Fig. 5 Deadlock times for each method

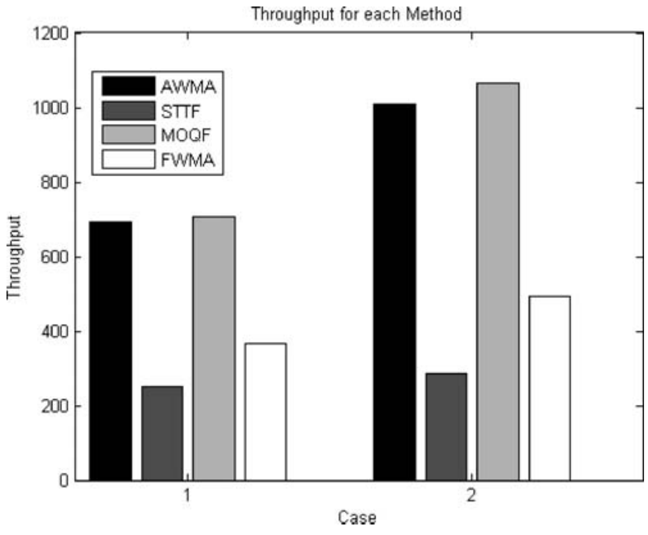

Fig. 7 Throughput for heavy load cases with deadlock avoidance policy

work properly, and they almost complete no jobs before get into deadlock. The AWMA method gets into deadlock eight times in the first setting and seven times in the second setting. The other three methods get into deadlock ten times in both settings. The STTF method gets into deadlock earliest and completes no jobs. This indicates that all methods may lead to deadlock, but the AWMA method gets into deadlock much later and sometimes does not get into deadlock. So the deadlock avoidance policy is necessary.

As in [11], several experiments under normal conditions are carried out. Such cases are called normal load cases. The settings of these cases are listed in Table 6. The deadlock times of each method are shown in Fig. 5. The throughput of each method is shown in Fig. 6.

The simulation result shows that under normal load, the STTF method is the easiest to get into deadlock. It gets into deadlock ten times in eight cases. The FWMA method is the second easiest to get into deadlock, and the MOQF method only gets into deadlock in three cases. The AWMA method does not get into deadlock under normal load. This shows the advantage of the AWMA method. As for the efficiency, when the system resources are rich (cases 3, 6, 9), the output of the four methods is nearly the same; when the system resources are scarce (other cases), the AWMA method gets the best results. The STTF method performs worst. The MOQF method gets better results than the FWMA method under cases 7, 10, 11, and 12 but worse or equal under other cases. The simulation result indicates that the AWMA method can make efficient use of system resources.

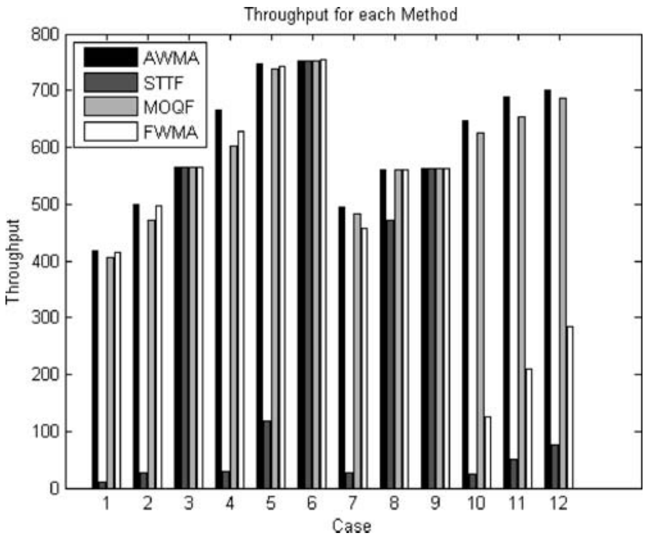

Fig. 6 Throughput for the normal load cases without deadlock avoidance policy

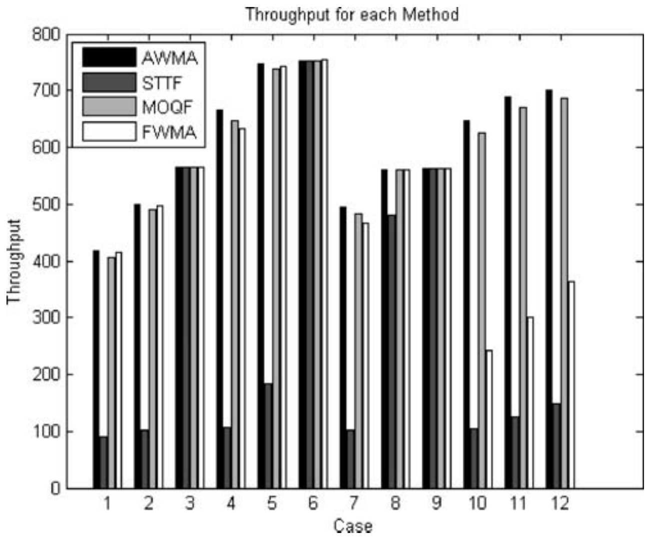

Fig. 8 Throughput for normal load cases with deadlock avoidance policy

4.2 Dispatching with deadlock avoidance policy

Then experiments with deadlock avoidance policy are carried out. The simulation settings are the same as before, whereas deadlock avoidance policy is integrated into all the dispatching methods. Just like before, heavy load cases are carried out first and then normal load ones. When deadlock avoidance policy is used, no deadlock occurs for every method under every case. The simulation result of heavy load cases is shown in Fig. 7 and that of normal ones in Fig. 8.

In the heavy load cases, the MOQF method performs slightly better than the AWMA method, and the STTF method and the FWMA method perform much worse. The STTF method completes the least jobs. The MOQF method gives the output buffer attribute largest weight, so it performs best. In the AWMA method, the load factors are adjusted relatively balanced, so this method performs also well. As the distance is relatively less important in these settings, the STTF method performs worst. So when load is heavy, the most important attributes are those related to block or deadlock.

In the normal load cases, the AWMA method gets the best results when resources are scarce. When resources are rich, all methods perform nearly the same. When resources are scarce, the STTF method performs the worst. It can be seen that single-attribute method is generally less competitive than multi-attribute one. Deadlock avoidance policy can greatly improve efficiency of those methods that easily get into deadlock. For the AWMA method does not get into deadlock under normal load, the performance of this method is not improved by the deadlock avoidance policy. But it is necessary to adopt the deadlock avoidance policy to ensure the system to be deadlock-free.

5 Conclusion

The aim of this work was to develop a flexible, efficient, and deadlock-free dispatching method for automated guided vehicle systems. A deadlock-free AWMA method was proposed. This method differs from previous methods in that it is able to easily adapt to different system settings and ensure the system to be deadlock-free. In this method, multi-attribute dispatching rule was adopted and the weight of each attribute was adjusted according to the system resources and loads of the related aspects. The weights of attributes with scarce resource and heavy loads would get larger values, so load balance can be achieved. To guarantee the system to be deadlock-free, a deadlock avoidance policy based on remaining capacity concept was adopted. This policy works by temporarily forbidding critical tasks according to the system state. Then, the deadlock-free dispatching procedure was given. Furthermore, the deadlock-free property of this method was proven. To show the efficiency of the proposed method, several simulation experiments were carried out on various situations under a hypothetical system. The simulation results show that when deadlock avoidance policy is employed, the AWMA method does not get into deadlock under any conditions. Under normal conditions, the AWMA method performs better than other three commonly used methods when system resources are scarce. This indicates that the proposed method can make efficient use of the system resources and avoid deadlock.

Future works include considering the traffic problem and extending to AGV systems with bidirectional guide path or free path. So combined deadlock avoidance policy for both dispatching and routing is desirable.

Acknowledgments The work reported in this paper is supported by the National Basic Research Program (973Program) of China under grant no 2002CB312204. The authors would like to thank the anonymous reviewers for their helpful comments and suggestions.

References

1. Li J, Dai X, Meng Z (2008) Improved net rewriting system-based approach to model reconfiguration of reconfigurable manufacturing systems. Int J Adv Manuf Technol 37:1168–1189. doi:10.1007/s00170-007-1037-5
2. Li J, Dai X, Meng Z (2009) Automatic reconfiguration of Petri Net controllers for reconfigurable manufacturing systems with an improved net rewriting system-based approach. IEEE Trans Autom Sci Eng 6(1):156–167. doi:10.1109/TASE.2008.2006857
3. Dou J, Dai X, Meng Z (2009) Graph theory-based approach to optimize single-product flow-line configurations of RMS. Int J Adv Manuf Technol. doi:10.1007/s00170-008-1541-2
4. Le-Anh T, De Koster MBM (2006) A review of design and control of automated guided vehicle systems. Eur J Oper Res 171:1–23. doi:10.1016/j.ejor.2005.01.036
5. Vis IFA (2006) Survey of research in the design and control of automated guided vehicle systems. Eur J Oper Res 170(3):677–709. doi:10.1016/j.ejor.2004.09.020
6. Qiu L, Hsu W (2002) Scheduling and routing algorithms for AGVs: a survey. Int J Prod Res 40(3):745–760. doi:10.1080/00207540110091712
7. Egbelu PJ, Tanchoco JMA (1984) Characterization of automatic guided vehicle dispatching rules. Int J Prod Res 22(3):359–374. doi:10.1080/00207548408942459
8. Jeong BH, Randhawa SU (2001) A multi-attribute dispatching rule for automated guided vehicle systems. Int J Prod Res 39 (13):2817–2832. doi:10.1080/00207540110051860
9. Naso D, Turchiano B (2005) Multicriteria meta-heuristics for AGV dispatching control based on computational intelligence. IEEE T Syst Man Cy B 35(2):208–226. doi:10.1109/TSMCB.2004.842249
10. Bilge U, Esenduran G, Varol N, Ozturk Z, Aydin B, Alp A (2006) Multi-attribute responsive dispatching strategies for automated guided vehicles. Int J Prod Econ 100(1):65–75. doi:10.1016/j.ijpe.2004.10.004
11. Guan X, Dai X (2008) Multi-attribute dispatching method with dynamically adjustable weights for multirobot transportation

systems. Proceedings of the IEEE International Conference on Information and Automation (ICIA2008), Zhangjiajie, China, pp 1368–1373

12. Kim CW, Tanchoco JMA, Koo PH (1999) AGV dispatching based on workload balancing. Int J Prod Res 37(17):4053–4066. doi:10.1080/002075499189925
13. Liu FH, Hung PC (2002) Control strategy for dispatching multi-load automated guided vehicles in a deadlock-free environment. J Math Model Algorithms 1:117–134. doi:10.1023/A:1016564209985
14. Yoo JW, Sim ES, Cao C, Park JW (2005) An algorithm for deadlock avoidance in an AGV System. Int J Adv Manuf Technol 26:659–668. doi:10.1007/s00170-003-2020-4
15. Srivastava SC, Choudhary AK, Kumar S, Tiwari MK (2008) Development of an intelligent agent-based AGV controller for a flexible manufacturing system. Int J Adv Manuf Technol 36:780–797. doi:10.1007/s00170-006-0892-9
16. Fanti MP (2002) Event-based controller to avoid deadlock and collisions in zone-control AGVS. Int J Prod Res 40(6):1453–1478. doi:10.1080/00207540110118073
17. Lee JH, Lee BH, Choi MH (1998) A real-time traffic control scheme of multiple AGV systems for collision free minimum time motion: a routing table approach. IEEE T Syst Man Cy A 28 (3):347–358. doi:10.1109/3468.668966
18. Yeh MS, Yeh WC (1998) Deadlock prediction and avoidance for zone-control AGVS. Int J Prod Res 36(10):2879–2889. doi:10.1080/002075498192526
19. Moorthy RL, Wee HG, Ng WC, Teo CP (2003) Cyclic deadlock prediction and avoidance for zone-controlled AGV system. Int J Prod Econ 83:309–324. doi:10.1016/S0925-5273(02)00370-5
20. Wu N, Zhou M (2005) Modeling and deadlock avoidance of automated manufacturing systems with multiple automated guided vehicles. IEEE Trans Syst Man Cybern B 35(6):1193–1202. doi:10.1109/TSMCB.2005.850141
21. Lehmann M, Grunow M, Günther HO (2006) Deadlock handling for real-time control of AGVs at automated container terminals. OR Spectrum 28:631–657. doi:10.1007/s00291-006-0053-4