15th Triennial World Congress, Barcelona, Spain

A PETRI NET BASED APPROACH FOR AGV DISPATCH SCHEDULING AND FLEET SIZE DETERMINATION

Chin-I Liu and P. A. Ioannou

Center for Advanced Transportation Technologies
University of Southern California, Los Angeles, CA 90089-2562

Abstract: In machine job shops with automated guided vehicles (AGVs) as mean of transportation of job parts, it's usually hard to determine the optimal fleet size of AGVs for achieving a desired performance. In this paper a heuristic AGV dispatching algorithm is constructed based on the concept of ideal pipeline and Petri Net theory to balance the job part scheduling and AGV dispatching that minimize the number of AGVs for achieving zero idle time of the machine. The problem is formulated as polynomial equations owing to the mathematical property of Petri Net and since obtaining the optimal solution is computationally difficult, a way to circumvent the difficulty is necessary for the real application of the algorithm. A heuristic dispatching algorithm that schedules the dispatch of AGVs on a selected window basis is used to reduce the computation time and makes the dispatching algorithm capable of real time implementation. *Copyright © 2002 IFAC*

Keywords: Automated guided vehicles, Petri Net, Heuristic, Scheduling, Dispatching

1. INTRODUCTION

In machine job shop with AGVs as mean of material handling, due to the unbalance of job scheduling and AGV dispatching, the phenomenon is well known that the machines in a job shop usually experience repeated cycles of busy periods interleaved with idle periods. The machines may be busy and find many jobs waiting in front of them, while sometimes they're idle without any job to be processed. Deployed more AGVs than required not only increase the operation cost, but also may result in increased congestion that cause delays in completing their tasks. Using too few AGVs results in underutilization of machining resources. It's desirable to achieve the performance requirements by using the smallest number of AGVs.

In order to resolve the required number of AGVs to be deployed, several approaches can be found in the literature. Maxwell (1982) presented an AGV operational featured, time-independent analytical model to estimate the minimum number of AGVs to support the material handling needs in their pioneer work. Egbelu (1987) presented four non-simulation deterministic analytical procedures for estimation the required number of AGVs. Sinriech (1992) developed a multi-criteria optimization model that considers trade-off ratio between cost and throughput to determine the AGV fleet size.

Heuristic methods with appeal stems from their ability to obtain near optimal solution in polynomial time by restricting the search domain toward feasible, efficient schedules have become an important area of research and application. Pipeline is a well-known technique that is used to improve the throughput performance of production lines and some electronic devices. Petri Net as a tool for the optimal scheduling by through the token planning that reduce the number of transitions fired or cycle time of a manufacturing systems is well known (Sun, 1994 & Jeng 1999). In this paper, a heuristic AGV dispatch scheduling algorithm that constructs an AGV dispatch schedule based on the concept of ideal pipeline and Petri Net theory is introduced. The pipeline concept together with the token movement behavior in a Petri Net model can make a balanced dispatching schedule of AGVs through the marking planning. That is, AGVs are represented by tokens and the planned marking in the places governs the dispatching of the AGV in a way that the AGVs supplies jobs for machine and makes the machine as busy as possible. Due to the mathematical feature of Petri Net, the problem can be formulated as a set of polynomial equations, and the AGV's dispatching becomes that follows the solutions of the polynomial equations. For large number of job parts, the solution can't be obtained without long time computation. A heuristic algorithm that applies small schedule window, i.e. schedules a few AGVs' dispatches at a time, can greatly reduce the calculation time and make the algorithm capable of real time implementation. In the simulation, we see that very good performance can be obtained using the proposed dispatching algorithm.

The rest of the paper is organized as follows. Section 2 is the problem formulation. The proposed scheduler model is presented in section 3. The exact solution is discussed in section 4 and the heuristic dispatching algorithm is presented in section 5. Section 6 is a simulation example and section 7 is the conclusion.

2. PROBLEM FORMULATION

In this paper, we focus on solving the AGV dispatch scheduling and the AGV fleet size problems on one job type and one machine job shop, with constant deterministic job processing time and known parameters

associated with machine and AGVs. Since delay of AGVs caused by conflicts is usually inevitable, in our model, we allow delay of AGVs. The problem is formulated as follows. There is only one job type with *M* job parts stored in the shop that needs to be processed by one machine. The tools for transporting the job parts are unit-load AGVs and the processing orders of parts by the machine is based on first come first serve rule with constant service time $T_{service}$ required for each part. The machine can handle at most one job part at a time and the AGV is assumed to stay there until the machine finished the part it carries before it can be dispatched again. It is also assumed that initially the AGVs stay in the machine area, and only one get dispatched every $T_{service}$. Since it is easy to reach the zero idle time of the machine by using much more AGVs than required and introduce more operation cost. It is desirable to find the minimum number of AGVs for achieving the required performance. In this paper we will minimize the number of AGVs subjecting to zero idle time of the machine. More precisely, the problem is to determine the minimum number of AGVs and scheduling the minimum number of AGVs in a way that achieves the zero idle time of the machine. The problem will further be expressed in mathematical formulation.

- **Ideal Minimum Number of AGVs**

The detail of the AGV movement cycle in the shop can be illustrated by a timed place Petri Net model as shown in Figure 1. The minimum mean movement cycle time of AGVs and $N_{\min}$ the ideal minimum number of AGVs required for achieving the zero idle time of the machine is investigated. Then $N_{\min}$ will be used as the initial number of AGVs in the proposed algorithm. The minimum mean movement cycle time of AGV will be used as the criterion for determining the minimum number of AGVs in the schedule window. In the model, circles are places that represent the operation status of the AGVs and bars are transitions that represent events for separating different operation statuses and tokens denote AGVs. A complete AGV movement cycle is the AGV complete the sequence $\{p_0, t_2, p_1, t_3, p_2, t_4, p_3, t_5, p_4, t_1\}$. The places p_0, p_1, p_2, p_3 and p_4 represent the operation statuses of AGV that are serving by the machine, traveling to load a job part, loading the job part, traveling back to the machine and waiting in the machine's queue respectively. The transitions t_1, t_2, t_3, t_4 and t_5 represent the events that the AGV is the first one in the machine's queue, finished the service by the machine, arrived at the job part destination, finished the loading of the job part and arrived at the machine respectively. The process time is associated with places and the transitions are timeless. Followed Figure 1, as the time of each place specified, the movement cycle time of AGV for job part *K*, equals

$$T_{cycle_k} = T_{travel1_k} + T_{load_k} + T_{travel2_k} + T_{que_k} + T_{service} \quad (1)$$

$$T_{trave11_k} = \frac{d_{1_k}}{v_{unloaded}} + T_{delay1_k}$$

$$T_{trave12_k} = \frac{d_{2_k}}{v_{loaded}} + T_{delay2_k}$$

where subscript *k* represents job part *K*, and $v_{unloaded}$ and v_{loaded} are the AGV's speed when it is unloaded and loaded respectively, and d_{1_k}, d_{2_k} are the total travel distances from the machine to the job part and from the job part to the machine respectively. These two distances can be easily calculated for a given job shop configuration. T_{delay1_k} and $T_{dealay2_k}$ are the times that the AGVs need to stop to avoid collision for traveling from machine to the job part and from the job part to machine respectively. The time for processing the job part by a machine or the time the AGV staying in the machine area equals $T_{service}$. T_{load_k} is the time required for the AGV to load the job part and T_{que_k} is the time the AGV waits in the queue to be served by the machine. If there is no other AGV in the queue before the arrival of an AGV, T_{que_k} is 0 for that AGV.

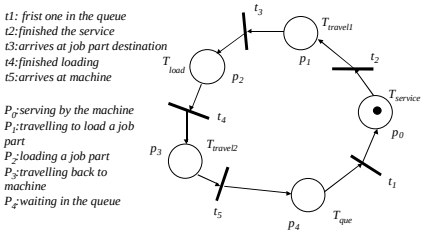

Figure 1: A timed place Petri Net model of an AGV movement cycle

Let the total delay time, T_{delay_k}, for a job part *K* be the summation of $T_{delay1_k}, T_{delay2_k}$ and T_{que_k}, and the real work time required for the job part be T_{work_k}, then

$$T_{delay_k} = T_{delay1_k} + T_{delay2_k} + T_{que_k} \quad (2)$$

$$T_{work_k} = T_{cycle_k} - T_{delay_k} \quad (3)$$

Therefore, the minimum AGV total work time for *M* job parts in the shop to be moved by the AGVs to the machine and served by the machine is

$$T_{\min\_total} = \sum_{k=1}^{M} T_{work_k} \quad (4)$$

The minimum mean time of AGV movement cycle is

$$T_{\min\_mean} = \frac{T_{\min\_total}}{M} \quad (5)$$

Then, $N_{\min}$, the ideal minimum number of AGVs that needs to be deployed in the shop to achieve the zero idle time of the machine is

$$N_{\min} = ceil\left(\frac{T_{\min\_mean}}{T_{service}}\right) \quad (6)$$

where *ceil*(*x*) denotes the smallest integer greater than *x*.

- **The Minimum Number of AGVs Required in the Worst Case**

Let $\overline{T}_{cycle}$ be the time that an AGV departs from a machine for a job part and bring the job part back to the

machine without including the time waiting in the queue and the time served by the machine, i.e.

$$\overline{T}_{cycle} = T_{cycle} - T_{service} - T_{que} \quad (7)$$

For the worst case, suppose that all of the job parts have $\overline{T}_{cycle} = \max(\overline{T}_{cycle})$ and let $M_{\min}$ be the minimum number of AGVs required for achieving the zero idle time of the machine. In the beginning, after the first AGV is dispatched, there are $M_{\min} - 1$ AGVs before the dispatched AGV to be served by the machine. The AGV needs to be back to the machine before time $(M_{\min} - 1) \times T_{service}$ to keep the machine busy. Therefore, $M_{\min}$, needs to satisfy $(M_{\min} - 2) \times T_{service} < \max(\overline{T}_{cycle}) \le (M_{\min} - 1) \times T_{service}$. That is

$$M_{\min} = ceil\left[\frac{\max(\overline{T}_{cycle})}{T_{service}}\right] + 1 \quad (8)$$

After the $N_{\min}$ and $M_{\min}$ are obtained, the one machine job shop AGV scheduling problem becomes searching the real minimum number of AGVs N, $N_{\min} \le N \le M_{\min}$, to achieve the zero idle time of the machine and it can be formulated as

$$\text{Minimize } (N)$$
$$\text{s.t. } N_{\min} \le N \le M_{\min}$$
$$T_{idle} = 0$$
$$M \ge N$$

T_{idle} is the idle time and the number of job parts M needs to be greater than N the minimum number of AGVs.

3. PROPOSED MODEL

An ideal pipeline is that if a job can be divided into n even portions of sub-jobs and each sub-job can be processed by one machine, then line up the n machines and ignore the overhead, the throughput of the line can be n times faster. The ideal pipeline is modeled using Petri Net as shown in Figure 2 that the pipeline has n stages, represented by the resource places p_1 to p_n. The place p_{in} and p_{out} are input and output buffers respectively. The processing time of each stage is T, that is the transitions t_i, i=1,..,n+1, can be fired every time T, if the place $^\bullet t_i$ is marked.

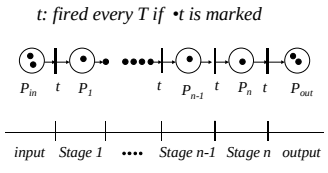

Figure 2: Petri Net Model of ideal pipeline.

Let d be the smallest integer greater than $\overline{T}_{cycle}$ divided by $T_{service}$, i.e.

$$d = ceil\left(\frac{\overline{T}_{cycle}}{T_{service}}\right) \quad (9)$$

Since the time for the machine to process a job part is equal to $T_{service}$, by the time d $\times$ $T_{service}$ after the dispatched of the AGV from the machine area for a job part, the AGV will carry the job part back to the machine. If the time $T_{service}$ is treated as the time for a stage of an ideal pipeline, then initially the job part as shown in Equation (9), is d stages away from the machine. Since the AGV after dispatched is moving toward the machine, every $T_{service}$ AGV is one stage closer to the machine and finally it will reach the machine at time $d \times T_{service}$. The AGVs moving in the shop can be viewed as the products moving in a pipeline stage by stage.

The Petri Net pipeline scheduler is shown in Figure 3, that place p_0 represents the resource place where the machine located, and the processing time of the machine for a job part is $T_{service}$. When the machine finished processing a job part, the finished job part goes to place p_{out} and the AGV goes to the transient place p_{temp} where AGVs get dispatched. The place p_{out}, the output buffer for holding the done job parts can be treated as infinite size. The model is worked in such a way that when the machine in p_0 finished processing a job part, or every $T_{service}$, all the transitions t_i, i =1,…,n will be fired, if the $^\bullet t_i$ is marked. It will make the AGV in the places $\{p_1, p_2, ..., p_n\}$ move one place forward, i.e. the tokens in places $\{p_1, p_2, ..., p_n\}$ move to the next places closer to the place p_0. At the same time, if there is a token in place p_0, it goes to place p_{temp} to be dispatched for a new job part by firing one of the transitions in $\{T_1, T_2, ..., T_n\}$. The move of the token from p_0 to p_{temp} and from p_{temp} to any places in $\{p_1, p_2, .., p_n\}$ can be treated as timeless, that is the dispatching of the AGV can be viewed as simultaneous activity as the moving forward of the tokens in places $\{p_1, p_2, ..., p_n\}$.

Since each place in $\{p_0, p_1, ..., p_n\}$ is a stage of the pipeline, from optimization point of view each place can only be occupied by one AGV. Therefore, in order to have optimal schedule the AGV when dispatched can only be assigned to pick up the job parts that make the AGV fill in the empty places. Since the fired of transitions T_d will move the token from place p_{temp} to p_d, and the d is dependent on $\overline{T}_{cycle_k}$ of the job part K that the AGV is going to pick, or the place p_d is the dth place apart from place p_0. Therefore the assigning of an AGV to a job part K is governed by choosing the part K with $\overline{T}_{cycle_k}$ or the firing of transition T_d with d satisfies $m_{p_d} = 0$ and $d = \left(\frac{\overline{T}_{cycle_k}}{T_{service}}\right)$. The length of AGV pipeline scheduler is dependent on $M_{\min}$ the minimum number of AGV required for the worst case. The relation of $M_{\min}$ with the index n in Figure 3 is that $M_{\min}$ = n+1. Before further investigating the dispatching algorithm, there are some definitions and properties that related to Petri Net (Murata 1989) and the scheduler model need to be illustrated.

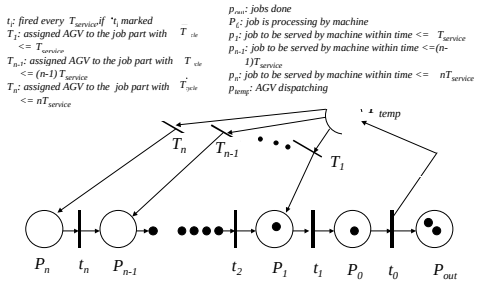

Figure 3: A Petri Net model of the AGV pipeline scheduler of one machine AGV system.

***Definition 1*:** Since the marking of the Petri Net may change every $T_{service}$ by the firing of some transitions in $\{t_n, t_{n-1}, ..., t_0\}$ and in $\{T_n, T_{n-1}, ..., T_1\}$, a step i is the period between the two firing of the transitions. A schedule state S_i is defined as the marking of the places in step i.

***Definition 2*:** For a Petri Net with x transitions and y places, the incidence matrix $A = [a_{ij}]$ is an $x \times y$ matrix of integers and its typical entry is given by $a_{ij} = a_{ij}^{+} - a_{ij}^{-}$, where $a_{ij}^{+} = w(i,j)$ is the weight of the arc from transition i to its output place j and $a_{ij}^{-} = w(j,i)$ is the weight of the arc from transition i to its input place j.

***Definition 3*:** For an ordinary Petri Net, i.e. all of its arc weights are 1's, with x transitions, the firing vector $U = [u_i]$ is a $x \times 1$ matrix of integers and its typical entry $u_i = 1$ if the i transition is fired, otherwise $u_i = 0$. Then the $U_j = [u_i]_j$ is the firing vector of the jth step.

***Property 1*:** The AGV pipeline scheduler model is stable and controllable.

Proof: Since the place p_{temp} is transient and timeless and can be treated as input place for the places $\{p_1, p_2, ..., p_n\}$ and the size of p_{out} is assumed to be infinite. We should only concern about the marking of places $\{p_n, p_{n-1}, .., p_0\}$. Let the incidence matrix between the transitions $\{t_n, t_{n-1}, ..., t_0\}$ and places $\{p_n, p_{n-1}, ..., p_0\}$ be A_1 and the incidence matrix between transition $\{T_n, T_{n-1}, .., T_1\}$ and places $\{p_n, p_{n-1}, .., p_0\}$ be B, then

$$A_1 = \begin{bmatrix} -1 & 0 & 0 & . & . & 0 \\ 1 & -1 & 0 & . & . & 0 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 0 & 0 & . & . & -1 & 0 \\ 0 & 0 & . & . & 1 & -1 \end{bmatrix}, B = \begin{bmatrix} 1 & 0 & 0 & . & . & 0 \\ 0 & 1 & 0 & . & . & 0 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 0 & 0 & 0 & . & . & 1 \\ 0 & 0 & 0 & . & . & 0 \end{bmatrix}$$

where A_1 is $(n+1)\times(n+1)$ and B is $(n+1)\times n$. The schedule state equation can be written as

$$\begin{bmatrix} m_{p_n} \\ m_{p_{n-1}} \\ . \\ . \\ m_{p_1} \\ m_{p_0} \end{bmatrix}_i = \begin{bmatrix} m_{p_n} \\ m_{p_{n-1}} \\ . \\ . \\ m_{p_1} \\ m_{p_0} \end{bmatrix}_{i-1} + A_1 \begin{bmatrix} t_n \\ t_{n-1} \\ . \\ . \\ t_1 \\ t_0 \end{bmatrix}_{i-1} + B \begin{bmatrix} T_n \\ T_{n-1} \\ . \\ . \\ T_1 \end{bmatrix}_{i-1} \quad (10)$$

where $[m_{p_n} \quad m_{p_{n-1}} \quad . \quad . \quad m_{p_1} \quad m_{p_0}]_i^T$ is the schedule state that represents the marking of the places $\{p_n, p_{n-1}, .., p_0\}$ at ith step, and $[t_n \quad t_{n-1} \quad . \quad . \quad t_1 \quad t_0]_{i-1}^T$ and $[T_n \quad T_{n-1} \quad . \quad . \quad T_1]_{i-1}^T$ are the firing vectors of transitions $\{t_n, t_{n-1}, .., t_0\}$ and $\{T_n, T_{n-1}, .., T_1\}$ respectively. Since the firing vector of $\{t_n, t_{n-1}, ..., t_0\}$ at step $i-1$ is based on the marking of the places $^{\bullet}t_j, j \in \{1, 2, \ldots, n\}$, that is exactly the schedule state at step $i-1$. Therefore, the firing vector $[t_n \quad . \quad . \quad . \quad . \quad t_0]_{i-1}^T$ of $\{t_n, t_{n-1}, ..., t_0\}$ at step $i-1$ is equal to the schedule state $[m_{p_n} \quad . \quad . \quad . \quad . \quad m_{p_0}]_{i-1}^T$. The state equation (10) can thus become

$$\begin{bmatrix} m_{p_n} \\ m_{p_{n-1}} \\ . \\ . \\ m_{p_1} \\ m_{p_0} \end{bmatrix}_i = \begin{bmatrix} 0 & 0 & 0 & . & . & 0 \\ 1 & 0 & 0 & . & . & 0 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 0 & 0 & . & . & 0 & 0 \\ 0 & 0 & . & . & 1 & 0 \end{bmatrix} \begin{bmatrix} m_{p_n} \\ m_{p_{n-1}} \\ . \\ . \\ m_{p_1} \\ m_{p_0} \end{bmatrix}_{i-1} + B \begin{bmatrix} T_n \\ T_{n-1} \\ . \\ . \\ T_1 \end{bmatrix}_{i-1} \quad (11)$$

Replace the schedule state $[m_{p_n} \quad m_{p_{n-1}} \quad . \quad . \quad . \quad m_{p_0}]^T$ by S and firing vector $[T_n \quad T_{n-1} \quad . \quad . \quad . \quad T_1]^T$ by U and let

$$A = \begin{bmatrix} 0 & 0 & 0 & . & . & 0 \\ 1 & 0 & 0 & . & . & 0 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 0 & 0 & . & . & 0 & 0 \\ 0 & 0 & . & . & 1 & 0 \end{bmatrix},$$

Then state equation (11) becomes

$$S_i = AS_{i-1} + BU_{i-1} \quad (12)$$

By looking at equation (12), we can verify the stability and controllability as follows:

Stability: Since all of the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_{n+1}$ of matrix A are within the unit circle, i.e. $|\lambda_j| \le 1, j \in \{1, 2, \ldots, n+1\}$, the system is stable.

Controllability: Since its controllability matrix has full row rank, i.e. $rank[B, AB, ..., A^nB] = n+1$, the system is controllable.

***Definition 4*:** A sequential schedule state S_{i+1} of S_i is that the tokens in places p_{z+1} in step i move to the places p_z in step $i+1$.

***Definition 5*:** An admissible schedule set π is a set of schedule states such that any two adjacent schedule states S_{i+1} and S_i, S_{i+1} is a sequential schedule state of S_i, and each schedule state in the set has its places p_0 marked, i.e. $m_{p_0} = 1$.

***Theorem 1*:** For one machine AGV system scheduling using AGV pipeline scheduler, let F be a sequence of firing vectors, if F makes a set of schedule states θ and $\theta \in \pi$, then the sequence F is a schedule that will achieve zero idle time of the machine.

Proof: The schedule set θ is a subset of admissible schedule set that has the place p_0 always marked. Since p_0 is always marked, it means that the machine is always busy, i.e. the idle time of the machine is zero.

The AGVs follow the sequence of firing vector F can achieve the zero idle time of the machine. Therefore, the sequence F is a schedule that achieves zero idle time of the machine.

Without loss of generality, let matrix B' be an identical matrix $I_{(n+1)\times(n+1)}$ and $U' = [T_n \quad T_{n-1} \quad . \quad . \quad . \quad T_1 \quad 0]^T$, then state equation (12) can be written as

$$S_i = AS_{i-1} + B'U'_{i-1} = AS_{i-1} + U'_{i-1} \qquad (13)$$

Therefore, the zero idle time of machine job shop using minimum number of AGVs scheduling problem is that: To find the minimum number of AGVs N and a job part associated with the AGVs with $\overline{T}_{cycle}$ in each step i, s.t.

$$S_i = AS_{i-1} + U'_{i-1}$$
$$m_{p_0} = 1 \qquad \textit{for each schedule state } S_i$$
$$m_{p_n} + m_{p_{n-1}} + ... + m_{p_0} = N$$
$$m_{p_j} \le 1 \quad \textit{for } j = 1,2,3,....., n$$
$$ceil\left(\frac{\overline{T}_{cycle}}{T_{service}}\right) = d$$
$$M \ge N \ , \ M \textit{ is the total number of job parts}$$

4. EXACT SOLUTION

By expanding the formulation above for M job parts scheduling, the solution of the zero idle time using minimum number of AGVs scheduling problem by enumerating all of the possible cases is equivalent to solve the following two procedures iteratively.

Procedure 1: Find the firing sequences $[(T_{d_0})_0 \ (T_{d_1})_1 ..... (T_{d_{M-1}})_{M-1}]_b$, $b = 1,2,....$ that satisfy

$$(T_n)_j + (T_{n-1})_j + ... + (T_1)_j = 1, \quad j = 0, 1, ..., M-1$$

$$\begin{cases} (m_{p_n})_k = (T_n)_{k-1} \\ (m_{p_{n-1}})_k = (m_{p_n})_{k-1} + (T_n)_{k-1} \\ \quad . \\ \quad . \qquad\qquad k = 1,2,.....,M \\ \quad . \\ (m_{p_1})_k = (m_{p_2})_{k-1} + (T_1)_{k-1} \\ (m_{p_0})_k = (m_{p_1})_{k-1} \end{cases}$$

$$(m_{p_n})_i + (m_{p_{n-1}})_i + ... + (m_{p_{n-1}})_i = N, \ i = 0, 1, ..., M-1 \qquad (14)$$
$$(m_{p_y})_0 \le 1, (m_{p_y})_1 \le 1, ..., (m_{p_y})_M \le 1, \quad y = 1,2,....., n$$
$$(m_{p_0})_j = 1, \quad j = 1, 2, ..., M$$

Procedure 2: Sequencing the job parts to satisfy any of the firing sequences in $[(T_d)_0 \ (T_d)_1 .....(T_d)_{M-1}]_b$, such that each job part K with it's $\overline{T}_{cycle_k}$ for an AGV satisfies

$$ceil\left(\frac{\overline{T}_{cycle_k}}{T_{service}}\right) \le [(d)_{k-1}]_b \quad k = 1,....,M \qquad (15)$$

b=1, 2, … are the possible sequences obtained by solving Equation (14). The sequences that have its final scheduling state $S_M = \{0, ..0, 1, 1, ...,1\}$ with the places p_0 to p_{N-1} marked and the M job parts satisfied equation (15) is the dispatching schedule that the AGVs can follow to achieve the zero idle time using the N AGVs. If the M job parts in the shop can't make any of the firing sequences in the set b to have equation (15) satisfied, increasing the number of AGVs by one, i.e. $N = N+1$ and repeat the procedures 1 and 2, until equation (15) is satisfied or $N = M$.

5. PROPOSED DISPATCHING ALGORITHM

As the computation for optimal scheduling has a combinatory flavor, a scheduling for a smaller number of job parts is required to avoid lengthy computation. Therefore, a window that confines the number of job parts to be scheduled at a time is applied and a criterion for determining the minimum number of AGVs in a scheduling window is also necessary. The size of the window governs the number of job parts to be scheduled and picked by AGVs. If the window size is w, then only w job parts are scheduled to be picked by the AGVs at a time.

5.1 The Criterion for Determining Minimum Number of AGV in a Schedule Window

Since only w job parts are scheduled in a window, by using arbitrary number of AGVs, the firing sequences can be easily obtained using equation (14) and it may also easy to find plenty set of w job parts that satisfy equation (15). Therefore, since the use of window, a criterion is required for determining the minimum number of AGVs in the window. The minimum mean cycle time of AGVs is used as the criterion to determine the number of AGVs to be deployed, i.e. the number of AGVs can achieve the zero idle time and also have the job parts satisfied $\frac{\sum_{i=1}^{x} T_{work_i}}{x} \ge T_{\min\_mean}$, where $x = w, 2w, 3w,..$ is the number of job parts that have been scheduled to be processed or already been processed by machine.

5.2 Time Lending and Time Borrowing

Since the schedule is on a window by window basis, the first state of current window is the sequential state of the last state in the previous window and the first state of next window is the sequential state of the last state of current window, and so forth. The scheduling of current window is related to the schedule of the previous and next windows. We know that, in order to reach the zero idle time, the places p_0 and p_1 need to be always marked. If some AGVs are assigned to pick up the far job parts, then there are some AGVs needed to be assigned to the near job parts in order to have the places p_0 and p_1 marked. Therefore, if in the previous window, AGVs are assigned to pick up far job parts, then in the current window, the AGVs may be forced to pick up near job parts. The summation of the job parts, $\sum T_{work}$, scheduled in each window is dependent on the adjacent window. Then the total real work time for the job parts in a window is $T_{win} = \frac{\sum_{i=1}^{w} T_{work_i} + T_{lend} - T_{borrow}}{w}$, where T_{lend} is the time borrowed by the previous window and T_{borrow} is the time borrowed from the next window.

5.3 Dispatching Algorithm

The dispatch of AGVs is scheduled in a window by window basis. It is desirable to start from using the ideal minimum number of AGVs $N_{\min}$ to search the schedule for meeting the zero idle time requirement. A dispatch scheduling algorithm for generating a non-delay schedule, a schedule in which the machine is kept busy at any time when it could begin processing some operation is now presented using the following steps.

Step 1 Calculate the minimum mean time of the AGV movement cycle $T_{\min\_mean}$, the ideal minimum number of AGVs $N_{\min}$ and the length of the pipeline scheduler *n*.

$$T_{\min\_mean} = \frac{1}{M}\sum_{k=1}^{M} T_{work_k}\ ,\ N_{\min} = ceil\left(\frac{T_{\min\_mean}}{T_{service}}\right) \text{and}$$

$$n = ceil\left[\frac{\max(\overline{T}_{cycle})}{T_{service}}\right] - 1$$

Step 2 Initialize the schedule state S_0 to be {0,.., 0, 1, 1,..,1} with $N_{\min}$ of 1's in the places p_0 to $p_{(N_{\min}-1)}$. Let $N = N_{\min}$ and determine the window size *w*.

Step 3 Use window size *w*, the length of the scheduler *n*, N and the initial state of the window S_0 to find the possible firing sequences *F* for the *N* AGVs such that *F* = $[(T_{d_0})_0\ (T_{d_1})_1 \ldots\ldots (T_{d_{w-1}})_{w-1}]$ satisfies equation (14), with *M* replaced by *w* in (14).

Step 4 Find the *w* job parts with $\overline{T}_{cycle}$ satisfies that

$$ceil\left(\frac{\overline{T}_{cycle_k}}{T_{service}}\right) = [(d)_{k-1}]_b \quad k = 1,\ldots,w \qquad (16)$$

for the *d* of T_d's in each set of firing sequence and with

$(T_{win})_b = \dfrac{\sum_{i=1}^{w} T_{worki} + T_{lend} - T_{borrow}}{w}$ for the firing sequence *b* such that

$$\frac{\sum_{c=1}^{q}(T_{win})_c + (T_{win})_b}{g+1} \geq T_{\min\_mean} \qquad (17)$$

where *g* is the number of windows that have been scheduled before current window.

Step 5 If a firing sequence *b* satisfies equation (17), then the firing sequence *b* is the schedule to be followed and the job parts satisfy equation (16) are the parts scheduled to be processed and goes to step 6. Otherwise increase N by 1, i.e $N = N + 1$, and goes to step 3.

Step 6 Execute the tasks by following the selected firing sequence and job parts. Use the final state of the current window as S_0 of the next window and goes to step 3 to find the schedule for the next window, until no more job parts in the shop or the process needs to be ended.

6. SIMULATION

In the simulation, we only compare 6, 7 and 8 job part cases and the results are shown in Table 1. The job parts are randomly selected from a shop. Initially, the AGVs depart from the machine area every $T_{service}$, i.e. initial schedule state S_0 = {0,.., 0, 1,...,1} with $N_{\min}$ (*N*) of 1's in the places p_0 to $p_{(N_{\min}-1)}$. The ideal minimum number of AGVs is $N_{\min}$ and the window size is *w*=5 for all cases. The $\overline{T}_{cycle}$ of the selected parts is listed in the table from the smallest to the biggest. The minimum number of AGVs *N*, the machine idle time, and the scheduled sequence of the exact solution and heuristic algorithm are listed in the table separately. The results show that the heuristic solutions are the same as the exact solutions for all of the random selected cases.

Table 1: Comparison of exact solutions and heuristic solutions for 6,7 and 8 job parts

No of parts	$\overline{T}_{cycle}$ of job parts (sec.)	$N_{\min}$	Exact solution			Heuristic solution			
			N	Idle Time (sec.)	Scheduled job part Sequence	T_{min_mean}	N	Idle time (sec.)	Scheduled job part Sequence
6	56.7, 77.1, 116.7, 141.9, 189.6, 250	6	6	0	250.2, 116.7, 141.9, 189.6, 56.7, 77.1	167.7	6	0	250.2, 189.6, 77.1, 116.7, 56.7, 141.9
7	54.3, 56.1, 79.2, 129.3, 153.2, 217 264.6	6	6	0	264.6, 129.3, 79.2, 217.5, 153.3, 54.3, 56.1	166.3	6	0	153.3, 217.5, 264.6, 56.1, 79.2, 129.3, 54.
8	41.1, 74.7, 96.6, 107.4, 131.4, 164 229.2, 232.5	6	6	0	232.5, 229.2, 74.7, 107.4, 41.4, 164.4, 96.6, 131.4	164.25	6	0	164.4, 96.6, 229.2, 232.5, 74.7, 107.4, 41.4, 131.4

7. CONCLUSION

In this paper, a heuristic AGV dispatching algorithm based on the concept of ideal pipeline and Petri Net theory is constructed and implemented for balancing the dispatching of AGVs to achieve the zero idle time of machine using minimum number of AGVs. Since the algorithm applies small schedule window that only schedules a few AGVs' dispatches at a time, the calculation time for finding the suitable solution is greatly reduced so that makes the algorithm capable of real time implementation. The simulation result shows that the use of the proposal algorithm obtains very close result as the optimal solution in some random selected cases.

REFERENCES

Maxwell W. L., et al., (1982) "Design of Automatic Guided Vehicles Systems," *IIE Transactions*, v14, pp114-124.

Egbelu P.J., (1987) "Pull versus Push Strategy for Automated Guided Vehicle Load Movement in a Batch Manufacturing System," *Journal of Manufacture System*, v6, pp209-221.

Sinriech D.,et al., (1992) "An Economic Model for Determining AGV Fleet Size," *International Journal of Production Research* ,v30, No.6, pp1255-1268.

Murata T, (1989) "Petri Nets: Properties, Analysis and Applications", *Proceedings of the IEEE*, vol. 77, No 4, pp. 541-579.

Sun T.-H., et al., (1994) "A Petri Net Based Approach to Modeling and Scheduling for an FMS and a Case Study", *IEEE Transcations on Industrial Electrics*, vol. 41, No 6, pp. 593-601.

Jeng M. D.,et al., (1999) " Heuristic Search Based on Petri Net Structures for FMS Scheduling", *IEEE Transcations on Industry Applications*, vol. 35, No 1, pp. 196-202.