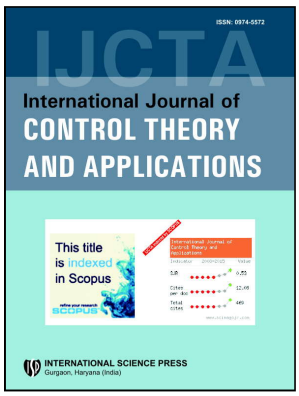


International Journal of Control Theory and Applications

ISSN : 0974-5572

Volume 10 • Number 14 • 2017

Deadlock Avoidance and Re-routing of Automated Guided Vehicles (AGVs)

B. Satish Kumar[1], G. Janardhana Raju[2], and G. Janardhana Ranga[3]

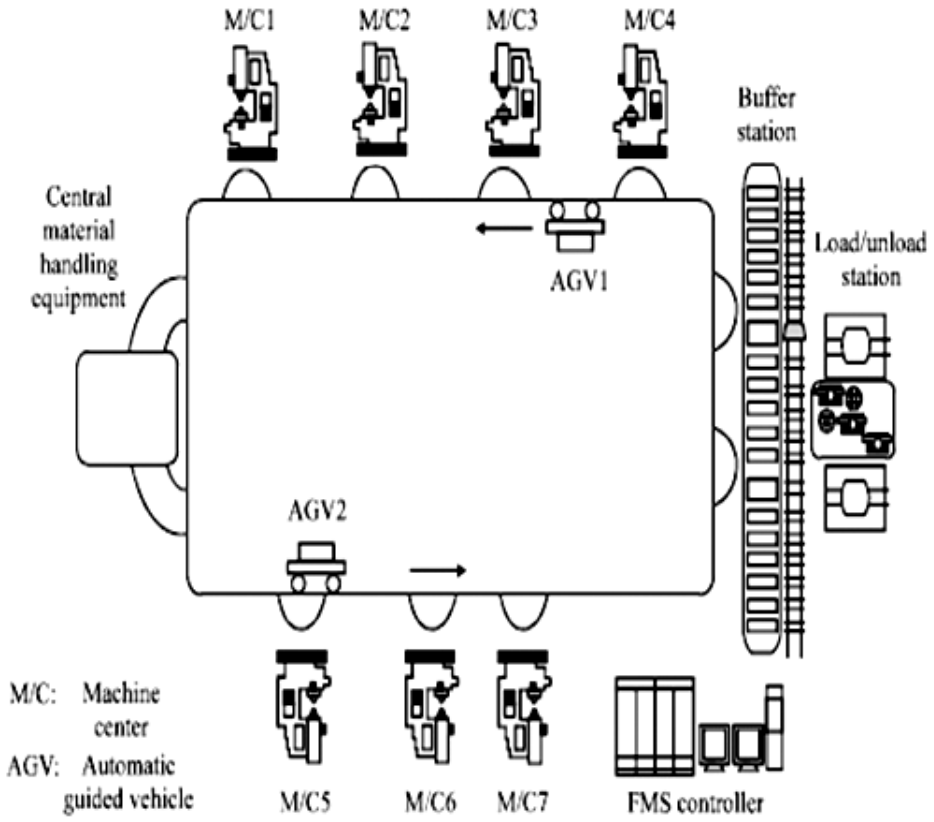
[1] *Department of Mechanical Engineering, S.R. Engineering College, Warangal. Telangana. INDIA, Email: satishbk91@gmail.com*

Abstract: The main aim of this paper is to create an algorithm, to code it and to execute it when executed to see that as it generates a safe sequence which is to be followed to avoid the deadlocks. The flexible manufacturing system (FMS) is a manufacturing system in which there is finite amount of flexibility one that allows a system changes to react in known possibilities either predicted or not predicted. Generally it is considered to be taken in two types. The TYPE - I, is a machine flexibility which allows the work stations to perform various operations on a part when there is a sudden change in demand. The TYPE - II, is a routing flexibility that expands literally work floor then by utilizing multiple no: of machines to execute the operations via routing. Automated Guided Vehicles (AGV) entails as TYPE - II and it acts as the hosts for an execution of various functions which includes the picking and a dropping, docking & undocking that is required when this routing flexibility comes into picture.

Keywords: AGVs, Deadlock, FMS, Machine cell, Rerouting.

1. INTRODUCTION

Advanced manufacturing systems today have to deal with the criteria involving a multiple workstations and different machines, which are varying the AGVs and the customer requirements with a time. All these parameters which are to be controlled and coordinated reveals how industries are to automate most parts of their work floor inorder to achieve an un-interrupted production in turn satisfying the customer demands with possible low production costs. One such application of the automated systems in the material storage and retrieval systems involve in the Automated Guided Vehicles (AGVs) and which are human - man power less, computer controlled machines which move around the shop floor performing actions like storage and retrieval, a picking and a dropping, the docking and an undocking kind of the tasks which have been linked to a production process. This paper entails deadlock avoidance and the rerouting of the AGVs by developing one particular strategy, then by writing an algorithm and by encoding it in a software language. Deadlock can be defined as a situation where a part or the entire system of AGV stalls. When this condition prevails on the work floor, then the activities of which were crucial to a production and also a stall causing the breakdown in the production the one which is an unfavorable situation we developed a strategy by analyzing the Maximum total number of AGVs, current status,



Allocated AGVs, free (Available AGVs) and future requirements of a different work stations for various tasks that to be done by an AGVs. All above information that is stored in a database from the user, AGVs and a machine cell that can access data required therefore by providing an ability to create an order on work floor for required task to be done voluntarily.

With above data available in data base the execution of an algorithm starts by receiving/issuing all inputs and follows predefined rational steps and mathematical rules written program will be viewed in further in flow chart and therefore it yields a result that will be in form of sequence of the processes that to be followed in order to avoid deadlock, ensure safe path of an AGV that started from source to the end at destination. The o/p generated will be the sequence of all processes that to be carried out in order to avoid possible interruptions in the production and in the deadlock conditions.

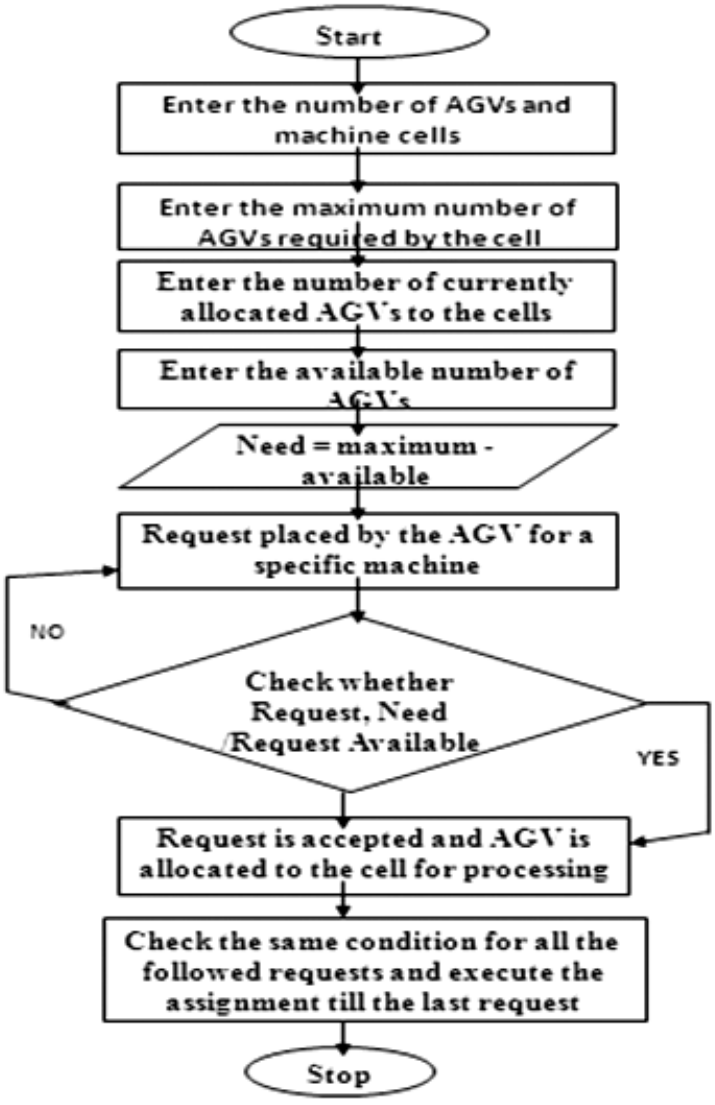
2. LITERATURE SURVEY

Deadlock detection has been a very significant exploration topic; there are few papers that were written in this field. But most of them are literatures which are suitable for simple AGVS layout. Lee and Lin applied the theory of petri-nets in to AGV systems. They are extended an idea of applying petri-nets on Fms and it is proposed by Viswanadham, Narahari and Johnson for the deadlock prediction & the avoidance. In his paper, actual implementation of the algorithm was clear. Complexity of a method is O (nm2) where 'n' is the no: of nodes and 'm' is the no: of arcs, Cho, Kumaran, Richard & Wysk's approach for the deadlock detection and for the resolution, it is to classify network layout as the bounded circuits. The definition of a bounded circuit is similar to that of the cycles. Using these results from graph theories, a method, which is able to detect an impending deadlock situation. The complexity of an algorithm is O(wn(n+m)2) where w is the no: of cycles. A conflict resolution route planning method was revised by Revelators using a modified version of banker's algorithm who proposed by Chang, Tanchoco and the Koo. The method in brief comprises of an incrementing a route for

a vehicle to one zone at a time. This is done by first testing whether the system is in a "safe" mode before deciding vehicle's next zone. The complexity of an algorithm was shown to be O (V2) where V is the no: of vehicles.

3. ALGORITHM

Execution of algorithm starts when the request is placed in system by the work station for an AGV to perform a prescribed task. A request is processed by comparing requested demand (No. of AGVs) to available AGVs & two possibilities that are generated as follows. Firstly, when requested order is above available then it denies as it is not possible for system to allocate more AGVs than available. In the other case if it is below, algorithm checks for circular and wait condition existing in queue & then request is accepted with a condition that AGV is left free at an instant the task has been completed as requested by work station & adds up to available set. Completed process is then terminated from queue & an update is sent to the data base which comprises of refreshed information about no: of available, allocated and needed sets of AGVs for other processes. The cycle is again repeated for every request that made by the machine cell till all the processes in queue that are finished as shown pictorially.



4. MATH

Safety Algorithm: The algorithm for finding whether it is a system or not a system which is in a safe state. Then, this algorithm can be described as follows.

Let work and finish be vectors of length m and n respectively.

Initialize

Work=Available and Finish[i] =false for i=0, 1,…. n-1.

Find an i such that both

a. Finish[i] = false

b. Need = work

If no such one exists, go to the step 4.

Work = work + Allocation

Finish[i] = true

Go to step2

If Finish[i] = true for all one, then system is in a safe state.

Machine - Request Algorithm: The algorithm which is determined

If the requests can be safely granted

Request, the request vector for AGV Ai, if Request[j] = k, then AGV Ai wants k instances of machines type Mj, when a request for the machines which is made by AGV, is Ai, following actions which have taken.

If Request ≤ need, go to step 2.

Otherwise,

Raise an error condition,

Since the AGV has exceeded its maximum claim.

If Request ≤ Available, go to step 3.

Otherwise Ai, must wait, since the machines are not available.

Have the system pretend to have allocated the requested machines to AGV Ai by modifying the state as follows;

*Available=Available-Request;

*Allocation=Allocation +Request;

*Need=Need-Request;

If the resulting machine - allocation position, state is safe, then the transaction is completed, & AGV Ai is allocated to its machines. However, if the new state is unsafe, then Ai must wait for Request, and for the old machine-allocation state that is restored.

5. EQUATIONS

Several instances of a Machine type:

It defines the instances and states machine in the work station denoted as below.

Available: A vector of length m indicates no: of available machines of each type.

Avail[i][j]=k

Max. : the n*m matrix that defines max. no: of machines that may needed by AGVs for its processing.

Max[i][j]=k

Allocation: An n*m matrix defines the number of machines of each type currently allocated to each AGV.

Allot[i][j]=k

Request: The n*m matrix indicates current request of each AGV

if Request[i][j] equals k, then

AGV Ai is requesting k more instances of machine type Mj.

Let work and Finish be vectors of length m and n respectively,

Initialize Work = Available. For i=0,1…n-1.

If allocation ‘‘‘ 0, then finish[i] = false

Otherwise finish[i] = true

Find an index I such that both

Finish[i] = false

a. Request ≤ work

If no such one exists, then, go to step 4

Work = work + allocation

Finish[i] = true, Then Go to step2

If finish[i] = false for some i, 0 ≤ i < n, then the system is in a deadlock state. Moreover if finish[i] = false then job Ji is dead locked [7].

6. RESULT

The input parameters which are fed into a system that includes a no: of AGVs & machine cells on a shop floor, the maximum and currently allocated AGVs to work stations, available no: of AGVs, the first request initiated by a specific cell to AGV like the factors that are compiled. When a program is runned then the o/p obtained is a proper sequence of the tasks to be accomplished by the AGVs in order to avoid any chance of occurrence of the deadlocks. When situations like above were aroused then the algorithm forces AGVs to perform tasks as per the priority one which is labeled depending upon a role importance of a task in production instead of the following sequence is obtained. It implies the deviation of performing tasks from sequence which is generated by a program which leaves the system in an unsafe state. A deadlock avoidance algorithm is awaken when this kind of situations are encountered on a shop floor. Depending upon no: and the values given as i/p to a system then, a sequence is generated that does not include all the possible tasks that are incomplete then, the sequence is obtained, one which notifies that the immediate process after this sequence may lead to the deadlock. The deadlock avoidance algorithm is basically a decision making strategy with few logical steps one which assigns a priority to requests placed by machine cells for AGVs.

7. CONCLUSION

The banker's algorithms which were implemented in resolving of problems related to routing of AGVs which frequently undergo the deadlock like situations that were generated a solution which was found to be very effective in the form of a sequence of jobs what to be done by AGVs so as to avoid deadlock condition prevailing shop floor. The algorithm written & coded in a computer language C++ is capable of producing results for a no: of inputs given allows a user to obtain the safe sequence with less computational efforts. The work can conclude that, the algorithm which is presently encoded in C++ language if generated in further advanced computer language the implementation can be further taken a step ahead to industrial Robots, AGVs, CNCs and all other equipments on the shop floor which are governed mostly by computers hence laying a path for fully automated, human less production operations on work floor. The algorithm not only produces a safe sequence for any number of inputs but also instant as all the required information about the status of enterprise and work floor like machines, AGVs, MSP are already provided and stored in a central database which can be easily accessed.

8. FUTURE SCOPE

- Application of the INTRANET concept becomes knowledgeable of another AGV, thus no deadlock can prevail - as their destinations & sources will be predefined.
- Implementing Graphical Information System (GIS) as the vision system so that the AGV can re-route itself when senses any obstacle in its path.
- The program written in software language when interacted with simulation soft ware like FLEXSIM and AUTOMOD animation can be viewed

REFERENCES

[1] Lee, C.C., Lin, J.T. (1995). Deadlock Prediction and Avoidance based on Petri netsfor zone control automated guided vehicle systems. INT.J.PROD.RES., v33, n12,2349-3265.

[2] IJSRD - International Journal for Scientific Research & Development| Vol. 3, Issue 09, 2015 | ISSN (online): 2321-0613,pp 471 - 475

[3] Viswanadham N., Narahari Y., and Johnson, T.L. (1990). Deadlock prevention and deadlock avoidance in flexible manufacturing systems using Petri net models. IEEE Transactions on Robotics and Automation, 6(6), 713-723.

[4] Hyuenbo, Cho, Kumaran, T.K., and Richard, A.Wysk. (1995). Graph heoreticDeadlock Detection and Resolution for Flexible Manufacturing Systems. IEEETransactions on Robotics and Automation, v11, n3, 413-421

[5] Spyros A. Reveliotis. Conflict Resolution in AGV Systems. School of Industrial & Systems Engineering, Georgia Institute of Technology.5. Chang, W.K., Tanchoco, J.M.A., and Koo, P.H. (1997). Deadlock Prevention inManufacturing Systems with AGV Systems: Banker's Algorithm Approach. Journalof Manufacturing Science and Engineering, v119, 849-854.

[6] International Journal of Engineering Research & Tech,(IJERT)ISSN:2278-0181,www.ijert.orgIJERTV4IS010011,Vol. 4 Issue 01, January-2015, pp – 35 - 37

[7] Benjamin Zhan F. (1996). Three Fastest Shortest Path Algorithms on Real RoadNetworks: Data Structures and Procedures. Journal of Geographic Information andDecision Analysis, vol.1, no.1, pp. 69-82.

[8] Journal for Research| Volume 01| Issue 11 | January 2016 ISSN: 2395-7549, pp 1- 4

[9] Coffman, E.G., Elphick, M.J., and Shoshani, A. (1971). System deadlocks. ACMComputing Surveys, 3(2), 67-78.

[10] Goirgio Gallo, Stefano Pallottino (1988). Shortest Path Algorithms. Annals ofOperations Research. J.C. Baltzer Scientific Publishing Company. Gold, E. Mark. (1978). Deadlock Prediction: Easy and Difficult Cases. SIAM J. onComputing, v7, n3, 320-336.