Hindawi
Complexity
Volume 2018, Article ID 7235105, 15 pages
https://doi.org/10.1155/2018/7235105

WILEY | Hindawi

Research Article

Distributed Control of a Manufacturing System with One-Dimensional Cellular Automata

Irving Barragan-Vite, Juan C. Seck-Tuoh-Mora, Norberto Hernandez-Romero, Joselito Medina-Marin, and Eva S. Hernandez-Gress

Engineering Department, Autonomous University of Hidalgo State, Carr. Pachuca-Tulancingo, Col. Carboneras, Mineral de la Reforma, Hidalgo 42184, Mexico

Correspondence should be addressed to Irving Barragan-Vite; irving.vite@gmail.com

Received 27 April 2018; Revised 12 July 2018; Accepted 29 July 2018; Published 4 October 2018

Academic Editor: Zhiwei Gao

We present a distributed control modeling approach for an automated manufacturing system based on the dynamics of one-dimensional cellular automata. This is inspired by the fact that both cellular automata and manufacturing systems are discrete dynamical systems where local interactions given among their elements (resources) can lead to complex dynamics, despite the simple rules governing such interactions. The cellular automaton model developed in this study focuses on two states of the resources of a manufacturing system, namely, busy or idle. However, the interaction among the resources such as whether they are shared at different stages of the manufacturing process determines the global dynamics of the system. A procedure is shown to obtain the local evolution rule of the automaton based on the relationships among the resources and the material flow through the manufacturing process. The resulting distributed control of the manufacturing system appears to be heterarchical, and the evolution of the cellular automaton exhibits a Class II behavior for some given disordered initial conditions.

1. Introduction

The demand for top-quality products, as well as a variety of them, has led most manufacturing industries to automate their production processes [1]. By doing this, enterprises achieve an increase in their productivity as well as flexibility to rapidly respond to customers' demand. However, many issues arise at modeling automated manufacturing systems due to the rapid changes in global markets and the difficulty to foresee events such as machine failures and backlogging among other problems during a manufacturing system operation. Thus, decisions are required as soon as an undesired event occurs, and this involves having a control capable to react and prevent the manufacturing system to deviate from the optimal operating conditions.

Controlling a manufacturing system means to manage the activities of the system with the aim at operating according to the production plans and schedules, monitoring at every moment the flow of the parts being processed [2]. In a manufacturing scheme, the distributed control consists in dividing the control task into several entities capable to make decisions [3]. This allows making decisions rapidly and offers robustness against unforeseen failures as well as flexibility, scalability, modularity, and reconfigurability which are features of today manufacturing systems and the manufacturing paradigms [4].

Although not new, agent-based and holonic manufacturing models meet most or all the features described above and are suitable for distributed control as it is shown in [5–7]. However, they are not easy to implement, and thus, other approaches have been developed. In [8], a distributed control method is proposed based on fuzzy logic and the decomposition of the system into several subsystems. The fuzzy logic is also applied in [9] to control the flow in a manufacturing system together with a multicriteria decision-making method. In [10], a supervisory control is developed in order to decide

when to increase the production capacity based on local controllers and the cumulative demand.

Among the tools for modeling automated manufacturing systems, Petri nets are one of the most used, due to their graphical representation and mathematical formalism [11–13]. However, Petri nets are difficult to manage when there is a large number of components in the system leading to a large number of possible states in the net. Computer simulation models have been also one of the most used tools to model automated manufacturing systems [14] as a way to overcome the difficulties resulted from analytical models [15]. A disadvantage of computer-based models is that they represent the logic of the systems and measure their performance for specific scenarios by simulation. Most of the graphical, analytical and simulation models rely on assumptions that are unrealistic, and hence, there is a need of modeling methods easy to analyze and that match with the real operation of the increasingly complex manufacturing systems.

This paper proposes a procedure to model an automated manufacturing system with cellular automata, inspired by the local interactions among the cells and the global dynamics they produce. Cellular automata are dynamical systems composed of identical elements called cells. Every cell can take a state over a finite set of states, and each cell interacts locally with its nearest neighboring cells to change its state. Once a local evolution rule has been determined, all the cells change their state synchronously over discrete time steps. When the cells are arranged in a linear way, the cellular automaton is one-dimensional.

One of the major contributions to the one-dimensional cellular automata study has been their dynamic classification [16]. No matter which initial state is chosen for a cellular automaton, their evolution exhibits properties of one of four classes. On account of the dynamical features of cellular automata, they have been used to model systems in biology [17, 18], ecology [19, 20], physics [21], and chemistry [22]. Urban systems, as well as land use, have been also application fields of cellular automata as it is shown in [23–26].

Cellular automata were proposed in [27, 28] to model automated manufacturing systems considering the resource activities; however, the dynamics of the proposed cellular automaton model are performed asynchronously. Assembly simulation and assembly sequence identification were addressed in [29] with cellular automata. In [30], cellular automaton models are proposed for flexible manufacturing systems. These models are obtained by means of Petri net representations which are reduced in order to obtain subnets which form cells; however, the reduction of the Petri nets seems to be difficult to perform for large models.

In the cellular automaton model proposed in this manuscript, the cells represent the resources of the system and they are arranged according to the sequence of operations that a part follows throughout the system. The contribution of this study relies on the construction of the local evolution rule, which considers the constraints of the flow of parts due to resource sharing and capacity, and how this rule is used to resolve and prevent conflicts in the manufacturing system based on deterministic decisions, yielding a way to control the flow. The evolution of the cellular automaton shows a Class II behavior, and the control of the manufacturing system seems to be heterarchical; that is, it does not depend on centralized decisions.

The rest of the paper is structured as follows: Section 2 details the characteristics of automated manufacturing systems as well as that of cellular automata. Section 3 presents some preliminaries and considerations about the proposed modeling method. The modeling method is explained in Section 4. Section 5 shows the dynamics of the automated manufacturing system example by the cellular automaton evolutions. Section 6 presents a number of cases where the performance of the automated manufacturing system example under the control of the cellular automaton evolution rule is compared with the simulation of the system with a Petri net model. Finally, a discussion on the method is given in Section 7 and conclusions are provided in Section 8.

2. Basic Concepts

2.1. Automated Manufacturing Systems. Automated manufacturing systems (AMS) are discrete dynamical systems where events occur asynchronously [31, 32] such as the starting and the ending of an operation or the arrival or departure of parts [33]. Typically, an AMS is composed of numerically controlled machines which are connected by automated material handling devices like robots and automated guided vehicles. All the machines and material handling devices are controlled by a centralized computer system [34, 35]. The elements of the AMS interact each other to produce several parts which follow a determined sequence of operations carried out by machines, operators, or both of them.

An automated manufacturing system is a system capable to process a wide variety of parts simultaneously due to its high degree of automation [36]. An AMS is composed of workstations where the parts are automatically processed, an automated material transport system that interconnects the entire AMS and a computer control system that integrates all the manufacturing tasks. The operation of an AMS consists in processing a number of different parts. The parts are considered discrete entities which flow through a determined number of operations. Resources are required to perform the operations such as numerically controlled machines and material handling devices such as robots to load or unload the machines. Storage areas are also common components of an AMS, especially those which decouple operations referred to as buffers. Generally, the number of resources is limited and they need to be shared in order to carry out the operations. Resource sharing may originate well-known problems like conflicts and deadlocks [37] and diminish the system performance.

Figure 1 shows a drawing of an AMS. The manufacturing process carried out by the system consists in machining raw parts which follow a fixed sequence of operations. Firstly, the parts go to an operation performed by Machine 1, then, they go to a second operation carried out by Machine 2, and finally, the parts finish their processing at Machine 3. The parts are transported by a conveyor throughout the

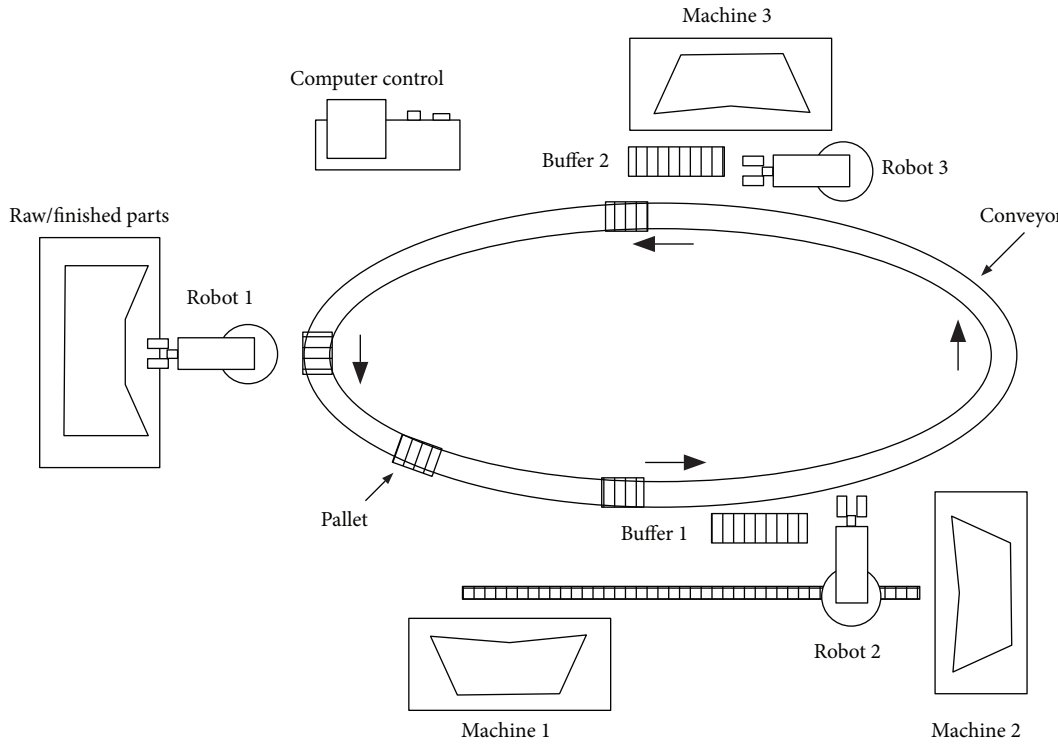


Figure 1: An automated manufacturing system (the arrows in bold shows the movement of the conveyor).

operations while robots load and unload the machines, the buffers, and the conveyor. The process begins when Robot 1 takes a raw part from a storage and puts it on pallets which are moved by the conveyor. Then, Robot 2 takes the part from the pallet, once the conveyor has stopped, and puts the part in Buffer 1. When the operation at Machine 1 has finished, Robot 2 withdraws the part and puts it in Buffer 1. If Machine 2 is idle, Robot 2 takes the part from Buffer 1 to Machine 2. As it can be seen in the figure, Robot 2 moves the parts from the conveyor belt to Buffer 1, from Buffer 1 to Machine 1 or Machine 2, from the Machines 1 and 2 to Buffer 1, and from Buffer 1 to the conveyor belt. All these movements are performed once Robot 2 is free and a part is ready to be moved.

When a part has been processed by Machine 2, it goes to the third and final operation. Once the part has reached the corresponding station, Robot 3 takes it to Buffer 2, and if Machine 3 is idle, then, the part is moved to the machine by Robot 3. When the third operation has finished, the part is taken to the storage for finished parts. Robot 1 takes the finished part from the conveyor belt and puts it in the storage area.

2.2. One-Dimensional Cellular Automata. A cellular automaton is a dynamical system whose evolution is discrete in time and space [38]. Cellular automata consist of identical elements called cells, and every cell can assume a state from a finite set of states. Each cell interacts with the nearest ones forming a neighborhood in order to change its state in a local way, causing a global state change of the automaton. Cells evolve individually and simultaneously to the next time step according to an evolution rule that determines which will be the next state of each cell based on its current state and the ones of its neighbors. This is referred to as a local evolution rule. The following explanations are focused on cellular automata in one dimension.

Formally, the local evolution rule is defined by $\varphi : S^n \to S$, where S is the finite set of states and n is the size of the neighborhoods. We define the size of the neighborhoods as $n = l + r + 1$ such that r and l are the numbers of neighbors to the right and to the left, respectively, of every cell. r and l are fixed values and constant for every cell c_i; the index i represents the position of the cell in the one-dimensional array of cells. In this way, let $c_i^t = a$ be the state of cell c_i at the instant t such that $a \in S$. Thus, $c_i^{t+1} = \varphi(c_{i-l}^t, \cdots, c_{i-1}^t c_i^t c_{i+1}^t, \cdots, c_{i+r}^t)$. A configuration C consists in an assignment of states to every cell in the linear arrangement. In particular, C^0 is the initial configuration, at the instant $t = 0$. Figure 2 shows the application of the local evolution rule in a one-dimensional cellular automaton of two states and neighborhoods of size $n = 3$, with $r = 1$ and $l = 1$.

Although theoretically one-dimensional cellular automata have an infinite number of cells, in practice and due to computational constraints, they consist of a limited number of cells. However, the cell that is at the rightmost side of the linear array and the one that is at the leftmost side lack from the right or the left neighbors, accordingly. A common practice to resolve this problem and to maintain a homogenous neighborhood size for every cell is to consider periodic boundaries where the rightmost cell in the array takes as right

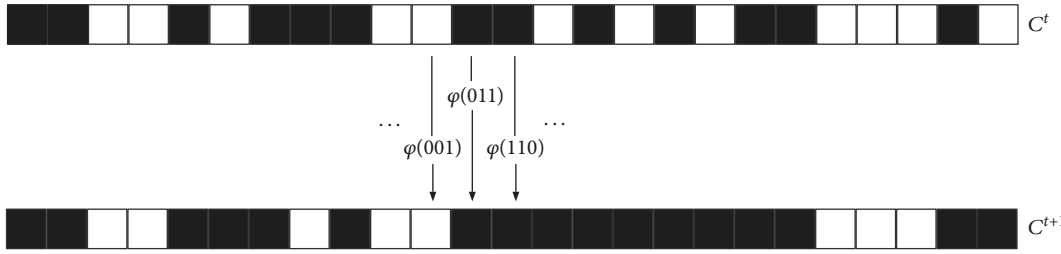


Figure 2: Example of the evolution of a one-dimensional cellular automaton with $n = 3$ and states 0 and 1 depicted as white and black squares, respectively.

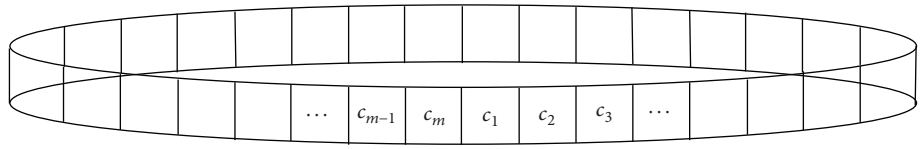


Figure 3: Ring formation of the linear array of cells, where m is the number of cells.

neighbors the first r cells of the left side of the array and likewise the leftmost cell takes as left neighbors the last l cells of the right side of the array. This can be thought of as forming a ring with the linear array as it is shown in Figure 3.

According to the long-term evolution of an automaton and Stephen Wolfram's classification, its global dynamics can be organized into four classes [39, 40]. Class I evolution tends to a fixed or stable state; Class II shows periodic and stable evolutions. For Class III, evolutions present a chaotic behavior and Class IV is considered the most interesting since evolutions produce complex behaviors over a stable or periodic background.

3. Modeling Method Assumptions

In this section, we give the basics of our proposed methodology to model an AMS with one-dimensional cellular automata. The modeling method is based on the assumptions that only one type of part is processed in the AMS and that the parts flow through the manufacturing process following the same sequence of operations. One way to model manufacturing systems is by means of fluid models where the parts go through a sequence of operations until they depart from the system. Thus, one-dimensional cellular automata are used in this paper taking into account such type of models, providing a way to represent not only the sequence of activities but the interaction among the resources. Additionally, many studies have been conducted to determine the properties of one-dimensional cellular automata [41–43], and a De Bruijn diagram is one of the classical tools to perform preliminary analyses. We take advantage of this tool to define the evolution rule by which the manufacturing system will be controlled.

The modeling approach does not take into account the time an operation takes to be executed, but only considers if a part is at an operation or not. If a part is at an operation, the resource that performs such an operation is busy and, no matter how much time the operation takes, the next state of the part is to be in another operation according to the sequence of operations, and the resource may become idle.

In most AMS, some resources must be shared by the operations and this causes conflicts. A conflict arises when such operations request the shared resource at the same time. In the proposed modeling approach, the shared resources are not modeled by the cells of the automaton, but they are taken into account in determining the evolution rule. Only resources that are not shared are represented by each cell of the automaton. The possible states of the cells are 1 or 0, depending on whether the corresponding operation is being executed or not, respectively. A cell on state 1 not only means the corresponding operation is being executed but also means that a part is ready to be moved to the next operation such that a shared resource like a robot may be required to carry out the movement.

The movement of a part from one operation to another is represented by the evolution of the automaton, but the movement is subject to constraints in the flow of parts. Two types of constraints are considered: the first one is related to the request of a shared resource at the same time, and the second one is about the capacity of the resources to process or hold only one part at a time.

The sequence of operations is modeled in such a way that the first cell at the left of the array of cells represents the first operation; the second cell, the second operation; and so on. A cell changes from 0 to 1 whenever the previous cell is on state 1 which means that the part moves from one operation to the following one whenever there is no flow constraint such as the ones mentioned above.

Due to that a resource may be used at different stages of the manufacturing process, different cells may represent the same resource at different positions in the linear array of cells like a buffer that can be modeled by a cell before a machining operation and then another cell can model the same buffer after finishing the operation, meaning that the buffer holds

Table 1: Operations of the AMS shown in Figure 1 and the resources involved.

Resource	Activities	Resources requested to be available
Storage area	Hold raw parts and finished parts	Robot 1
Robot 1	Load and unload the conveyor	Conveyor and storage area
Robot 2	Load and unload Buffer 1, Machine 1, and Machine 2	Buffer 1, Machine 1, and Machine 2
Robot 3	Load and unload Buffer 2, Machine 3, and conveyor	Buffer 2, Machine 3, and conveyor
Machine 1	Operation 1	Robot 2
Machine 2	Operation 2	Robot 2
Machine 3	Operation 3	Robot 3
Conveyor	Transport parts	Robot 1, Robot 2, and Robot 3
Buffer 1	Temporally hold parts	Robot 2
Buffer 2	Temporally hold parts	Robot 3

unprocessed and processed parts, correspondingly. Then, another operation can follow and the same buffer can be used to hold the parts in the same way as the previous operation.

3.1. General Considerations to Obtain the Local Evolution Rule. The dynamics of the AMS rely on the construction of the local evolution rule of the cellular automaton. Resource sharing and capacity constraints are key to define the local evolution rule in such a way that the possible interactions between a resource and its nearest neighbors in the AMS operation are represented by the neighborhoods.

The size of the neighborhoods is determined by the cells representing an operation requesting the use of the same resource. For example, consider operations i and j which are modeled by cells c_i and c_j, with $i < j$ due to the sequence of activities, and assume that such activities request the use of the same resource such that c_j is the rightmost cell in the array requesting such a resource. Then, cell c_i is affected by the state of cell c_j in such a way that if $c_i = 1$ and $c_j = 1$, they are said to be in conflict, and therefore, c_j must be included into the neighborhood of c_i. Moreover, all of the cells between c_i and c_j, if any, will be also members of the neighborhood of c_i whether or not the middle cells request the same resource.

On the other hand, the way the flow of parts is modeled indicates that a cell changes its state from 0 to 1 whenever the previous cell is on state 1. Under this consideration, cell c_i is affected not only by cell c_j but by the state of cell c_{i-1}. Note that cell c_{i-1} does not request the same shared resource as cells c_i and c_j do. In this way, cells $c_{i-1}c_ic_{i+1}, \dots, c_{j-1}c_j$ form a neighborhood, such that $c_i^{t+1} = \varphi(c_{i-1}c_ic_{i+1}, \cdots, c_{j-1}c_j)$. If other neighborhoods can be constructed in the same way, but not for operations that share a different resource, the neighborhood with the maximum number of cells from corresponding cells c_{i-1} to c_j defines the size of all of the neighborhoods. Additionally, cells $c_{i-1}c_ic_{i+1}, \dots, c_{j-1}c_j$, which define the size of the neighborhoods, become the *standard neighborhood* in order to determine the evolution rule. That is, each time c_i^{t+1} is going to be determined for each neighborhood formed by every cell in the linear array, the flow constraints among the operations represented in the standard neighborhood $c_{i-1}c_ic_{i+1}, \dots, c_{j-1}c_j$ are taken into account as if every neighborhood has the same cells, namely, those of the standard neighborhood.

In order to resolve the flow constraints, such as the conflicts and the capacity of the resources described before, a policy must be defined since such situations are unavoidably represented by some of the neighborhoods according to the relations given among the cells of the standard neighborhood. Suppose $c_i^t = 1$ and $c_j^t = 1$ such that they request the same resource at the same time to move the corresponding part from the current operation to the next one. To provide a solution to this situation, a priority is given to activity j; that is to say, the part at operation c_j is allowed to be moved first, whether or not such a movement can be done. Meanwhile, c_i remains unchanged at the next time step. This way to solve the conflicting situation is called the *priority policy* in the proposed modeling method. The priority policy is also applied to the capacity constraint when $c_i^t = 1$ and $c_{i+1}^t = 1$ such that the flow from c_i to c_{i+1} cannot be performed until the part at c_{i+1} releases the corresponding resource.

4. Modeling Method

4.1. AMS Description. The first step of the proposed method is to have a description of the operations of the AMS and the resources used to perform them. Likewise, it is needed to have an understanding of the whole operation of the AMS in order to identify the constraints related to carrying out the activities. As with any modeling method, these considerations are general and they can be applied to different manufacturing systems to obtain the corresponding cellular automaton model. The explanations given in this section of how to obtain the evolution rule as well as the arrangement of the cells are intended to be generalized to similar manufacturing systems since these types of systems do not vary significantly in their structure and operation, and it could be better for systems with more resources than the one exemplified here. Then, let us refer to the AMS that is shown in Figure 1. Table 1 provides the activities performed by all of the resources in the AMS as well as the resources or conditions required to carry out such activities. It is considered for this AMS that the conveyor is always available, there

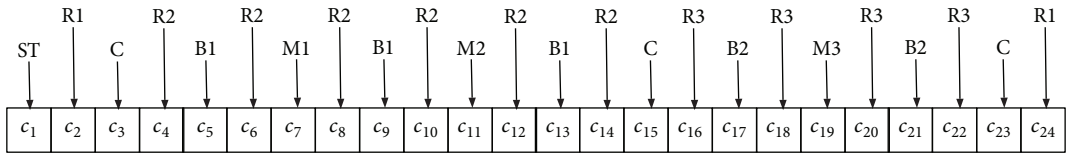


Figure 4: Linear array of cells representing the activities of the AMS of Figure 1.

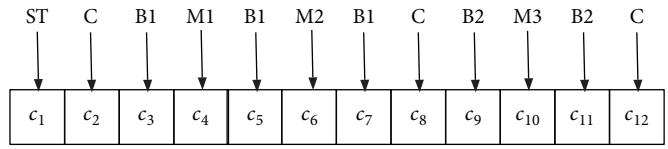


Figure 5: Linear array of cells resulting after discarding the representation of the activities of the robots.

are enough pallets to transport the parts, and the buffers can hold a large number of parts.

In the AMS of Figure 1, parts arrive to the system at a storage area. Then, Robot 1 takes a part from the storage area and puts it on a pallet which is transported by the conveyor until the next station where Machine 1 executes Operation 1. In this station, the conveyor as well as Machine 1, Machine 2, and Buffer 1 shares Robot 2 in order to be loaded or unloaded accordingly. After this station, the parts are transported to the next one where Machine 3 performs Operation 3. In this station, Robot 3 is shared by the conveyor, Machine 3, and Buffer 2. Finally, after being processed by Machine 3, the parts go to the storage area where Robot 1 unloads the conveyor and puts the parts in the storage. The conveyor keeps moving pallets and stops when a pallet with a part arrives at one of the stations where machines perform the corresponding operation or when a finished part needs to be stored. It is noticed that while Robot 2 loads either Machine 1, Machine 2, or Buffer 1, it is not possible to be loading any of the two remaining resources. A similar situation happens with Robot 3 where if it is loading the conveyor, it cannot be unloading Machine 3.

4.2. Determining the Number and the Arrangement of Cells. Once the activities of the resources and the operational constraints related to them have been identified, the next step is to represent each activity performed in the AMS with a cell and form the linear array of cells according to the sequence of activities. Figure 4 shows the linear array of cells where each cell represents an activity performed by a resource at the corresponding stage of the manufacturing process. Although shared resources are not considered to be modeled by the cells, they are shown in this figure to have a complete idea of how the cells represent the sequence of operations. As it was pointed out before, the parts flow from the left to the right in the linear array, going through all the stages of the manufacturing process. According to the AMS, the process begins when there are parts at the storage area (ST); then, the conveyor (C) transports the parts. Robots 1, 2, and 3 (R1, R2, and R3) load or unload Buffers 1 and 2 (B1 and B2) and Machines 1, 2, and 3 (M1, M2, and M3).

In the linear array of cells shown in Figure 4, only one cell is used to model the storage area. That is, after cell c_{24}, it should go cell c_{25} modeling the storage area where the finished parts are disposed; however, this cell (c_{25}) is substituted by cell c_1 which in fact models the storage area, and we consider a partial periodic boundary condition for cells c_{24} and c_1 in order to complete the last activity modeled by cell c_{24}.

The number of cells is reduced by discarding the cells modeling the activities of the robots. This procedure simplifies the construction of the evolution rule by having a fewer number of cells to consider in the neighborhoods. The resulting linear array of cells is illustrated in Figure 5 where the cells are renumbered.

4.3. Construction of the Evolution Rule. The next step of the proposed modeling method is to obtain the evolution rule. The first task is to determine the size of the neighborhoods according to the number of cell modeling activities that share a resource. From the linear array of cells of Figure 5, it can be observed that the cells c_2, c_3, c_4, c_5, c_6, and c_7 need Robot 2 in order to move a part from their corresponding activity to the next one. For example, conveyor (c_2) needs Robot 2 to move a part to Buffer 1 (c_3), and Buffer 1 (c_3) needs Robot 2 to move a part to Machine 1 (c_4). In a similar way, Robot 2 is required by cells c_4, c_5, c_6, and c_7. This means that Robot 2 is shared by these six cells. There are also a total of four cells sharing Robot 3, namely, c_8, c_9, c_{10}, and c_{11}, and a total of two cells, c_1 and c_{12}, sharing Robot 1.

The largest number of cells that share a resource is used to determine the size of the neighborhoods. In this case, cells c_2, c_3, c_4, c_5, c_6, and c_7 account for the maximum number of cells sharing a resource, where cell c_2 is the first cell sharing Robot 2 (c_i) and c_7 is the last one (c_j). According to the neighborhood structure, an extra cell is appended to the left of the ones that share the resource, completing the size of the neighborhoods. For the cellular automaton of this example, cell c_1 is appended to cells c_2, c_3, c_4, c_5, c_6, and c_7. Hence, $c_1c_2c_3c_4c_5c_6c_7$ is taken as the standard neighborhood to determine the evolution rule. The size of the neighborhood is $m = 7$, with $r = 5$ and $l = 1$, yielding a total

of $2^7 = 128$ neighborhoods with the form $c_{i-1}c_ic_{i+1}, \ldots, c_{m-1}c_m$, where c_i is the cell designated to evolve.

The constraints indicated for cells c_2, c_3, c_4, c_5, c_6, and c_7 with regard to sharing Robot 2 are used to define the evolution rule. Although such constraints may not apply to all the cells and their corresponding neighborhoods, they are used to set policies to the flow of parts to model the AMS dynamics. Therefore, all of the 128 neighborhoods are considered to be possible states of cells c_1, c_2, c_3, c_4, c_5, c_6, and c_7. For example, the neighborhood 0100000 means that Robot 2 is required to move the part from the conveyor (c_2) to Buffer 1 (c_3). Thus, for this neighborhood, $\varphi(0100000) = 0$ since the remainder of the cells after c_2 is not requesting the shared resource and the buffer is free to receive the part, leaving empty the conveyor after the movement since there is not any other part arriving at the buffer. The evolution of the automaton corresponding to this situation is partially shown in Figure 6.

Consider now the neighborhood 0010010 where Buffer 1 (c_3) and Machine 2 (c_6) request Robot 2 to move the corresponding part to the following activities: namely, to Machine 1 and Buffer 1 (c_4 and c_7, correspondingly). In this case, c_3 and c_6 are in conflict and both of the movements cannot occur simultaneously since they are mutually exclusive. By the priority policy applied to the activity at the downstream of the manufacturing process, the movement from Machine 2 (c_6) to Buffer 1 (c_7) is allowed to occur first. Thus, $\varphi(0100100) = 1$ since cell c_2 must remain unchanged allowing the occurrence of the other activity. The evolution of the cellular automaton for this case is shown in Figure 7. It can be observed that in the following neighborhood formed by cell c_3 (1000100), such a cell must stay on state 0 at the next time step because cell c_2 does not change its state. That is, there is no movement from the conveyor (c_2) to Buffer 1 (c_3) since Robot 2 is used to move a part from Machine 2 (c_6) to Buffer 1 (c_7), and thus, $c_3^{t+1} = 0$. This way to determine the states of the cells at the next time step is the guideline to determine the evolution rule for all of the 128 neighborhoods. That is, the thing considered is not only the current state of the cells in the neighborhood but also the previous movements in the flow of the parts.

The general process of the evolution rule can be seen as going through the paths of the De Bruijn diagram of the cellular automaton obtained from the AMS of Figure 1, forming all the possible neighborhoods while traversing the paths. The De Bruijn diagram consists of 2^{n-1} nodes, and each node is a partial neighborhood of $n-1$ cells. To form a complete neighborhood, two nodes u and v are connected by a directed link from u to v as long as the rightmost $n-2$ cells of node u overlap with the leftmost $n-2$ cells of node v. At most, each node has $|S|$ leaving links. Therefore, to construct the evolution rule, a pair of nodes u and v must be chosen such that u overlaps with v, and then state c_i^{t+1} is determined as it was explained above. Next, another node w is chosen such that v overlaps with w, and one more time state c_i^{t+1} is determined. The process continues forming a path $u \to v \to w \to \cdots$, and eventually, the path will either form a cycle or fall into a cycle.

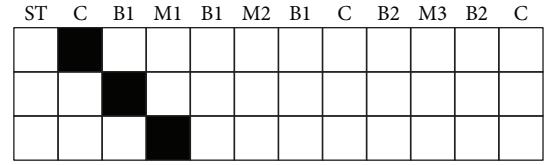


Figure 6: Evolution representing Robot 2 moving a part from the conveyor to Buffer 1.

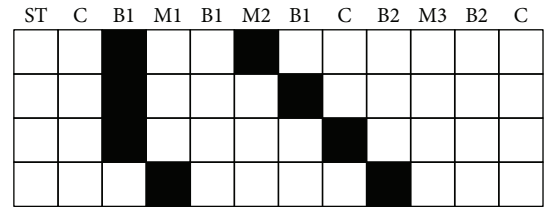


Figure 7: Robot 2 moving a part from Machine 2 to Buffer 1 and then to the conveyor, while another part in Buffer 1 waits until Robot 2 finishes the movements.

To illustrate the process, let us consider neighborhood 0100010 formed by nodes 010001 and 100010, as it is shown in Figure 8. For this neighborhood, $\varphi(0100010) = 1$ since cells c_2 and c_6 request the same resource at the same time and, by the priority policy, the operation at the downstream of the manufacturing process is allowed to use first the shared resource. After node 100010, one can go to either node 000100 or node 000101. Let us take the latter, that is, 000101, and form neighborhood 1000101. According to the decision made in the previous neighborhood, c_1 remain unchanged at $t+1$ in the current neighborhood, then c_2 do not receive any part from the previous operation, and thus, $\varphi(1000101) = 0$. As it can be observed, each time a node is chosen to continue the path, the partial neighborhood can end in 0 or 1. In order to give a brief description of the process to determine the evolution rule, the neighborhoods which end in 0 are chosen deliberately. In this way, after 000101, node 001010 is chosen such that neighborhood 0001010 is obtained. In this case, there is not any part to move from cell c_1 to cell c_2 and this last cell is not requesting the shared resource; therefore, $\varphi(0001010) = 0$. The same happens with neighborhood 0010100 formed by nodes 001010 and 010100, such that $\varphi(0010100) = 0$.

After 010100, the process continues with node 101000 and the resulting neighborhood is 0101000. In this situation, c_2 and c_4 are requesting the same resource, and by the priority policy, $\varphi(0101000) = 1$. Observe that according to the first neighborhood that originated this process, namely, 0100010, c_6 should change to 0 by applying the priority policy; however, due to the conditions of the new neighborhood 0101000, such a cell (now c_2) must remain unchanged at the next time step. As it can be noticed from the last node 101000, this one overlaps with node 010001 that started the path. Neighborhood 1010001 results from overlapping nodes 101000 and 010001, and for this case, $\varphi(1010001) = 0$; that is c_1 does not send any part to c_2 according to the previous

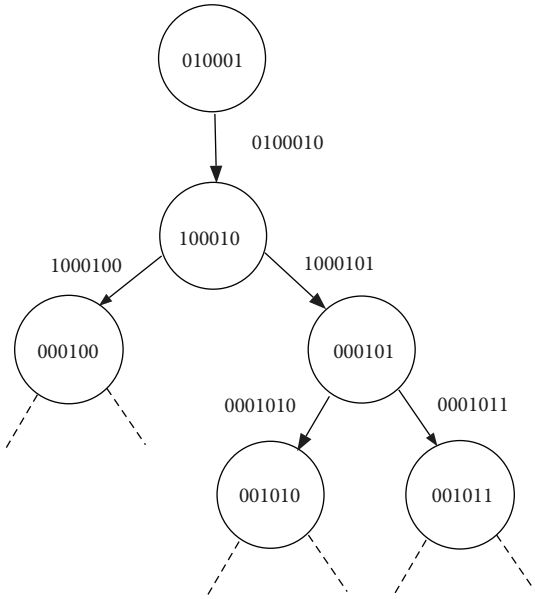


Figure 8: Process to determine the evolution rule following a path in the De Bruijn diagram.

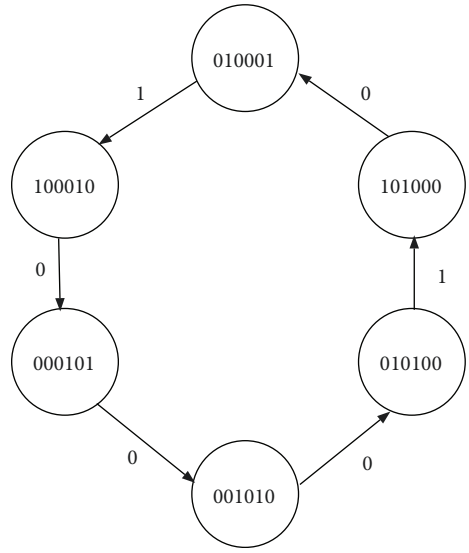


Figure 9: Path forming a cycle in the De Bruijn diagram starting and ending at node 010001. The result of the evolution is indicated by a number (0 or 1) next to the arrows.

neighborhood. Up to here, the process to determine the evolution rule is finished since the path followed has formed a cycle. The states determined for each neighborhood must be consistent with the flow of parts as it is shown in the De Bruijn subdiagram in Figure 9.

The process is applied to all of the remaining neighborhoods in the paths that have not been explored. Eventually, two or more paths will have common nodes, and for the neighborhoods formed by such nodes, the evolution rule definition must be consistent with the flow of parts considered in each path. Figure 10 presents some paths that share nodes with the cycle shown in Figure 9. Observe that the states determined for the evolution rule in each path is consistent with the flow of parts and the constraints considered for such a flow. Table 2 shows the complete local evolution rule definition, obtained as indicated in this section.

5. Cellular Automata Evolution Representing the Dynamics of the AMS

Once the evolution rule has been determined, the evolution of the cellular automaton can be carried out for different initial conditions of the AMS, which are represented by the initial configuration. In order to represent the dynamics of the AMS, a modification to the classical periodic boundary conditions is made, particularly to the right boundary. Since each cell of a configuration forms its neighborhood with $l = 1$ cell to the left and $r = 5$ cells to the right, the last five cells could not do it. Instead of completing the neighborhood of the last five cells with the ones at the beginning of the configuration, the missing cells are given state 0 as if the configuration was infinite to the right side, but with the cells with state 0. In fact, only the missing cell which is next to the last cell of the configuration will take the state of the first cell of the configuration. Figure 11 shows an evolution with the resulting modification of the boundary condition. In that figure, the cell representing the third machine (M3) in the initial configuration is missing three of the five cells to the right to complete its neighborhood, namely, the third, the fourth, and the fifth. By the modification to the right boundary condition, the third cell takes the state of the first cell of the initial configuration, that is, 1. The fourth and the fifth cells take state 0 forming neighborhood 0100100.

Figure 12(a) shows the evolution where only one part is in the AMS at the storage area (c_1). According to the AMS, the part goes through the whole manufacturing process from one operation to another since there is no constraint in its flow towards the end of the process. In Figure 12(b), a conflict situation is shown when initial state 100100100011 is chosen. In this initial condition, the storage area has a part ready to enter to the manufacturing process (c_1) where Robot 1 is requested to move the part to the conveyor. Likewise, a part is at Machine 1 (c_4) such that it requests Robot 2 to move such a part to Buffer 1 once the machining operation has finished. At the same time, Buffer 1 (c_7) requests Robot 2 to move the part in the buffer to the conveyor in order that it can continue with the final operation at Machine 3. Finally, there is a part in the conveyor (c_{12}) ready to be put in the storage by Robot 1. All of these conditions lead to a conflict since according to the evolution rule definition and by the priority policy, there is a circular request among the involved cells or operations.

Figure 12(c) shows a part in the storage area and one in Buffer 1 (c_3). According to the evolution rule, the part at the storage area is not moved to the conveyor until the part at Buffer 1 has been moved forward in the manufacturing process such that it is out of the neighborhood of c_1. The initial state in Figure 12(d) shows a part in the storage area and two parts in Buffer 1. One of the parts in Buffer 1 is ready to go to Machine 1, namely, the one at c_3, and the other one (c_5) to Machine 2. Since both movements cannot occur

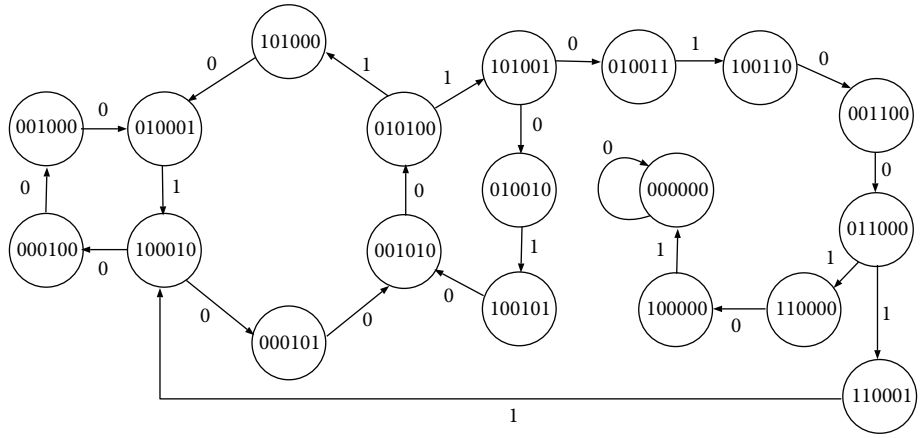


Figure 10: Some paths of the De Bruijn diagram showing the states determined for the evolution rule.

Table 2: Local evolution rule where $c_i^{t+1} = \varphi(c_{i-1}^t c_i^t c_{i+1}^t, \cdots, c_{j-1}^t c_j^t)$.

Neighborhood	c_i^{t+1}	Neighborhood	c_i^{t+1}	Neighborhood	c_i^{t+1}	Neighborhood	c_i^{t+1}
0000000	0	0100001	1	1000010	0	1100011	1
0000001	0	0100010	1	1000011	0	1100100	1
0000010	0	0100011	1	1000100	0	1100101	1
0000011	0	0100100	1	1000101	0	1100110	1
0000100	0	0100101	1	1000110	0	1100111	1
0000101	0	0100110	1	1000111	0	1101000	1
0000110	0	0100111	1	1001000	0	1101001	1
0000111	0	0101000	1	1001001	0	1101010	1
0001000	0	0101001	1	1001010	0	1101011	1
0001001	0	0101010	1	1001011	0	1101100	1
0001010	0	0101011	1	1001100	0	1101101	1
0001011	0	0101100	1	1001101	0	1101110	1
0001100	0	0101101	1	1001110	0	1101111	1
0001101	0	0101110	1	1001111	0	1110000	1
0001110	0	0101111	1	1010000	0	1110001	1
0001111	0	0110000	1	1010001	0	1110010	1
0010000	0	0110001	1	1010010	0	1110011	1
0010001	0	0110010	1	1010011	0	1110100	1
0010010	0	0110011	1	1010100	0	1110101	1
0010011	0	0110100	1	1010101	0	1110110	1
0010100	0	0110101	1	1010110	0	1110111	1
0010101	0	0110110	1	1010111	0	1111000	1
0010110	0	0110111	1	1011000	0	1111001	1
0010111	0	0111000	1	1011001	0	1111010	1
0011000	0	0111001	1	1011010	0	1111011	1
0011001	0	0111010	1	1011011	0	1111100	1
0011010	0	0111011	1	1011100	0	1111101	1
0011011	0	0111100	1	1011101	0	1111110	1
0011100	0	0111101	1	1011110	0	1111111	1
0011101	0	0111110	1	1011111	0		
0011110	0	0111111	1	1100000	0		
0011111	0	1000000	1	1100001	1		
0100000	0	1000001	1	1100010	1		

Figure 11: Evolution of the cellular automaton considering the boundary condition modification to the last five cells to the right side of each configuration.

simultaneously, the part at c_5 is chosen to be moved first by the priority policy at $t = 1$. At the next two time steps, the priority policy is applied again such that the part is moved from c_6 to c_7 at $t = 2$ and from c_7 to c_8 at $t = 3$. At $t = 4$, any part can be moved according to the evolution rule definition.

Some evolutions of the cellular automaton are provided in Figure 13 for disordered initial configurations of different widths. For a long number of generations depicted in Figures 13(c)–13(e), the cellular automaton exhibits periodic configurations in determined regions as well as the preservation of data, matching the properties of Class II of Wolfram classification [44]. The evolutions show the tendency of the parts to move towards the end of the manufacturing process. For an initial configuration with large blank spaces (strings of 0s), such a feature is easily noticed (Figures 13(a) and 13(d)). On the contrary, when there are fewer blank spaces in the initial configurations, the system tends to be blocked (Figures 13(c) and 13(d)) and the expected movement of the parts is not recovered for a while. In regard to the latter, when large blank spaces are present in the initial configurations, the evolutions rapidly reach a periodic behavior, but once a conflict arises, a halt in the movement of parts is transmitted towards the beginning of the process, just as it was modeled by the evolution rule. However, the movement of parts can be recovered once the conflicts are resolved at the final stages of the process and this is also transmitted to the beginning after some time steps in the evolution. The time required to resolve the conflicts makes the system to be in conflict in a periodic fashion.

6. Experiments and Results

In this section, the operation of the system shown in Figure 1 is performed using the proposed control method and a Petri net model simulating the normal operation of the system without a predefined control of the parts flow. The comparison is made in terms of the time required to produce a certain number of parts and in terms of the resource utilization, namely, machines and robots. A Petri net (PN) is a well-known tool to model manufacturing systems, and a generalized Petri net model is appropriate to compare the operation of the system. Generalized Petri nets are not timed nor stochastic, and the evolution of markings can be easily matched with the configurations of the cellular automaton such that discreteness in space and time is satisfied for both types of simulations. However, some considerations are needed to make a fair comparison between the two models (cellular automaton and Petri net). Broadly speaking, in a PN, the change of states occurs asynchronously; that is, if

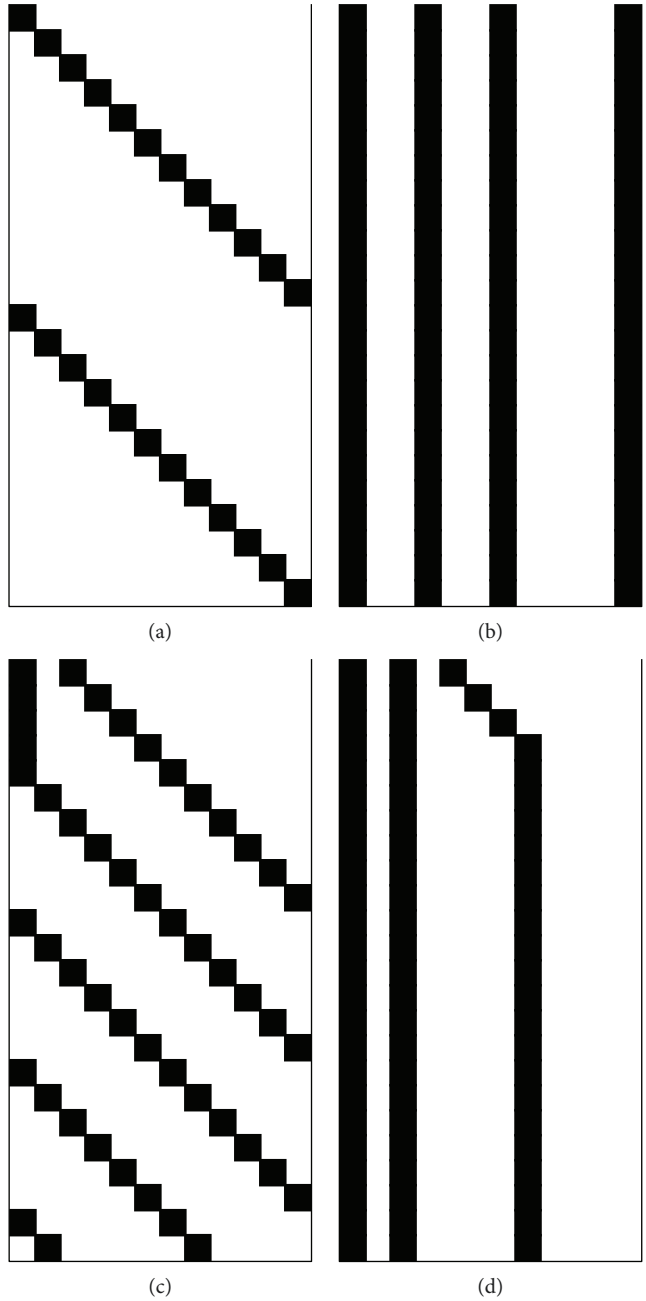
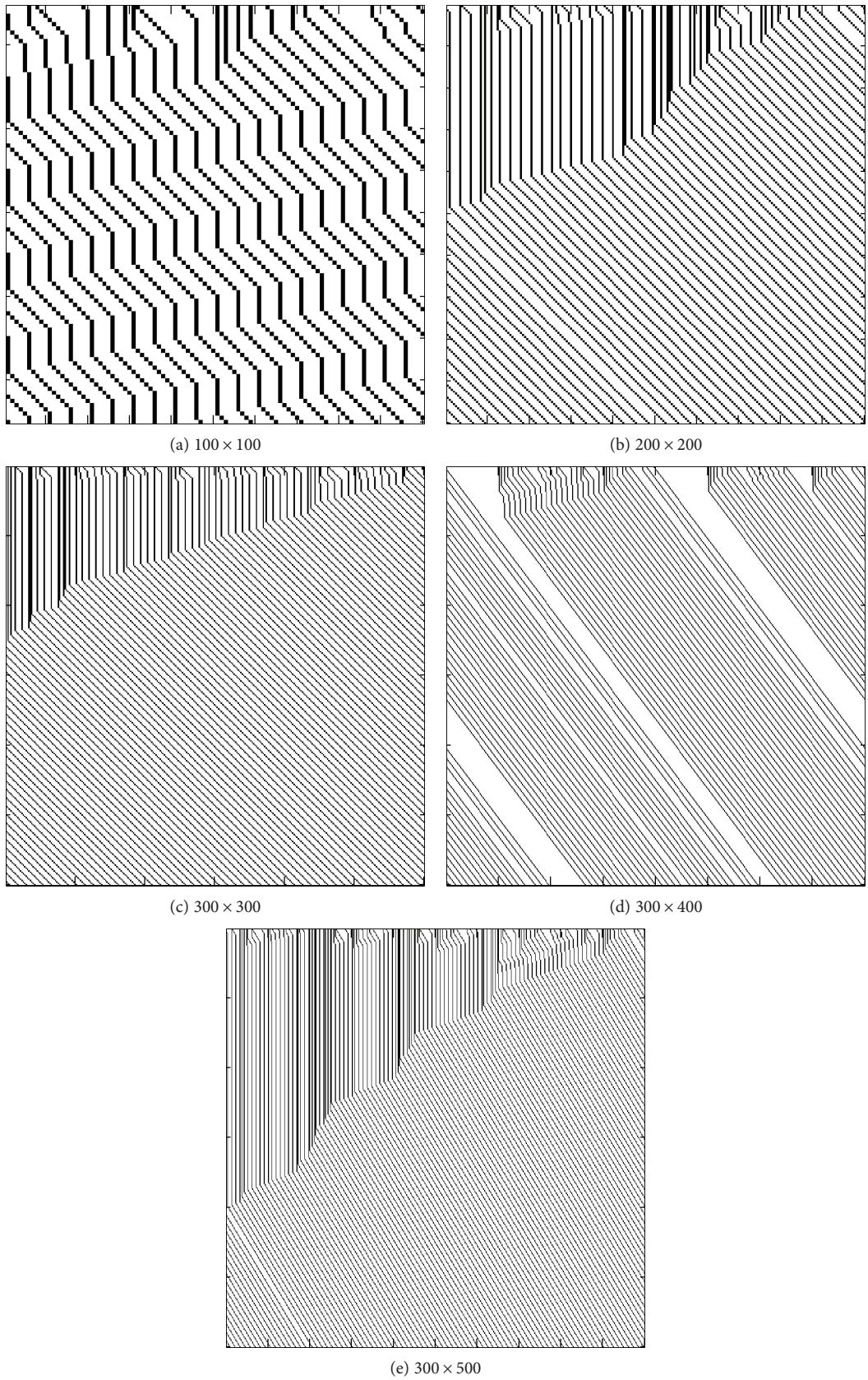


Figure 12: Evolutions of the cellular automaton representing different initial states of the AMS of Figure 1.

two movements can be done at the same time in the natural operation of a system, in the PN simulation, one of the movements is carried out first and then the other one, but not necessarily right one after the other. This is because of a matter of analysis which leads to the reachable marking tree, but if the two movements are parallel such that the occurrence of one them does not disable the occurrence of the other one at the next time step, then in the PN, the two simultaneous movements can be represented by firing the corresponding transitions one immediately after the other, reaching the same marking as if the two movements were done simultaneously. Another consideration is that for each



(a) 100×100

(b) 200×200

(c) 300×300

(d) 300×400

(e) 300×500

Figure 13: Evolutions of the cellular automaton for disordered initial configurations of different sizes (number of generations × number of cells).

Table 3: Simulation results for the PN model and the cellular automaton.

Case	Parts	Machine utilization (%)						Robot utilization (%)						Time steps	
		M1		M2		M3		R1		R2		R3			
		PN	CA	PN	CA	PN	CA	PN	CA	PN	CA	PN	CA	PN	CA
0	1	8.33	8.33	8.33	8.33	8.33	8.33	50	50	33.33	33.33	16.67	16.67	12	12
1	2	16.67	12.5	11.11	12.5	11.11	12.5	61.11	68.75	44.44	50	16.67	18.75	18	16
2	2	7.69	7.69	15.39	15.39	15.39	15.39	61.54	61.54	61.54	61.54	15.39	15.39	13	13
3	3	6.67	5.88	13.33	35.29	20	17.65	46.67	41.18	80	70.59	20	17.65	15	17
4	3	8.33	6.67	8.33	6.67	25	20	41.67	33.33	91.67	73.33	25	20	12	15
5	4	6.25	55.56	12.5	11.11	18.75	16.67	43.75	38.89	75	66.67	25	22.22	16	18
6	4	5.26	4.55	10.53	50	21.05	18.18	42.11	36.36	78.95	68.18	21.05	18.18	19	22
7	5	8.33	7.69	12.5	11.54	25	19.23	58.33	53.85	75	69.23	20.83	19.23	24	26
8	5	4.76	7.69	14.27	42.31	42.86	19.23	47.62	53.85	85.71	69.23	23.81	23.01	22	26

time step in the cellular automaton evolution, two firings are needed in the PN simulation to reach a marking equivalent to the configuration reached in the automaton. These two firings represent the robots' movements which are not modeled in the cellular automaton.

Nine cases are used to compare the proposed control method with the normal operation of the system. Each case consists of a number of parts distributed at different stages in the system; this number of parts ranges from one to five. The PN model was constructed and simulated on a free application (http://pipe2.sourceforge.net/). For each case, the shortest path of firings was calculated for the PN. The shortest path is the minimum number of firings from the initial marking to the final one. The final marking is where all the parts have left the system, and they are stored in the storage area at the end of the system. For example, we developed the Case 0 to show the equivalence between the evolution of markings of the PN and the evolution of the cellular automaton. Case 0 considers only one part located at the storage area as an unprocessed part. The shortest path for this case is [T0 T1 T4 T5 T2 T3 T6 T7 T8 T9 T10 T11 T12 T13 T14 T15 T16 T17 T18 T19 T20 T21 T22 T23]. Since there are twelve stages where this part can be located through the system, at the conveyor, at Buffer 1, at Machine 1 being processed, and so on, the total number of time steps to reach the storage area for finished parts is twelve. This number of time steps is obtained from the shortest path by dividing each into two transitions from T0 to T23. In this way, the shortest path calculated for each of the eight cases represents the minimum time to process all the parts in the system. The shortest paths were recalculated from those obtained with the aforementioned PN application since the original ones do not consider simultaneous movements. The results are shown in Table 3.

7. Discussion

The evolution rule representing the dynamics of the AMS shown in Figure 1 is based on some considerations taken in regard to the number of operations that share resources and to the capacity constraint. Therefore, one can observe in the evolutions shown in Figure 12 that some part movements could have been done, but they are not carried out due to constraints imposed to the evolution rule. For example, in Figure 12(c), Robot 1 is needed to move the part from c_1 to c_2 (storage area to conveyor) and Robot 2 is needed to move the part from c_3 to c_4 (Buffer 1 to Machine 1). However, by the priority policy, only the movement from c_3 to c_4 is allowed, although both of the movements are carried out by different resources. A similar situation occurs in the evolution shown in Figure 12(d), where the part at c_1 can be moved to c_2 since Robot 1 is needed to carry out the movement while Robot 2 is required to move the part from c_3 to c_4. Despite this, the procedure to obtain the evolution rule can be seen as a general policy to avoid conflicts in the flow of parts. For example, in Figure 12(c), the priority policy avoids a conflict situation if both parts were moved simultaneously, arriving at the operations at Machine 1, Machine 2, and Buffer 1 where Robot 2 is shared. In a real situation, the decision to give the priority to some operation can be made under the consideration of, for example, the cost related to the operation and such a consideration can be taken into account in obtaining the evolution rule.

In regard to the results obtained from the experiments, only in Case 1, the cellular automaton produces the parts in a shorter time than in the PN simulation, with 16 time steps of the cellular automaton against 18 time steps of the PN. Moreover, the utilization of the resources is greater than in the PN simulation result. However, from Case 2 to Case 8, the utilization of the machines and robots with the cellular automaton is smaller than in the PN; however, the time needed to produce the parts is greater than in the PN simulation. Figure 14 shows the plots for the number of parts in the system of Case 0, Case 1, Case 3, Case 5, and Case 7. For Case 0, where there is only one part, the cellular automaton and the PN have the same plot. Such an equivalence is confirmed by the results shown in Table 3. For Cases 3, 5, and 7 where the cellular automaton produces the parts in slightly more time steps than the PN, it is observed that from the beginning of the cellular automaton evolution to the middle of such an evolution, the parts are produced faster than in the PN simulation. This leads to a low utilization of the resources.

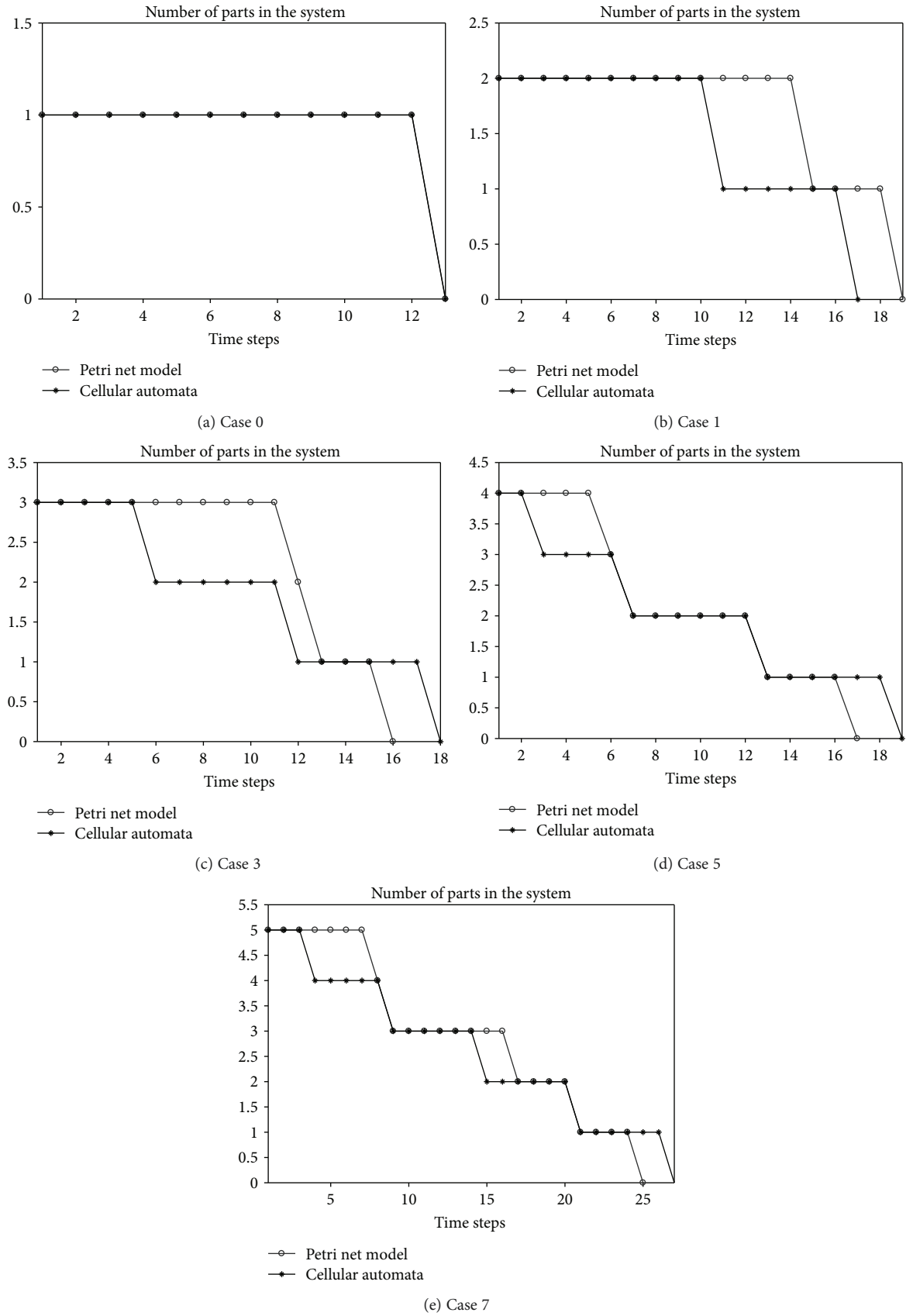


Figure 14: Plots showing the reduction of parts as they depart from the system.

Furthermore, a pattern in the production of parts is observed in the plots of Case 3, Case 5, and Case 7, which matches with the evolutions shown in Figure 13.

8. Conclusions

A method to model an automated manufacturing system with one-dimensional cellular automata has been presented, providing a new perspective in modeling the dynamics of such type of systems. Some considerations have been taken in order to represent the dynamics of an AMS such as that only one type of part is processed in the system and with a fixed sequence of operations. Likewise, the resources like the machines have been considered to have the capacity to process only one part at a time. As it can be seen in this paper, the decisions to control the flow of parts are taken locally, at each operation performed by the resources, which are modeled with the cells of the automaton and therefore obtaining a distributed control of the system. Moreover, due to the locality of the decisions, the control is not centralized on any entity but divided along to some of the entities of the system, namely, those modeled by the cells of the automaton.

The evolution of the cellular automaton shows the flow of the parts as it is expected in the real system. The local evolution rule delays the advance of the parts whenever a potential conflict can occur downstream in the process, and once the situation is resolved by a priority policy, the flow continues. However, it can be noticed from the evolutions that when there is an excess of parts in the system (initial configurations), this is blocked and therefore diminishing the efficiency of the system. Nevertheless, the comparative experiments show that the utilization of the resources is low for almost all the experiment cases. In a further work, different modeling strategies could be considered such as more states for the cells like failures and repairs; likewise, different local evolutions rule to manage different types of decisions. Indeed, the discreteness of the rules can be used instead of fuzzy rules, as it is done in many studies.

Data Availability

All the numerical results obtained to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by PRODEP grant F-PROMEP-38/Rev 03 and by the National Council for Science and Technology of Mexico (CONACYT) with project number CB-2014-237323.

References

[1] H.-O. Guenther and T. E. Lee, "Scheduling and control of automated manufacturing systems," *OR Spectrum*, vol. 29, no. 3, pp. 373-374, 2007.

[2] P. Leitão, "Agent-based distributed manufacturing control: a state-of-the-art survey," *Engineering Applications of Artificial Intelligence*, vol. 22, no. 7, pp. 979–991, 2009.

[3] D. Trentesaux, "Distributed control of production systems," *Engineering Applications of Artificial Intelligence*, vol. 22, no. 7, pp. 971–978, 2009.

[4] J. Barbosa, P. Leitão, E. Adam, and D. Trentesaux, "Dynamic self-organization in holonic multi-agent manufacturing systems: the ADACOR evolution," *Computers in Industry*, vol. 66, pp. 99–111, 2015.

[5] R. V. Barenji, A. V. Barenji, and M. Hashemipour, "A multi-agent rfid-enabled distributed control system for a flexible manufacturing shop," *The International Journal of Advanced Manufacturing Technology*, vol. 71, no. 9–12, pp. 1773–1791, 2014.

[6] T. Borangiu, S. Raileanu, D. Trentesaux, T. Berger, and I. Iacob, "Distributed manufacturing control with extended cnp interaction of intelligent products," *Journal of Intelligent Manufacturing*, vol. 25, no. 5, pp. 1065–1075, 2014.

[7] J. A. Arauzo, R. del Olmo-Martínez, J. J. Laviós, and J. J. de Benito-Martín, "Programación y control de sistemas de fabricación flexibles: un enfoque holónico," *Revista Iberoamericana de Automática e Informática Industrial RIAI*, vol. 12, no. 1, pp. 58–68, 2015.

[8] K. Tamani, R. Boukezzoula, and G. Habchi, "Intelligent distributed and supervised flow control methodology for production systems," *Engineering Applications of Artificial Intelligence*, vol. 22, no. 7, pp. 1104–1116, 2009.

[9] K. Rudnik and D. Kacprzak, "Fuzzy topsis method with ordered fuzzy numbers for flow control in a manufacturing system," *Applied Soft Computing*, vol. 52, pp. 1020–1041, 2017.

[10] K. Tamani, R. Boukezzoula, and G. Habchi, "Supervisory-based capacity allocation control for manufacturing systems," *International Journal of Manufacturing Technology and Management*, vol. 20, no. 1–4, pp. 259–285, 2010.

[11] M. Zhou and K. Venkatesh, *Modeling, Simulation and Control of Flexible Manufacturing Systems: A Petri Net Approach*, World Scientific, New York, NY, USA, 1998.

[12] Z. Li, N. Wu, and M. Zhou, "Deadlock control of automated manufacturing systems based on petri nets—a literature review," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 4, pp. 437–462, 2011.

[13] B. K. Choi, H. Kim, D. Kang, A. A. Jamjoom, and M. A. Abdullah, "Parameterized acd modeling of flexible manufacturing systems," *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 2, pp. 637–642, 2014.

[14] J. Novak-Marcincin, "Computer modelling and simulation of automated manufacturing systems," *Annals of the Faculty of Engineering Hunedoara-International Journal of Engineering*, vol. 11, no. 2, pp. 23–26, 2013.

[15] Q. Wang and C. R. Chatwin, "Key issues and developments in modelling and simulation-based methodologies for manufacturing systems analysis, design and performance evaluation," *International Journal of Advanced Manufacturing Technology*, vol. 25, no. 11-12, pp. 1254–1265, 2005.

[16] S. Wolfram, "Universality and complexity in cellular automata," *Physica D: Nonlinear Phenomena*, vol. 10, no. 1-2, pp. 1–35, 1984.

[17] S. Athithan, V. P. Shukla, and S. R. Biradar, "Dynamic cellular automata based epidemic spread model for population in

patches with movement," *Journal of Computational Environmental Sciences*, vol. 2014, Article ID 518053, 8 pages, 2014.

[18] P. Eosina, T. Djatna, and H. Khusun, "A cellular automata modeling for visualizing and predicting spreading patterns of dengue fever," *Telkomnika*, vol. 14, no. 1, pp. 228–237, 2016.

[19] M. Martnez Molina, M. A. Moreno-Armendriz, and J. C. Seck Tuoh Mora, "On the spatial dynamics and oscillatory behavior of a predator-prey model based on cellular automata and local particle swarm optimization," *Journal of Theoretical Biology*, vol. 336, pp. 173–184, 2013.

[20] X. Ke, F. Wu, and C. Ma, "Scenario analysis on climate change impacts of urban land expansion under different urbanization patterns: a case study of wuhan metropolitan," *Advances in Meteorology*, vol. 2013, Article ID 293636, 12 pages, 2013.

[21] Z. Wang, S. Luo, H. Song, W. Deng, and W. Li, "Simulation of microstructure during laser rapid forming solidification based on cellular automaton," *Mathematical Problems in Engineering*, vol. 2014, Article ID 627528, 9 pages, 2014.

[22] M. Eibinger, T. Zahel, T. Ganner, H. Plank, and B. Nidetzky, "Cellular automata modeling depicts degradation of cellulosic material by a cellulase system with single-molecule resolution," *Biotechnology for Biofuels*, vol. 9, no. 1, p. 56, 2016.

[23] X. Li, X. Yan, X. Li, and J. Wang, "Using cellular automata to investigate pedestrian conflicts with vehicles in crosswalk at signalized intersection," *Discrete Dynamics in Nature and Society*, vol. 2012, Article ID 287502, 16 pages, 2012.

[24] M. E. Larraga and L. Alvarez-Icaza, "Cellular automata model for traffic flow with safe driving conditions," *Chinese Physics B*, vol. 23, no. 5, article 050701, 2014.

[25] M. Bukacek and P. Hrabak, "Boundary induced phase transition in cellular automata models of pedestrian flow," *Journal of Cellular Automata*, vol. 11, pp. 327–338, 2016.

[26] H. Li, C. Shao, H. Wu, J. Tian, and Y. Zhang, "Cellular automata approach for modeling lane changing execution," *Journal of Cellular Automata*, vol. 11, pp. 339–350, 2016.

[27] H. C. Shen and W. P. Yan, "Modelling autonomous assembly systems and fms using cellular automata," *The International Journal of Advanced Manufacturing Technology*, vol. 7, pp. 333–338, 1992.

[28] H. C. Shen, H. L. Chau, and K. K. Wong, "An extended cellular automaton model for flexible manufacturing systems," *The International Journal of Advanced Manufacturing Technology*, vol. 11, no. 4, pp. 258–266, 1996.

[29] S. Minami, K. F. Pahng, M. J. Jakiela, and A. Srivastava, "A cellular automata representation for assembly simulation and sequence generation," in *Proceedings. IEEE International Symposium on Assembly and Task Planning*, pp. 56–65, Pittsburgh, PA, USA, August 1995.

[30] I. Barragán, J. C. Seck Tuoh, and J. Medina, "Relationship between Petri nets and cellular automata for the analysis of flexible manufacturing systems," in *Advances in Computational Intelligence. MICAI 2012*, I. Batyrshin and M. Gonzlez, Eds., vol. 7630 of Lecture Notes in Computer Science, pp. 338–349, Springer, Berlin, Heidelberg, 2013.

[31] Z. Li and M. Zhou, *Deadlock Resolution in Automated Manufacturing Systems: A Novel Petri Net Approach*, Springer-Verlag, London, 2009.

[32] W. Nai Qui and M. Zhou, Eds., *Systems Modeling and Control with Resource-Orientes Petri Nets*, CRC Press, Boca Raton, Fl, USA, 2010.

[33] M. P. Cabasino, M. Dotoli, and C. Seatzu, "Modelling manufacturing systems with place/transition nets and timed Petri nets," in *Formal Methods in Manufacturing*, J. Campos, C. Seatzu, and X. Xie, Eds., pp. 3–27, CRC Press, 2014.

[34] P. J. O'Grady, Ed., *Controlling Automated Manufacturing Systems*, Springer, Dordrecht, 1986.

[35] H. K. Shivanand, *Flexible Manufacturing System*, New Age International, 2006.

[36] P. M. Swamidass, Ed.P. M. Swamidass, Ed., "Automated Manufacturing System," in *Encyclopedia of Production and Manufacturing Management*, pp. 50-51, Springer, Boston, MA, USA, 2000.

[37] W. C. Yeh, "Real-time deadlock detection and recovery for automated manufacturing systems," *The International Journal of Advanced Manufacturing Technology*, vol. 20, no. 10, pp. 780–786, 2002.

[38] A. Mitra and H. N. Teodorescu, "Detailed analysis of equal length cellular automata with fixed boundaries," *Journal of Cellular Automata*, vol. 11, no. 5-6, pp. 425–448, 2016.

[39] S. Wolfram, "Computation theory of cellular automata," *Communications in Mathematical Physics*, vol. 96, no. 1, pp. 15–57, 1984.

[40] L. D'Alotto, "A classification of one-dimensional cellular automata using infinite computations," *Applied Mathematics and Computation*, vol. 255, pp. 15–24, 2015.

[41] H. V. McIntosh, *One-Dimensional Cellular Automata*, Luniver Press, London, 2009.

[42] W. Jin and F. Chen, "Topological chaos of universal elementary cellular automata rule," *Nonlinear Dynamics*, vol. 63, no. 1-2, pp. 217–222, 2011.

[43] Z. Bie, Q. Han, C. Liu, J. Huang, L. Song, and Y. Pei, "Chaotic behavior of one-dimensional cellular automata rule 24," *Discrete Dynamics in Nature and Society*, vol. 2014, Article ID 304297, 8 pages, 2014.

[44] S. Wolfram, *Theory and Applications of Cellular Automata, Volume 1 of Advanced Series on Complex Systems*, World Scientific, Philadelphia, 1986.