# Distributed Control of Job-shop Systems via Edge Reversal Dynamics for Automated Guided Vehicles

Omar Lengerke, Hernán González Acuña
Universidad Autónoma de Bucaramanga, UNAB
Mechatronics and Control Research Group
Bucaramanga, Santander, Colombia
{olengerke, hgonzalez3}@unab.edu.co

Max Suell Dutra[1], Felipe França[2]
Felix Mora Camino[3]
[12]COPPE UFRJ, Brazil
[3]ENAC, MAIAA Laboratory Toulouse, France
[1]max@mecanica.coppe.ufrj.br, [2]felipe@cos.ufrj.br,
[3]felix.mora@enac.fr

***Abstract*—Flexible Manufacturing Systems (FMS), in which the use of Automatically Guided Vehicles (AGVs) is typical, are a growing trend in many industrial scenarios. A novel, distributed, algorithmic approach to the execution control of activities (work-center oriented) is introduced in this paper, as is, in an integrated way, transportation (AGV oriented) scheduling. The relationship between *jobs*, modeled as *processes*, and *work centers*, modeled as *resources*, and *sinks* defines an undirected graph *G* representing a target Job-shop system. Analogously, the transportation performed by AGVs, also modeled as processes, and their corresponding physical paths, modeled as resources, can also be seen as a dual Job-shop problem. The new approach is based on the Scheduling by Edge Reversal (SER) graph dynamics which, from an initial acyclic orientation over edges, that can be defined via traditional and/or efficient heuristics, let jobs and AGVs proceed in a deadlock-and-starvation-free fashion without the need for any central coordination.**

***Keywords-Job-shop; Distributed algorithm; Flexible Manufacturing System; Graph dynamics; Scheduling by Edge Reversal.***

## I. INTRODUCTION

With the current interest in Flexible Manufacturing System (FMS), there is a growing need for scalable Job-shop solutions. This article presents a new approach to the distributed representation and control of Job-shop systems. The novel approach consists of mapping a Job-shop system into an undirected graph $G = (N, E)$, where $N = \{1, \dots, n\}$ is the set of activities and $E$ is defined as follows: if $R_i$ is the set of resources used by node $i$ in order to operate, an edge $(i, j) \in E$ exists whenever $SR_i \cap SR_j \neq \emptyset$, that is, activities *i* and *j*, share at least one atomic resource.

Next, an initial acyclic orientation *w* is defined over *E*. As shown in the following sections, this setup can be produced via well-known heuristic criteria, such as Earliest Due Date (EDD), Shortest Processing Time (SPT) and Priority (P). The Scheduling by Edge Reversal (SER) dynamics is then applied over *G*, where activities having all of its edges oriented to themselves have the right to operate upon shared resources and then reverse all associated edges, becoming source nodes in a new acyclic orientation *w'*. This ensures that neighboring activities in the system cannot operate simultaneously upon shared resources. In this context, SER acts as a decentralized control mechanism, ensuring mutual exclusion, coordinating all planned activities, regardless of whether they are concurrent or sequential. Besides, the proposed algorithm takes into consideration transport times, integrating transport and activity schedules, and also providing scalable solutions. In addition, it produces optimal minimum make-spam solutions comparable to traditional methods, while creating a deadlock-and-starvation-free system by construction.

SER is our subject in Section II. The two sections that follow (Section III and IV) are devoted to contextualizing the Job-shop and dispatching problem into the FMS domain. Sections V and VI discuss the construction of the proposed algorithm, and show the effective use of SER for distributed control of Job-shop systems, as well as the final conclusions.

## II. SCHEDULING BY EDGE REVERSAL

In order to implement a distributed scheduling algorithm for decentralized control of Job-shop systems employed throughout, we decided to use a scheduling scheme which ensures by construction a deadlock-and-starvation-free system. The adopted approach is based on the algorithm presented in [1][2][3] to ensure mutual exclusion on distributed asynchronous systems, namely Scheduling by Edge Reversal (SER). In this context, SER is a simple and powerful distributed algorithm, originally conceived to support Distributed Systems under heavy load condition, when processors are constantly demanding access to all resources that they use.

Important SER properties, and the NP-completeness of the problem of finding optimal concurrency amounts provided by the SER dynamics over a given distributed system, are established in [3]. SER works as follows: (*i*) the target distributed system is described by an undirected graph $G = (N, E)$, where $N = \{1, \dots, n\}$ is the set of processing nodes and $E$ is defined as follows: if $SR_i$ is the set of resources used by node $i$ in order to operate, an edge

 ISBN: 978-1-61208-224-0 

$(i,j) \in E$ exists whenever $SR_i \cap SR_j \neq \emptyset$, that is, nodes $i$ and $j$ share at least one atomic resource; (*ii*) an initial acyclic orientation $w$ is defined over $E$; (*iii*) all, and only sink nodes in $w$, i.e., nodes having all of its edges oriented to themselves, have the right to operate upon shared resources and then reverse all associated edges, becoming source nodes in a new acyclic orientation $w'$. This ensures that neighboring nodes in the target distributed system cannot operate simultaneously upon atomic shared resources. SER is the graph dynamics defined by the endless iteration of (*iii*) over $G$ (Figure 1).

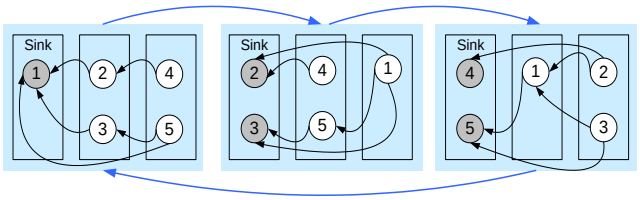


Figure 1. SER Operation

Considering $G$ finite and, consequently, a finite number of possible acyclic orientations over $G$, eventually a repetition, i.e., a *period* of length $l$, will occur. An interesting property of SER lies in the fact that, inside any given period, each node operates, i.e., becomes a sink, the same number $q$ of times, ensuring "fairness", in the long run operation, among all processing elements of $G$ [3].

Many works devised a powerful family of SER-based distributed algorithms in different contexts [4]: presented how a SER dynamic can be used for sharing resources at non-uniform rates, allowing different processor priorities, breaking the symmetry rule that every processor should become a sink the same number q of times in a given period;[4] illustrates how to perform an optimal mapping of processors or machines in neighborhood-constrained systems and [5] demonstrated a novel algorithm named Scheduling by Edge Reversal with Hibernation (SERH), a distributed algorithm for scheduling of atomic shared resources in the context of dynamic load reconfiguration, where processors or nodes are able to relinquish the right of execution, allowing the reconfiguration of the whole of the distributed system.

Due to its simplicity, SER is being currently applied to different domains. Among them, we could list: (*i*) industrial plants, where process are jobs and resources are machines, Automated Guided Vehicles (AGV), consumption, etc.; (*ii*) computational grid scheduling, where processes are computing jobs, and resources are *CPUs*, data; disk space and network links are grid data movement, where applications geographically distribute every datum to be used by a distributed computation.

## III. FLEXIBLE MANUFACTURING SYSTEM AND JOB-SHOP SCHEDULING

Our interest in distributed Job-shop algorithms comes from the increasing interest in Flexible Manufacturing System (FMS) [6][7][8]. Flexibility measures the ability to adapt to a wide range of possible environments. The term FMS refers to a class of highly automated systems that consist of set of computer-numerically-controlled (CNC) machine tools and supporting workstations that are connected by an automated material handling system. The resulting system is controlled by a central computer that coordinates machine tools, material handling, and parts [9]. Especially, we consider the FMS composed by several Flexible Manufacturing Modules (FMM) or Flexible Manufacturing Cells (FMC), and, at least, one Material Handling System (MHS) consisting of one or more Automatic Guided Vehicles (AGVs). FMS scheduling is significantly different from traditional Job-shops where the human being is concerned. Deadlock situations may occur in FMS due to jobs in a circular waiting of resources (robots, buffers or paths). Consequently deadlocked situations have been identified as one of the most critical problems in the scheduling and control of FMSs.

In multi-operation shops, jobs often have different routes. More specifically, in a Job-shop, each part has its own route. Such environment is known as a generalization of a flow shop (a flow shop is a Job-shop in which each and every job has the same route). The simplest Job-shop models assume that a job may be processed on a particular machine at most once on its route through the system. In others a job may visit a given machine several times on its route through the system.

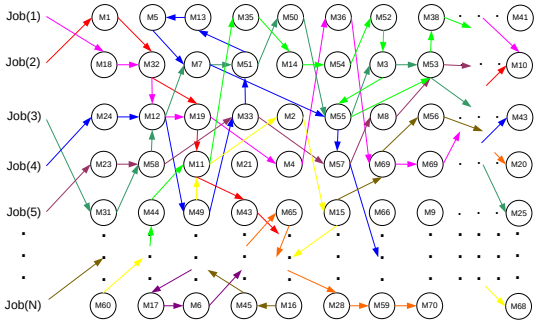


Figure 2. Job-shop Problem

These shops are said to be subject to re-circulation, which increases the complexity of the model considerably, besides the NP-completeness of the Job-shop problem [10].

In our formulation of the Job-shop problem, we assume that there are many jobs on each route. In practice, routes may correspond to various production processes, or to various types of products manufactured in a factory. In that case, the jobs may correspond to parts or lots, and there will indeed be many such jobs for each route. A generalization of the Job-shop is the flexible system with work centers that

 ISBN: 978-1-61208-224-0 

have multiple machines in parallel. From a combinatorial point of view, the flexible Job-shop with re-circulation is one of the most complex machine environments. It is a very common setting in the semiconductor industry

In the Job-shop scheduling problem, a set $J$ of $n$ jobs $J_1, J_2, \dots, J_n$, has to be processed on a set $M$ of $m$ different machines $M_1, M_2, \dots, M_n$. Each job $J_j$ consists of a sequence of $m_j$ operations $O_{j1}, O_{j2}, \dots, Ojm_j$, that must be scheduled in this order (Figure 2). Moreover, each operation needs to be processed only on a specific machine among the $m$ available ones. Pre-emption is not allowed and machines can handle at most one operation at a time. Operation $O_{jk}$ has a fixed processing time $p_{jk}$. The objective is to find an operating sequence for each machine to minimize the make-span $C_{max} = max_{j=1,n}, C_j$ , where $C_j$ denotes the completion time of the last operation of job $J_j (j = 1, \dots, n)$ [11]. Operations of the jobs in a Job-shop have to be scheduled to minimize one or more objectives, such as the make-span $C_{max}$ or the number of late jobs.

| | Operation 1 | | Operation 2 | | Operation 3 | |
|---|---|---|---|---|---|---|
| | Proc. time | Mach. number | Proc. time | Mach. number | Proc. time | Mach. number |
| $J_1$ | 3 | $M_A$ | 1 | $M_B$ | 5 | $M_C$ |
| $J_2$ | 9 | $M_C$ | 10 | $M_B$ | 3 | $M_A$ |
| $J_3$ | 10 | $M_B$ | 8 | $M_C$ | 6 | $M_A$ |

TABLE I. SAMPLE SCHEDULING PROBLEM (Liao and You (1992))

| Step $t$ | $A_t$ | $e_k$ | $m^*$ | $k$ |
|---|---|---|---|---|
| 1 | $O_{11}O_{21}O_{31}$ | 0 0 0 | B | $O_{31}$ |
| 2 | $O_{11}O_{21}O_{32}$ | 0 0 10 | C | $O_{21}$ |
| 3 | $O_{11}O_{22}O_{32}$ | 0 10 10 | A | $O_{11}$ |
| 4 | $O_{12}O_{22}O_{32}$ | 10 10 10 | B | $O_{22}$ |
| 5 | $O_{12}O_{23}O_{32}$ | 20 20 10 | C | $O_{32}$ |
| 6 | $O_{12}O_{23}O_{33}$ | 20 20 18 | A | $O_{33}$ |
| 7 | $O_{12}O_{23}$ | 20 24 24 | B | $O_{12}$ |
| 8 | $O_{13}O_{23}$ | 21 24 - | C | $O_{13}$ |
| 9 | $O_{23}$ | 26 26 - 27 | A | $O_{23}$ |

TABLE II. Construction of Schedule for Example TABLE I

We present a procedure that will allow the generation of as many non-delay schedules as desired [9]. Basically, we construct a schedule by scheduling one operation at a time using the following algorithm: *(i) Initialization.* Let stage *t = 1*, $S_1 = 0$ (where, $S_t$ is the partial schedule of (*t-1*) scheduled operations. $A_1$ contains the first operation of each ready job (where, $A_t$ is the set of operations available to be scheduled at stage *t*, that is, all predecessor operations are in $S_t$). *(ii) Selection*. Find $e^* = min_k \in A_t e_k$ (where, $e_k$ is the earliest time that operation $k \in A_t$ can be scheduled, that is, predecessors are completed and the needed machine is available). If several $e^*$ exist, the algorithm chooses it arbitrarily. Let $m^*$ be the machine needed by $e^*$. Choose any $k \in A_t$ that requires $m^*$ and has $e_k = e^*$. *(iii) Increment.* Add the selected operation k to $S_t$ to create $S_{t+1}$. Remove $k$ from $A_t$ and add the next operation for its job unless that job is completed; this creates $A_{t+1}$. Set *t = t + 1*. If *t = MJ* stop; otherwise go to (i). As an illustration, consider the following Job-shop problem (TABLE I), presented by [12]. The process continues until all 9 operations are assigned. Steps are summarized in TABLE II., and this new algorithm also produces the best known make-span of 27.

## IV. DISPATCHING RULES

Dispatching rules have received much attention from researchers over the past decades [13][14][15]. In general, whenever a machine is freed, a job with the highest priority in the processing queue is selected to be run on a machine or work center.

Dispatching is the job selection process from a queue, its immediate setup and processing, when a processor becomes available. Simple dispatching rules are often used in shop scheduling and a list of the more popular ones follows: (*i*) Shortest Processing Time (SPT): Highest priority is given to the waiting operation with the shortest imminent operation time. Processing time $(p_{ij})$ represents the time job *j* has to spend on machine *i*. Subscript *i* is omitted if the processing time of job *j* does not depend on the machine or if it only needs processing on one machine. If there are a number of identical jobs that all need a processing time $p_i$ on one machine, then we refer to this set of jobs as items of type *j*.

The production rate of type *j* items is denoted by $Q_j = \frac{1}{p_j}$ (number of items per unit time). (*ii*) Longest Processing Time (LPT): Highest priority is given to the waiting operation with the longest imminent operation time. (*iii*) Earliest Due date (EDD). Select a job with minimum processing time. The due date $d_j$ of job *j* represents the committed shipping or completion date (the date the job is promised to the customer).

Completion of a job after its due date is allowed, but a penalty is then incurred. When the due date absolutely must be met, it is referred to as a deadline. (*iv*) Most Work Remaining (MWKR): Highest priority is given to the waiting operation associated with the job having the most total processing time remaining to be done. (*v*) Least Work Remaining (LWKR): Highest priority is given to the waiting operation associated with the job having the least amount of total processing time remaining to be done. (*vi*) Total Work (TWORK): Highest priority is given to the job with the least total processing requirement on all operations. (*vii*) First In First Out (FIFO): Highest priority is given to the waiting operation that arrived at the queue first. (*viii*) Last In First Out (LIFO): Highest priority is given to the waiting operation that arrived at the queue last. (*ix*) RANDOM (Random): Select a job *au hazard*.

 ISBN: 978-1-61208-224-0 

## V. SER ON JOB-SHOP SYSTEMS FOR ROUTING PLANNING OF AGVS

AGV routing planning is an important problem in the transportation, distribution and logistics fields. Route is the customary series of stops during a trip (programming of a succession of procedures). Computing the firing sequence of transitions which will yield an optimal result and also avoid deadlocks which might be present is important to real-time control of the modeled system. If an FMS is modeled, the routes planning of AGVs using SER, an optimal firing sequence is an optimal schedule for the system. Hence a method to find an optimal firing sequence of transitions is beneficial to both SER and FMS scheduling. A perpetual deadlock can happen in FMS due to a number of works which are expected to move resources to each other. Therefore, a model that can handle such complex systems is necessary. Several works conducted these analysis types using different methods such as Petri networks [16][17][18], but most of these methods have limitations when there are several types of tasks or activities and large quantity of machinery and do not solve the problem of routing and perpetual deadlock.

### A. Definition

The problem of Job-shop systems can be developed from a scheduling distributed algorithm to control this category of decentralized systems. This is possible through a mapping of the Job-shop target in a graph *G = (N,E)* where each element of *N* is one of the planned activities, with pre-established time, to be implemented in exclusive mode on a limited set of resources, which access restrictions defined the edges set *E*. It is also shown as an acyclic orientation is performed directly on *E* the basics of criteria such as traditional heuristic EDD, SPT and P. The dynamics of scheduling by edge reversal can then be applied to *G*, acting as a decentralized control mechanism of coordination of the implementation of various activities planned, whether concurrent or sequential. Implementation of SER in such systems is a new concept that provided a description of the form of sharing (AND, OR, XOR, negative, among others) to solve the problem of planning routes of the AGVs.

Binary Operators: For OR sharing operates a single resource *M (machine)* in a process *J (job)* (Figure 3). The resource is released (edge reversal) when the processing time finishes $(p_{ij})$ in each of the process operations (*O*). For AND sharing are illustrated in Figure 4.

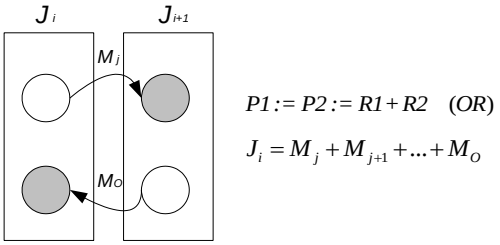


Figure 3. OR Sharing

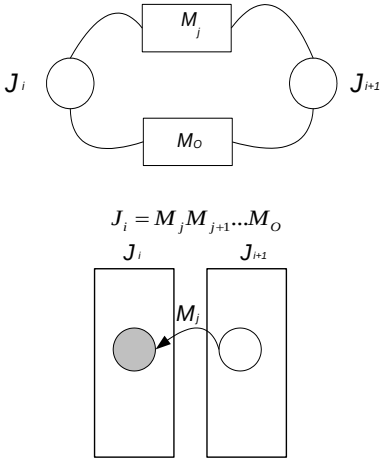


Figure 4. AND Sharing

Example: Applying the concept of SER for the example of TABLE I and represented by Figure 5, which was used the concept of algorithm, to solve problems Job-shop.

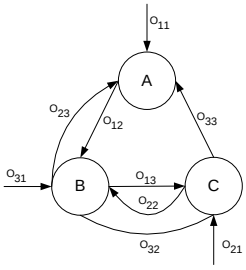


Figure 5. Schematic Diagram of the Problem - Table 1

The three jobs are represented as an expression given by the Equation (1) of the XOR sharing $(M_A \oplus M_B \oplus M_C)$.

$$J_1 = J_2 = J_3 = (M_A \oplus M_B \oplus M_C) = M_A\overline{M_B}\,\overline{M_C} + \overline{M_A}M_B\overline{M_C} + \overline{M_A}\,\overline{M_B}M_C \quad (1)$$

The dynamics of edge reversal corresponding of the system proposed by Equation (1), is shown in Figure 6. In this case, the initial acyclic orientated adopted is determined from criteria or classic dispatching rules (EDD, MWKR, Priority $J_1$ and Random). According to the dynamic and orientation criteria, the first set of operations to be processed

is $A_1 = O_{11}, O_{21}, O_{31}$ where $A_i$ are sinks, and the processing time $(p_{ij})$ is given by $\min t \in A_I$, $\mathrm{t_p} = 3$ and remainder time $(t_f)$ of $O_{21} = 6$ and $O_{31} = 3$, while $O_{11}$ is completed. The following edge reversal is selected operations $A_2 = O_{12}, O_{21}, O_{31}$. The next steps are summarized in Figure 6 and Figure 7, where the make-span is 27. For simplifying:

$$A = M_A\overline{M_B M_C}, B = \overline{M_A}M_B\overline{M_C}\ and\ C = \overline{M_A M_B}M_C.$$

An immediate benefit of this approach is the decentralization of the job control, which enables the distributed control to deal with any modification of the due time (asynchronous algorithm).

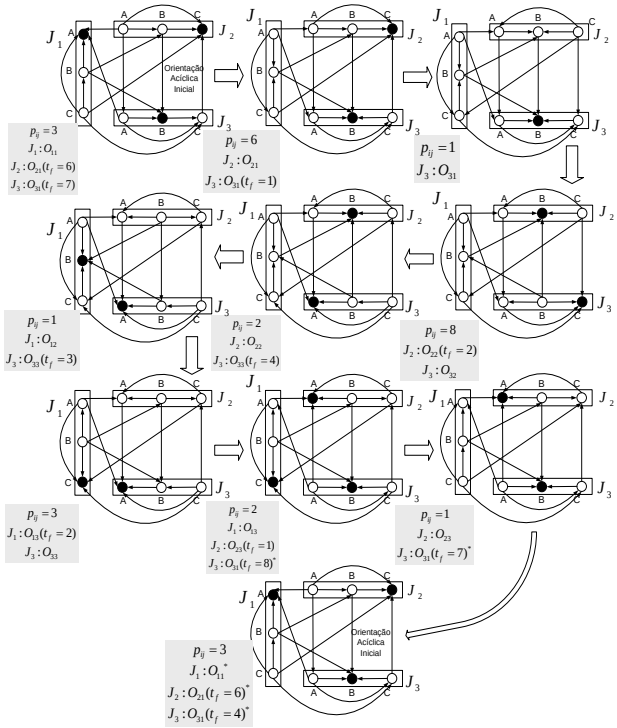


Figure 6 Example of SER

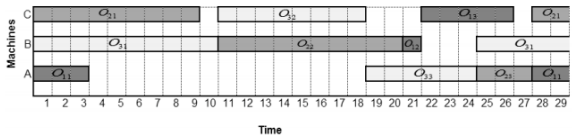


Figure 7 The Generated Schedule with make-spam of 27, with the endless operation of the SER

## VI. SER ON JOB-SHOP SYSTEMS FOR PATH PLANNING OF AGVS

The scheduling by edge reversal can be also used in the creation of a mechanism for dynamic planning programming of paths, allowing traffic concurrent of AGVs by the various regions (R) that constitute the layout of a FMS. Each layout of FMS is presented in a schematic diagram in order to show the paths, roads connected and ways of vehicles traffic. Each AGV needs some regions to complete its scheduled displacement, this displacement is related to an operation time or displacement $(\mathrm{t_0})$. The region number is defined as $R = R_1, R_2 \ldots, R_m$. In the example presented at Figure 9, different AGVs can compete for one or more regions (shared resources) that constitute the FMS. If there is a conflict, classic rules for dispatching (such as EDD, SPT, Priority) can be used.

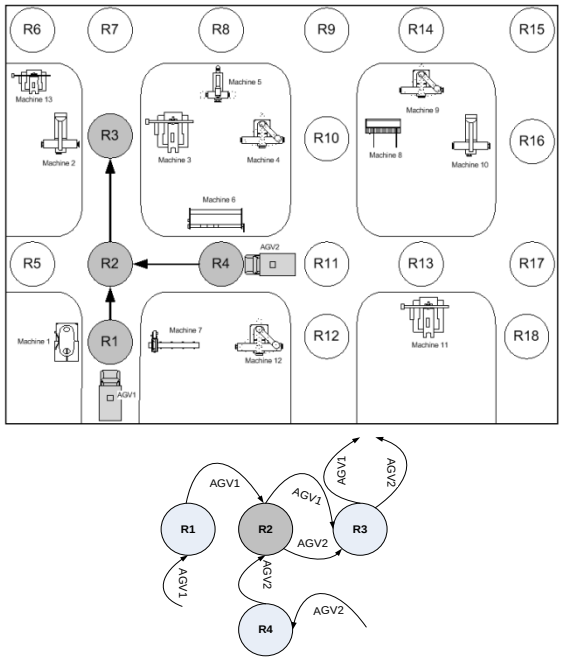


Figure 8. Planning Example for Path Planning Scheduling

Figure 8 shows a schematic example of scheduled displacement of AGVs into an FMS connected to 3 sequential regions, all of them aligned in the same direction. Vehicles can move around in accordance to the following: (*i*) AGV1 moves through the subsequent path, $R_1 \rightarrow R_2 \rightarrow R_3$ and (*ii*) AGV2 moves through $R_4 \rightarrow R_2 \rightarrow R_3$. (*iii*) we also know when (*t*) each AGV is willing to use each shared resource. The description of the processes and resources (Figure 8) is given by:

$$AGV1 = R_1(4t_0) \rightarrow R_2(3t_0) \rightarrow R_3(4t_0) \quad (2)$$
$$AGV2 = R_4(3t_0) \rightarrow R_2(4t_0) \rightarrow R_3(2t_0) \quad (3)$$

The Boolean expression that represents the dynamic is represented by:

$$AGV1 = R_1 \oplus R_2 \oplus R_3 = \underbrace{A}_{R1\overline{R2}R3} + \underbrace{B}_{\overline{R1}R2\overline{R3}} + \underbrace{C}_{R1R2R3} \quad (4)$$

$$AGV2 = R_4 \oplus R_2 \oplus R_3 = \underbrace{D}_{R4\overline{R2}R3} + \underbrace{E}_{\overline{R4}R2\overline{R3}} + \underbrace{F}_{R4R2R3} \quad (5)$$

In the dynamics of edge reversal for the example system, the initial acyclic orientation adopted is determined from the criteria of EDD, priority AGV1 and SPT. The first operations being processed are the displacements of AGV1 and AGV2 on $R_1 = R1, R4$ where $R_i$are sinks and operation time $t_0$ is given by $(t_f)$ $\min t \in R_i, t_0 = 3$ and remainder time to finish the displacement $(t_f)$ of R1 is 1, while $R_4$ is completed. In the following edge reversal, operations $R_2$ = R1, R2 and $t_0 = 1$ are selected. The next steps are summarized in Figure 9 and Figure 10. The previous shows that the problem of path planning can be treated as a Job-shop problem.

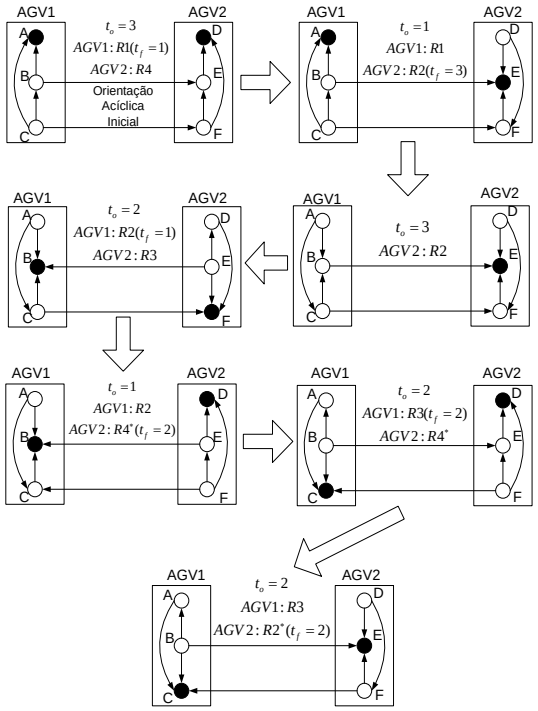


Figure 9. Example of SER on Path Planning

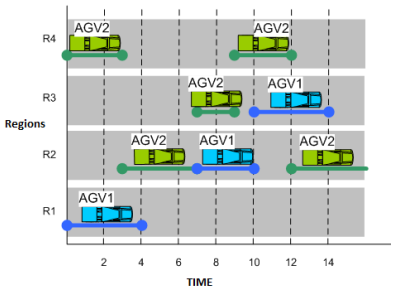


Figure 10. The Generated Schedule

## CONCLUSION

With the current growing interest in Flexible Manufacturing System (FMS), there is a growing need for distributed Job-shop algorithms. This article presents an implementation of a distributed scheduling algorithm for decentralized Job-shop systems that can be used for FMS control and scheduling. This novel approach allows decentralization of the job control and enables the distributed control to deal with any modification of the due time, caused by its asynchronous nature. The next step is the use of this algorithm in two real conditions: (*i*) AGV traffic control in automated container terminal and automated large scale freight transport systems and (*ii*) computational grid scheduling and grid data movement.

## REFERENCES

[1] Vieira, F.R.J., Rezende, J.F., Barbosa, V.C. and Fdida, S., "Scheduling links for heavy traffic on interfering routes in wireless mesh networks". Computer Networks, 2011.

[2] Barbosa, V.C., "An Introduction to Distributed Algorithms", MIT Press, 1996.

[3] Barbosa, V.C. and Gafni, E., "Concurrency in heavily loaded neighborhood-constrained systems", ACM Transactions on Programming Languages and Systems 11(4), 1989, pp. 562-584.

[4] França, F.M.G. and Faria, L., "Optimal mapping of neighbourhood-constrained systems", In: Proceedings, Irregular'95, Springer-Verlag, Lyon, France, Lecture Notes in Computer Science, 1995 , pp. 165-170.

[5] Carvalho, D, Protti, F., Gregorio, M.D. and França F.M.G., "A novel distributed scheduling algorithm for resource sharing under near-heavy load". Lecture Notes in Computer Science. 2005, pp. 431-442.

[6] Herrero-Perez, D. and Martinez-Barbera, H., "Modeling Distributed Transportation Systems Composed of Flexible Automated Guided Vehicles in Flexible Manufacturing Systems", IEEE Transactions on Industrial Informatics, vol. 6, 2010, pp. 166 – 180.

[7] Hartley, J., FMS at Work. IFS Publications Ltd., North-Holland Publishing Company, Division of Elsevier Science Publishers B. V, 1984.

[8] Harrison, D.K. and Petty, D.J. "Systems for Planning and Control in Manufacturing". Butterworth-Heinemannm, Elsevier Science, 2002.

[9] Askin, A.G, Standridge, C.R, "Modeling and Analysis of Manufacturing Systems". John Wiley & Sons, 1993.

[10] Vinod, V. and Sridharan, R., "Simulation modeling and analysis of due-date assignment methods and scheduling decision rules in a dynamic job shop production system". International Journal of Production Economics, 129(1), 2011, pp. 127–146.

[11] Leung, J.Y.T., "Handbook of Scheduling - Algorithms, Models, and Performance Analysis". Chapman & Hall CRC Press LLC, 2004.

[12] Liao, C.J., and You, C.T., "An improved formulation for the job-shop scheduling problem". The Journal of the Operational Research Society 43(11),1992, pp.1047- 1054.

[13] Lu, H.L., Huang, G.Q. and Yang, H.D., "Integrating order review/release and dispatching rules for assembly job shop scheduling using a simulation approach". International Journal of Production Research 49(3), 2011, pp. 647 – 669.

[14] Chan, F.T.S., Chan, H.K. and Lau, H.C.W.,"Analysis of dynamic dispatching rules for a exible manufacturing system". Journal of Materials Processing Technology 138, 2003, pp.325-331.

[15] Dominic, P.D.D., Kaliyamoorthy, S. and Kumar, M.S., "Efficient dispatching rules for dynamic job-shop scheduling". The International Journal of Advanced Manufacturing Technology 24(1-2), 2004, pp. 70-75.

[16] Meng, J, Soh, Y. and Wang, Y, "A tcpn model and deadlock avoidance for fms job shop scheduling and control system". In: IEEE International Workshop on Emerging Technologies and Factory Automation, Paris, France, 1995, pp 521- 532.

[17] Wu, N. and Zhow, M., "Modeling and deadlock control of automated guided vehicle systems", In: IEEE/ASME Transactions on Mechatronics, vol. 9, 2004, pp. 50-57.

[18] Zhang, H., Li, D., Yang, S. and Wang, W., "A new model of exible manufacturing system based on petri nets". In: International Conference on Mechatronics and Automation, ICMA 2007, 2007, pp. 3894-3899.

 ISBN: 978-1-61208-224-0 