

METHODS FOR IMPLEMENTING AGV PARALLEL SERVER LANE SELECTION RULES IN AUTOMOD

Sanjay S. Upendram
Production Modeling Corporation
Three Parklane Blvd, Suite 910 West
Dearborn, MI 48126

Onur M. Ulgen, Ph.D.
Production Modeling Corporation
&
University of Michigan-Dearborn
4901 Evergreen Road
Dearborn, MI 48128

1. ABSTRACT

The paper provides a detailed modeling methodology for implementing different Automated Guided Vehicle (AGV) parallel server lane selection rules in AutoMOD. The rules that are modeled include Sequential Server, Longest Idle Server, Least Accumulated Service Time, Earliest Expected Completion Time and First Available Buffer.

2. INTRODUCTION

Automated Guided Vehicle systems constitute one of the most flexible and complex material handling systems available today. The advancement in material handling technology and the improved guidance and communication systems have increased the flexibility and control of AGV systems. AGVs today can make intelligent decisions regarding scheduling, routing and traffic control as they flow through the system.

Typically, AGVs have been used in applications involving non-continuous flow of materials. Examples include delivering components to workstations, transferring work-in-process (WIP) parts in and out of buffers, and traditional warehousing applications. Recently, they are also being used to carry the full production flow as in assembly applications. In such systems, AGVs serve as moving platforms that transport the product to be assembled from one assembly area to another throughout the assembly process.

Parallel workstation assembly systems appear to have many benefits over traditional, in-line assembly systems. The parallel configuration, while expensive, does offer significant flexibility for assembling a variable product line. The more flexibly the system is designed, the more important it is to route the jobs to the appropriate parallel workstation based on the dynamic conditions in the system. The lane selection rules (routing rules) have a significant impact on the characteristics of flow of the jobs into and out of the parallel workstations, the blocking and starving times of jobs in the system, and the overall throughput of the system. Factors which may be considered in choosing a lane selection rule for a parallel assembly system include:

- a) Mean service or processing time of each server across the parallel lanes.
- b) Variance of the processing time.
- c) Indexing time of AGVs between one communication point to another point.
- d) Downtimes of servers.
- e) Sequence requirements of parts through the system.

The complexity of routing rules in AGV systems and the high costs involved in implementing such systems have made the use of simulation models imperative to test design, implementation, and operation of the AGV systems. One significant benefit to using simulation is the ability to test the applicability, flexibility, and robustness of the proposed and existing designs and control rules. Simulation models of AGVs are also used as a powerful decision tool in the guide path design, determination of optimal buffer sizes and locations, determination of number of AGVs required, implementing optimal server selection rules, and system throughput verification.

3. DESCRIPTION OF A PARALLEL ASSEMBLY SYSTEM

The assembly area contains parallel lanes to which jobs are delivered using AGVs as the material handling equipment (see Figure 1). When the jobs reach the entry point (Decision point), they need to determine which lane to choose for processing. The control system keeps track of past production as well as the system's current status. This information is very important for routing the next incoming job to the appropriate workstation. The job is assigned to a particular lane based on the *input lane selection rule* and travels to the upstream stop point or buffer in the respective lane. If the workstation is free, the job travels directly to the workstation for processing. If the station is currently occupied, the job waits in the upstream buffer point until the

workstation is available. After the appropriate amount of work is completed, it goes to its available downstream buffer point and proceeds to the next assembly system based on *output rules* such as FIFO or oldest job first or hot job first or any other redefined rule. The five input lane selection rules discussed in this paper are as follows:

- 1) *Sequential Server*
- 2) *Longest Idle Server*
- 3) *Least Accumulated Service Time*
- 4) *Earliest Expected Completion Time*
- 5) *First Available Buffer*

The input lane selection rules not discussed in this paper include Least Number of Jobs Processed, Most Number of Jobs Processed and Dedicated Lane Assignment rules.

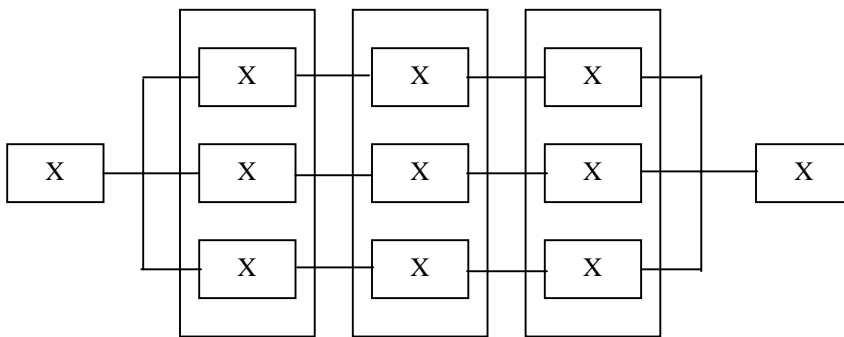


Figure 1. Parallel-work-station assembly system with 3 parallel stations.

4. CONTROL MECHANISMS FOR AUTOMATED GUIDED VEHICLES IN AUTOMOD

Before we discuss the various lane selection rules, it is important to discuss the modeling of AGVs in AutoMOD. AGVs in AutoMod can be modeled by one of two methods, namely: AGV as the Controller, and Load On-Board as the Controller.

4.1 AGV as the Controller

In this approach, the AGV is given control of its travel through the system. This can be accomplished by using the AutoMOD built-in features such as Work List, Park List, Load List, Vehicle List, and Search List. Modeling complex scheduling rules using these built-in features could prove to be cumbersome and also require an in-depth

knowledge of these constructs. This can be avoided by an the alternate AGV control method, the Load On-Board Approach.

4.2 Load On-Board as the Controller

In this approach, a load resides on the AGV and the AGV is controlled by commands issued to the load. This type of modeling simplifies the AGV control in terms of scheduling, routing and traffic control. The load executes the process procedures as it moves from one process to another.

In this paper, we recommend modeling of the various AGV parallel server selection rules using the "Load On-Board as the Controller" when any complex logic, routing or scheduling of AGVs are involved. Listed below are the general steps of this approach.

- 1) Define the AGV movement system.
- 2) Draw the Path.
- 3) Add Control points to the Chain.
- 4) Define the "Named List" giving the initial locations of the AGVs in terms of their Control Point.
- 5) Connect the Control point of Origin to a Process in the Process Connections.
- 6) Clone Loads equal in number to the number of AGVs to this Process.
- 7) The Loads automatically reside on or move onto the AGV since the AGV is the territory in the AGV movement system.
- 8) Commands given to the Load On-Board control the routing of the AGV through the system.
- 9) The Load residing on the AGV can essentially execute any of the actions available to a Load, e.g., wait on Order Lists, move into Queues, move from one Process to another, execute a Procedure, etc.

5. MODELING OF AGV PARALLEL SERVER SELECTION RULES

The following section provides a brief discussion of each of the selection rules followed by a modeling approach for its implementation. Detailed AutoMOD Procedure code for the rules are given in the Appendix. The modeling approach taken in implementing each of the rules is a two-step approach. In the first step, the logic at the decision point is described while in the second step, the logic at the workstation-server is described.

5.1 Sequential Server Rule

This is the simplest of all the rules and according to this rule, jobs select the parallel servers based on a fixed sequence. The sequence could either be round robin or any other sequence specific to the needs of the system, as determined by the decision makers.

The two-step approach is as follows:

I. Logic at the Decision Point

- i) The destination of the AGV is set in a Load Attribute of the Load residing on-board of the AGV. This destination is set using the "NextOf" distribution which contains the sequence of route.
- ii) Send the AGV to the server based on the recorded attribute.

II. Logic at the parallel workstation server

- i) Use the server for the service time.
- ii) Send the AGV to the next Process.

5.2 Longest Idle Server Rule

According to this rule, a job that finds more than one server idle, selects that server who has been idle for the longest amount of time. This rule essentially attempts to minimize the idle time of each server.

The two-step approach is as follows:

I. Logic at the Decision point

- i) Identify all the free servers
- ii) Check the amount of time each server has been free by taking the difference between the current time (ac) and the time when the previous AGV left that particular server (which is recorded in a Variable at the server workstation).
- iii) Compare the idle times for all the free servers.
- iv) Record the longest idle servers index in a Load attribute.
- v) Send the AGV to the Server based on the recorded attribute.

II. Logic at the parallel workstation server

- i) Use the server for the service time.
- ii) After the service time record the time when the server is freed into a Variable.
- iii) Send to the next Process.

5.3 Earliest Expected Completion Time Rule

According to this rule a job that finds all the workstations busy selects that server who is expected to complete its job earliest. The expected completion time of each server is the sum of the beginning times of service and the corresponding mean service time, of the previous job assigned to it. This rule is used when all the servers are busy and when some or all the corresponding buffer locations are empty. This rule attempts to be farsighted by assigning jobs to earliest expected free server and thereby decreasing the blocking upstream.

The two-step approach is as follows:

I. Logic at the Decision point

- i) Identify the servers whose buffer locations are empty.
- ii) Check the expected completion time for each of these servers. The expected completion time is equal to the sum of the time when the previous AGV entered the station and the mean service time of that particular server. This value is stored in a Variable.
- iii) Compare the expected completion times for the servers.
- iv) Record the index of the server with the earliest completion time in a Load Attribute.
- v) Send the AGV to the server based on the recorded attribute.

II. Logic at the parallel workstation server

- i) As the AGV enters the station, calculate the expected completion time for that server and record it in a variable.
- ii) Use the server for the service time.
- iii) Send to the next Process.

5.4 Least Accumulated Service Time Rule

According to this rule jobs select the server whose accumulated service time is the least among all the parallel servers. Even though the rule does not attempt to maximize the flow of jobs through the system, it is still favored by the servers, because at any given time, it attempts to distribute the work time equally among all the parallel servers, disregarding the actual status of the servers and buffer locations.

The two-step approach is as follows:

I. Logic at the Decision Point

- i) Compare the least accumulated service time for each of the servers.

- ii) Record the index of the server with the least accumulated service time in a Load Attribute.
- iii) Send the AGV to the server based on this recorded Attribute.

II. Logic at the parallel workstation server

- i) As the load on the AGV gets the resource, record the start clock time (ac) of the service in a variable.
- ii) Wait for the service time.
- iii) Calculate the amount of the service time by subtracting the start clock time from the current clock time(ac).
- iv) Accumulate the service time.
- v) Send the AGV to the next Process.

5.5 First Available Buffer Rule

According to this rule jobs select the lane which has the first available buffer location. This rule is applicable when all the parallel servers are busy and the corresponding upstream buffers are occupied. The rule attempts to minimize the blocking time by assigning jobs to lanes as soon as one buffer location is available.

The two-step approach is as follows

I. Logic at the Decision Point

- i) Check the total capacity of buffers and if full wait until space is available before any one of the servers.(Note: In AutoMOD, a Load trying to increment a Counter with a maximum capacity emulates a Wait Until condition. Thus in the above case a Counter is used whose capacity is set to the sum of the number of parallel servers and the capacity of the upstream buffer).
- ii) Check sequentially each lane for the available buffer location and record the index of that lane in a load attribute.
- iii) Send the AGV to the lane based on the recorded attribute.

II. Logic at the parallel workstation server

- i) Use the server for service time.
- ii) When the AGV is about to leave the station decrement the Counters which keep track of the total available space in the workstation and also the space available in each lane.
- iii) Send the AGV to the next Process.

6. CONCLUSION

In this paper we suggest modeling approaches for implementing some of the more important AGV parallel server selection rules in AutoMOD. Five input lane selection rules are discussed using a parallel assembly system with 3 parallel workstations. In addition, two approaches are suggested to model the AGV, namely: AGV as the Controller and the Load On-Board as the Controller. Load On-Board approach is recommended for the complex routing and decision making.

Advances in the material handling industry and microprocessor technology have equipped the manufacturing industry with intelligent material handlers that can execute sophisticated rules very easily. The drive towards a lean and agile manufacturing environment has resulted in the increased use of such equipment. AutoMOD provides an excellent environment to model these various material handlers. A useful venture for ASI would be to incorporate these rules as built-in features in the AGV template which would further enhance the capability of the software.

REFERENCES

- Garry A. Koff (1987), " Automatic Guided Vehicle Systems: Applications, Controls and Planning," *Material Flow*, 4, 3-16.
- Onur M.Ulgen and Pankaj Kedia (1990), "Using Simulation in the Design of a Cellular Assembly Plant with Automatic Guided Vehicles," *Proceedings of the 1990 Winter Simulation Conference*, Osman Balci, Randall P.Sadowski and Richard E.Nance, 683-691.

APPENDIX

The appendix contains AutoMOD II code to model the various lane selection rules discussed in section 5. While developing the code the authors have tried to follow the nomenclature standards of appending a suffix to each entity's name. The first letters of the suffix identifies the basic AutoMOD entity such as P for Process, R for Resource , C for Counter , V for variable, LA for Load Attribute etc. followed by a second letter which identifies the type of the entity such as I for Integer, R for Real, etc. For example, VR_MeanServiceTime represents a Real Variable named MeanServiceTime.

A.1 Procedure Code for the Sequential Lane Selection Rule

The following code is used to model the Sequential Lane Selection rule. When the AGV arrives at the decision control point, which is connected to the Process P_Decsn, a load attribute of type integer (LAI_NextLane) is set to the load on-board. This Attribute is set, based on a predefined sequence using the "nextof" distribution. The load is then sent to the corresponding server Process, P_Server, which is arrayed.

```

/*****
begin P_Decsn arriving procedure
    /* This process controls the AGV selection rules. This
    process is connected to the decision control point. */
    set LAI_NextLane to nextof ( 1, 2, 3)
    send to P_Server( LAI_NextLane )
end
*****/

begin P_Server arriving procedure
    /* This process is arrayed based on the number of parallel
    lanes present in the workstation and contains the Servers
    (Resources) */
    get R_Server(procindex)
    wait for VR_ServiceTime(procindex) sec
    free R_server(procindex)
    send to P_End
end
*****/

```

A.2 Procedure Code for the Longest Idle Server Rule

The following code is used to model the Longest Idle Server Selection rule.

```

/*****
begin P_Decsn arriving procedure

/* The following code checks to see if more than one lane is empty. The
current number of parts present in the workstation and in each lane are
tracked using a counter */

    if C_PartsTotal current = 0 or
       C_Lane(1) current = 0 or
       C_Lane(2) current = 0 or
       C_Lane(3) current = 0 then

```

```
/* This code sequentially verifies the amount of time for which each of the
empty stations has been idle and records the index of the longest idle station
in a load attribute LAI_NextLane. The job is then sent to that particular
server. */
```

```
    set VI_Index
    set VR_LongestIdleTime = 0
    While VI_Index <= 3 do
        /* 3 - the Number of parallel servers*/
    begin
        if C_Lane(VI_Index ) current = 0 then
            begin
                set VR_LongIdleTime(VI_Index ) =
                    ac-VR_ServerIdleTime(VI_Index )
                if VR-LongIdleTime(VI_Index ) >
                    VR_LongestIdleTime then
                    begin
                        set VR_LongestIdleTime to
                            VR_LongIdleTime(VI_Index)
                        set LAI_NextLane to VI-Index
                    end
                end
            end
        increment VI_Index by 1
    end
end
increment C_PartsTotal by 1
increment C_Lane(LAI_NextLane) by 1
send to P_Server( LAI_NextLane )
```

```
end
```

```
/******
```

```
/* Process P_Server is arrayed based on the number of parallel lanes present
in the workstation and contains the Servers (Resources). After the job is
processed the beginning of idle time is recorded in VR_ServerIdleTime().*/
```

```
begin P-Server arriving procedure
```

```
    /* This process is arrayed based on the number of parallel
lanes present in the workstation and contains the Servers
(Resources) */
    get R_Server(procindex)
    wait for VR_ServiceTime(procindex) sec
    free R_server(procindex)
    set VR_ServerIdleTime(procindex) to ac
    decrement C_PartsTotal by 1
    decrement C_Lane(procindex) by 1
```

```

        send to P_End
    end
/*****

```

A.3 Procedure Code for the Earliest Expected Completion Time Rule

The following code is used to model Earliest Expected Completion Time Rule.

```

/*****
begin P_Decsn arriving procedure
    /* The following code checks to see if at least one lane (server) is
    busy. The current number of parts present in the workstation and in
    each lane is tracked using a counter */

    if C_PartsTotal current < 6 or
        /* 6 = 3 servers + 3 input buffer locations */
        C_Lane(1) current <> 0 or
        C_Lane(2) current <> 0 or
        C_Lane(3) current <> 0 then

        /* This code sequentially verifies the expected completion time
        of all the busy servers who have an input buffer location
        available and records the index of the first expected server to
        complete, in a load attribute LAI_NextLane. The job is then
        sent to that particular server.*/

        begin
            set VI_Index = 1
            set VR_FirstExptd = 999999
            While VI_Index <= 3 do
                begin
                    if C_Lane(VI_Index ) current < 2 then
                        /* 2 = 1 server location + 1 input buffer location */
                        begin
                            set VR_ExptdServer(VI_Index ) =
                                VR_ExptdServerCmplt(VI_Index
                            if VR_ExptdServer (VI_Index) <
                                VR_FirstExptd then
                                    begin
                                        set VR_FirstExptd =
                                            VR_ExptdServer(VI_Index)
                                        set LAI_NextLane to VI_Index
                                    end
                                end
                            end
                        end
                    end
                end
            end
        end
end

```

```

        increment VI_Index by 1
    end
end
increment C_PartsTotal by 1
increment C_Lane(LAI_NextLane) by 1
send to P_Server( LAI_NextLane )
end
/*****/
/* Process P_Server is arrayed based on the number of parallel lanes
present in the workstation and contains the Servers (Resources). Just before
the server begins processing the job, the expected completion time of that
server is recorded in the variable VR_ExptdServerCmplt().*/

begin P_Server arriving procedure
    /* This process is arrayed based on the number of parallel
lanes present in the workstation and contains the Servers
(Resources) */
    set VR_ExptdServerCmplt(procindex) to
        ac + VR_MeanServiceTime(procindex)
    get R_Server(procindex)
    wait for VR_ServiceTime(procindex) sec
    free R_server(procindex)
    decrement C_PartsTotal by 1
    decrement C_Lane(procindex) by 1
    send to P_End
end
/*****/

```

A.4 Procedure Code for the Least Accumulated Service Time Rule

The following code is used to model the Least Accumulated Service Time selection rule.

```

/*****/
begin P_Dechn arriving procedure

/* This code sequentially verifies the accumulated service time for each
of the servers and records the index of the server who worked least in a
load attribute LAI_NextLane . The job is then sent to that particular
server.*/

        set VI_Index = 1
        set VR_LeastAccumTime = 999999
        While VI_Index <= 3 do

```

```

/* 3 - the Number of parallel lanes*/
begin
  if VR_AccumTimeServed(VI_Index) <
    VR_LeastAccumTime then
    begin
      set VR_LeastAccumTime =
        VR_AccumTimeServed(VI_Index)
      set LAI_NextLane to VI_Index
      end
      increment VI_Index by 1
    end
  send to P_Server( LAI_NextLane )
end
/*****
/* Process P_Server is arrayed based on the number of parallel lanes
present in the workstation and contains the Servers (Resources). The
service time at the station is noted and accumulated in the Variable
VR_AccumTimeServed().*/

begin P_Server arriving procedure
  get R_Server(procindex)
  set VR_ServiceTimeStart(procindex) to ac
  wait for VR_ServiceTime(procindex) sec
  set VR_TimeServed(procindex) =
    ac - VR_ServiceTimeStart(procindex)
  increment VR_AccumTimeServed(procindex) by
    VR_TimeServed(procindex)
  free R_server(procindex)
  send to P_End
end
/*****

```

A.5 Procedure Code for the First Available Buffer Rule

The following code is used to model the first available buffer selection rule.

```

begin P_Decsn arriving procedure

/* The job executes the following code if all the locations in the workstation
are occupied. The job then waits until it can increment the Counter,
C_PartsTotal, which keeps track of the total jobs in the workstation. Once it
increments this Counter, it indicates that at least one of the servers is empty,
and then it sequentially verifies the buffer availability of each lane and
records the index of the available lane into

```

the Load Attribute LAI_NextLane. The job is then sent to that particular lane. */

```
if C_PartsTotal 6 then
    /* 6 = 3 servers and the 3 buffer locations are busy */
    begin
        increment C_PartsTotal by 1
        set VI_Index = 1
        While VI_Index <= 3 do
            begin
                if C_Lane(VI_Index) < 2 then
                    begin
                        increment
                            C_Lane(VI_Index) by 1
                        set LAI_NextLane to VI_Index
                    end
                end
            increment VI_Index by 1
        end
    end
    send to P_Server( LAI_NextLane )
end
/*****/
/* Process P_Server is arrayed based on the number of parallel lanes
present in the workstation and contains the Servers (Resources).*/

begin P_Server arriving procedure
    get R_Server(procindex)
    wait for VR_ServiceTime(procindex) sec
    free R_server(procindex)
    decrement C_PartsTotal by 1
    decrement C_Lane(procindex) by 1
    send to P_End
end
/*****/
```