# DEADLOCK AVOIDANCE AND RE-ROUTING OF AUTOMATED GUIDED VEHICLES (AGVS) IN FLEXIBLE MANUFACTURING SYSTEMS (FMS)

MD. Saddam Hussain[1], B. Satish Kumar[2], Dr. G.Janardhana Raju[3]
Email:[1]Saddam.mohd321@gmail.com, [2]satishbk91@gmail.com,[3]gjraju_06@rediffmail.com

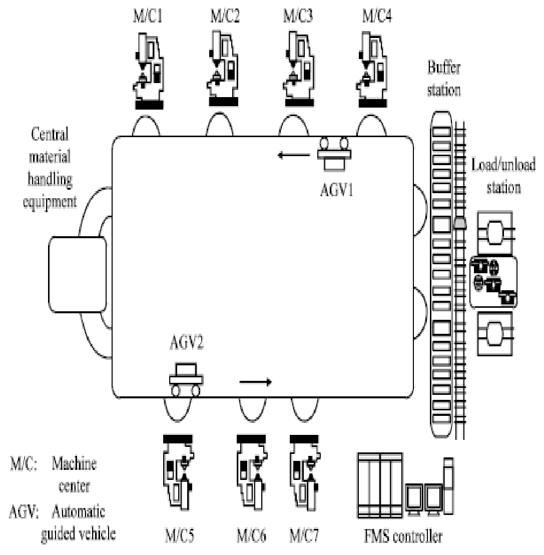
**Abstract— A flexible manufacturing system (FMS) is a manufacturing system in which there is some amount of flexibility that allows the system to react in case of changes, whether predicted or unpredicted. This flexibility is generally considered to fall into two categories. The first category, machine flexibility which enables the work stations to perform different operations simultaneously on a part when there is a sudden change in demand. The second category is routing flexibility which literally expands the work floor by utilizing multiple numbers of machines to perform the operations via routing. Automated Guided Vehicles fall into second category and acts as hosts for the execution of functions which includes picking and dropping, docking and undocking required when routing flexibility comes into play. The objective of the paper is to develop an algorithm and code it in a software language which when executed generates a safe sequence to be followed to avoid deadlock.**

**Key Words— AGVs, Deadlock, FMS, Machine cell, Rerouting.**

## I. INTRODUCTION

Advanced manufacturing systems today have to deal with criteria involving multiple workstations and machines, varying AGVs and customer requirements with time. All these parameters are to be well coordinated and controlled therefore forcing the industries to automate most parts of their work floor in order to achieve un-interrupted production in turn satisfying the customer demands with low production costs. Such application of automated systems in material storage and retrieval systems involves Automated Guided Vehicles (AGVs) which are human less, computer controlled machines which move around the shop floor performing actions like storage and retrieval, picking and dropping, docking and undocking kind of tasks which are linked to the production process [2].

This paper deals with the deadlock avoidance and rerouting of AGVs by developing a strategy, writing algorithm and encoding it in a software language. Deadlock can be defined as a situation where a part or entire system of AGV stalls. When this condition prevails on the work floor, the activities which are crucial to the production also stalls causing a breakdown in the production which is an unfavorable situation. We develop a strategy by analyzing the total number of AGVs (Maximum AGVs), current status (Allocated AGVs), free (Available AGVs) and future requirements (Need) of the different work stations for different tasks to be done by the AGVs. All the above information is stored in a database from where the user, AGVs and a machine cell can access the data required therefore providing ability to create and place order on the work floor for required task to be done voluntarily [7].



With the above information available in the data base execution of algorithm starts by taking all the inputs and follows certain predefined logical steps and mathematical rules written the program which will be viewed further in the flow chart and therefore yields a result which will be in the form of a sequence of processes to be followed in order to avoid deadlock and ensure safe path of AGV started from source to end at the destination. The output generated will be a sequence of processes to be carried out in order to avoid any interruptions in production and deadlock conditions.

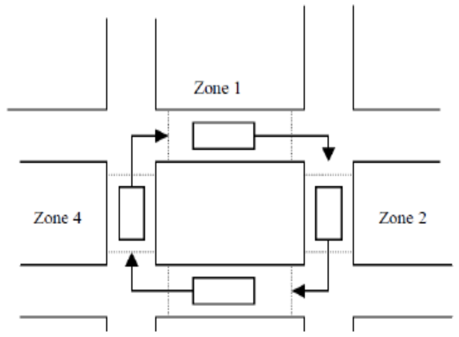
## II. LITERATURE SURVEY

**Deadlock:** A system is said to be in a deadlock if a part or the whole of the system stalls. In the AGVS, resources (zones between control points) are shared among the whole population of vehicles. Each of the vehicles can only occupy one zone at a time[3].

**Mutual exclusion:** At least one AGV must be held in a non-sharable mode; that is only one machine at a time can use the AGV

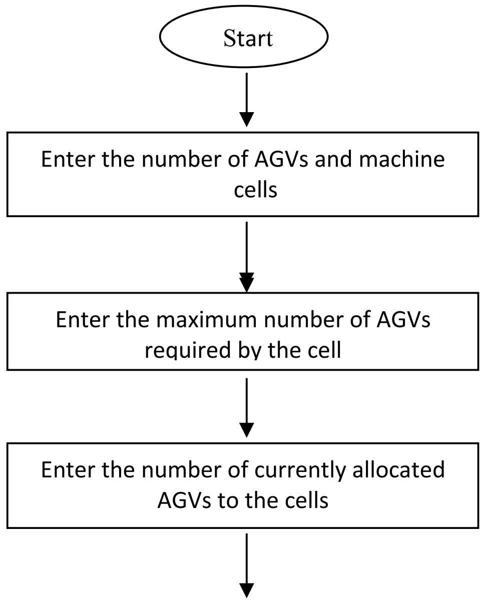
**No preemption:** AGVs cannot be preempted; that is an AGV can make itself available to the machines voluntarily only after the currently allocated stations is completed.

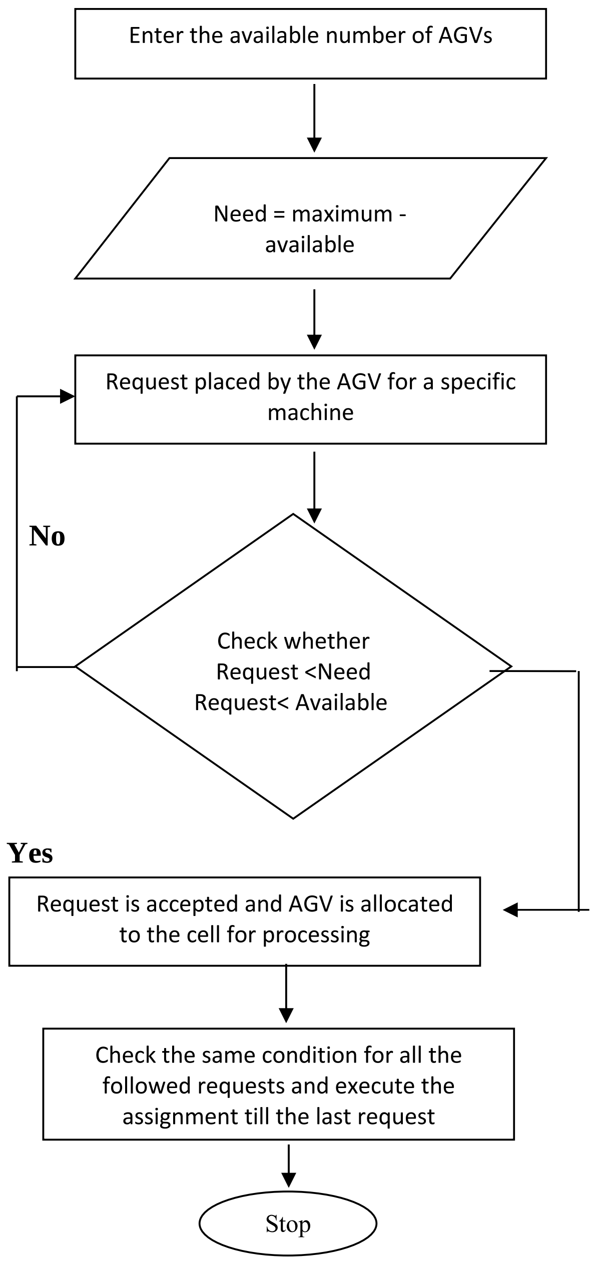
**Circular wait:** There must be a set such that each AGV is waiting for another AGV to finish its task as the other is waiting to execute at the same station which ultimately forms a cycle causing deadlock [12].



## III. ALGORITHM

Execution of algorithm starts when a request is placed in the system by a work station for an AGV to perform the prescribed task. The request is processed by comparing the requested demand (No. of AGVs) to the available AGVs and two possibilities are generated as follows. Firstly when the requested order is above the available then it is denied as it is not possible for the system to allocate more AGVs than available. In the other case if it is below, the algorithm checks for any circular wait condition existing in the queue and then the request is accepted with a condition that AGV is left free at the instant the task has completed as requested by the work station and adds up to the available set. The completed process is terminated from the queue and an update is sent to data base which comprises of the refreshed information about the number of available, allocated and needed sets of AGVs for the other processes. The cycle is repeated for every request made by the machine cell till all the processes in the queue are finished a shown pictorially [5].


Start
Enter the number of AGVs and machine cells
Enter the maximum number of AGVs required by the cell
Enter the number of currently allocated AGVs to the cells


Enter the available number of AGVs
Need = maximum - available
Request placed by the AGV for a specific machine
No
Check whether Request <Need Request< Available
Yes
Request is accepted and AGV is allocated to the cell for processing
Check the same condition for all the followed requests and execute the assignment till the last request
Stop

## IV. MATH

**Safety Algorithm**: The algorithm for finding out whether or not a system is in a safe state. This algorithm can be described as follows.
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 I exist, go to step4.
**Work= work+Allocation**
Finish[i] =true
Go to step2
If Finish[i]==true for all I, then the system is in a safe state.

**Machine-Request Algorithm:** The algorithm which determine if requests can be safely granted.
Request, be the request vector for AGV Ai, if Request[j]==k, then AGV Ai wants k instances of machines type Mj, when a request for machines is made by AGV Ai, the following actions are taken.
If Request≤ Need, go to step2. Otherwise, raise an error condition, since the AGV has exceeded its maximum claim.
If Request≤Available, go to step3. 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 state is safe, the transaction is completed, and AGV Ai is allocated its machines. However, if the new state is unsafe, then Ai must wait for Request, and the old machine-allocation state is restored.

EQUATIONS

**Several instances of a Machine type**:
This defines the instances and state machines in the work station denoted as below.

**Available**: A vector of length m indicates the number of available machines of each type.

**Avail[i][j]=k**

Maximum: An n*m matrix defines the maximum number of machines may needed by the 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:** An n*m matrix indicates the current request of each AGV if Request[i][j] equals k, then AGV Ai is requesting k more instances of machine type Mj.

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

```
a. Finish[i]==false
b. Request≤work
    If no such I exist, go to step4
    Work=work+allocation
    Finish[i]=true
    Go to step2
    If finish[i]==false for somei,0≤i<n, then the system is in a deadlock state.
    Moreover if finish[i]==false then job Ji is deadlocked[7].
```

## V. RESULT

The input parameters are fed into the system which includes the number of AGVs and machine cells on the shop floor, maximum and currently allocated AGVs to the work stations, available numbers of AGVs, the first request initiated by a specific cell to an AGV like factors and are compiled. When the program is run the output obtained is a proper sequence of tasks (requests) to be accomplished by AGVs in order to avoid any chance of occurrence of deadlock. It implies any deviation of performing tasks from the sequence generated by the program leaves the system in an unsafe state.

Depending upon the number and values given as input to the system sometimes a sequence is generated which does not include all the tasks that is an incomplete sequence is obtained, which notifies that immediate process after this sequence may lead to a deadlock. A deadlock avoidance algorithm is awaken when this kind of situations are encountered on the shop floor. The deadlock avoidance algorithm is basically a decision making strategy with a few logical steps which assigns some priority to the requests placed by the machine cells for the AGVs. When the situations like above are aroused the algorithm forces the AGVs to perform the tasks as per the priority which is labeled depending upon the role importance of the task in the production instead of following the sequence obtained.

## VI. CONCLUSION

The banker's algorithm which implemented in the resolving of problems related to routing of AGVs which frequently undergo deadlock like situations generated the solution which is found to be very effective in the form of a sequence of jobs to be done by the AGVs so as to avoid deadlock condition prevailing on the shop floor. The algorithm written and coded in a computer language C++ is capable of producing results for any number of inputs given which enables the user to obtain the safe sequence with less computational efforts. 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.

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 the work floor [8].

## VII. FUTURE SCOPE

- Application of INTRANET concept where every individual AGV becomes knowledgeable of every another AGV, thus no deadlock can prevail as their sources and destinations will be predefined.

- The program written in software language when interfaced with simulation soft ware like AUTOMOD and FLEXSIM animation can be viewed of what is going in virtual environment without going for real time application.

- ➢ Implementing Graphical Information System (GIS) as a vision system so that the AGV can re-route itself when senses any obstacle in its path.

- ➢ With the installation of Artificial Intelligence (AI) visual and rerouting ability can be provided so that deadlock can be overcome even if occurs in any circumstances

## REFERENCES

[1] ANDERSON, M. (1985) AGV system simulation-a planning tool for AGV route layout. Proceedings of the 3rd International Conference on AGV systems, 291-296.

[2] International Journal of Production Research ISSN 0020±7543 print/ISSN 1366±588X online # 2002 Taylor & Francis Ltd

[3] Ram Pratap Yadav *et al* / VSRD International Journal of Mechanical, Auto. & Prod. Engg. Vol. 2 (7), 2012

[4] DUTT, S. (1991) Guided vehicle systems - A simulation analysis. M. Sc. Thesis, Department of Industrial and Systems Engineering, Virginia Polytechnic Institute and State University.

[5] Spyros A. Reveliotis. Conflict Resolution in AGV Systems. School of Industrial & Systems Engineering, Georgia Institute of Technology.

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

[7] Chang, W.K., Tanchoco, J.M.A., and Koo, P.H. (2005). Deadlock Prevention in Manufacturing Systems with AGV Systems: Banker's Algorithm Approach. Journal of Manufacturing Science and Engineering,v119, 849-854.

[8] Hyuenbo, Cho, Kumaran, T.K., and Richard, A.Wysk. (1995). Graph Theoretic Deadlock Detection and Resolution for Flexible Manufacturing Systems. IEEE Transactions on Robotics and Automation, v11, n3,413-421.

[9] Stefano Pallottino, Maria Grazia Scutella (1998). Shortest path algorithms in transportation models: Classical and innovative aspects. University of Pisa.

[10] ARAKI, T., TAKAHASHI, T., SUEKANE, M., & KAWAI, M. Flexible AGV system simulator Proceedings of the 5th International Conference on AGV Systems, 77-86.Ram Pratap Yadav et al / VSRD International Journal of Mechanical, Auto. & Prod. Engg. Vol. 2 (7), 2012282

[11]DEWSNUP, M. C. (1995) How to model AGVS using ProModel for Windows. Proceedings of 1995 Winter Simulation Conf., 482-486.

[12]Yeh, M.S., and Yeh, W.C. (1998). Deadlock Prediction and avoidance for zonecontrol AGVS.INT.J.PROD.RES, v36, n10, 2879-2889.