Efficient Simulation Method for General Assembly Systems with Material handling Based on Aggregated Event-Scheduling

Yanjia Zhao, *Student Member*, *IEEE*, Chao-Bo Yan, *Student Member*, *IEEE*, Qianchuan Zhao, *Senior Member*, *IEEE*, Ningjian Huang, Jingshan Li, *Senior Member*, *IEEE*, Xiaohong Guan, *Fellow*, *IEEE*

***Abstract*—Performance evaluation of complex manufacturing systems is challenging due to problem complexity, uncertainties, etc. In many cases, simulation is an effective approach for modeling and analysis. For an assembly system with material handling, e.g., general assembly lines in automotive industry, the material handling has great impact on system performance; however, it increases the complexity in modeling and simulation. In this paper, we focus on developing an efficient simulation method for a general assembly system with material handling, where traditional simulation methods may suffer from computation intensity. Making use of the partial system decomposability, we introduce an aggregated event-scheduling simulation method with two-level framework. A dividing mechanism with boundary conditions is employed in top-level simulation to divide the global event list into small sizes. A timing-focuses strategy based on max-plus algebra is applied in bottom-level local simulation to further reduce local event lists. With this new method it is possible to mimic real production systems fast and accurately within a reasonable computational time frame. The effectiveness and efficiency of the new simulation method are validated through experimental results.**

***Note to Practitioners*—In manufacturing systems with material handling (e.g., general assembly lines in automotive industry), material handling has great impact on system performance. It also increases system complexity and in turn causes difficulties and challenges in modeling and simulation. In this paper, we focus on developing an efficient simulation approach for a general assembly system with material handling. An aggregated event-scheduling simulation method with two-level framework (a dividing mechanism with boundary conditions for top level and a timing focus strategy based on max plus algebra for bottom level) is introduced to reduce the size of event list. With this new method it is possible to mimic real production fast and accurately. The experimental results suggest that the method provides an efficient way for simulation modeling and analysis of assembly system with material handling.**

***Index Terms*—Discrete event dynamic system, Simulation, General assembly systems, Material handling, Aggregated event-scheduling**

This work was supported by a contract between Tsinghua University and General Motors Corporation. Qianchuan Zhao and Xiaohong Guan received additional support from NSFC (60574067, 60736027, 60721003), 863 High Tech Development Plan (2007AA04Z154), NCET program (NCET-04-0094), and the Program of Introducing Talents of Discipline to Universities (the 111 International Collaboration Project of China). Partial of the work was reported in the 4th *IEEE Conference on Automation Science and Engineering*, 2008.

Yanjia Zhao, Chao-Bo Yan, Qianchuan Zhao, and Xiaohong Guan are with Center for Intelligent and Networked Systems, Department of Automation, TNList, Tsinghua University, Beijing 100084, China.

Ningjian Huang is with Research & Development Center, General Motors Corp. Warren, MI 48090, US.

Jingshan Li is with Department of Electrical and Computer Engineering and Center for Manufacturing, University of Kentucky, Lexington, KY 40506, US.

Corresponding author is Prof. Qianchuan Zhao (zhaoqc@tsinghua.edu.cn).

Abbreviations & Notation

GA	General Assembly
MH	Material Handling
DEDS	Discrete Event Dynamic System
$\mathcal{I}$, I	set and total number of sections in a GA line.
$\mathcal{I}_s$, $\mathcal{I}_m$	sets of serial sections and merging sections in a GA line.
$\mathcal{J}_i$	set of stations in section i, $\forall i \in \mathcal{I}$.
J_i	total number of stations in section i, $\forall i \in \mathcal{I}$.
J	total number of all stations.
$\mathcal{L}$	set of lineside buffers in a GA line with MH.
$\mathcal{L}_{ij}$	set of lineside buffers belonging to station j of section i, $\forall i \in \mathcal{I}$, $\forall j \in \mathcal{J}_i$.
$\mathcal{D}$	set of drivers of an MH system.
$\mathcal{L}(d)$	set of lineside buffers supplied by driver d, $\forall d \in \mathcal{D}$.
τ_{ij}	cycle time of station j in section i, $\forall i \in \mathcal{I}$, $\forall j \in \mathcal{J}_i$.
E_i	index of the station where a merging section joins into serial section i, $\forall i \in \mathcal{I}_s$.
$S_i(t)$	state of section i at time t. *up*, *down*, and *idle* are indicated by $S_i(t) = 1$, -1 and 0, respectively.
$s_{ij}(t)$	state of station j in section i at time t. *up*, *down*, and *idle* are indicated by $s_{ij}(t) = 1$, -1 and 0, respectively.
λ_{ij}, μ_{ij}	the reciprocal of *Mean Time Between Failures* (MTBF) and the *Mean Time To Repair* (MTTR) of station j in section i, respectively.
$Cm_{ij}(t)$	accumulated number of vehicles processed on station j of section i before time t, $\forall i \in \mathcal{I}$, $j \in \mathcal{J}_i$.
$Cc_i(t)$	accumulated number of conveyor moves in section i before time t, $\forall i \in \mathcal{I}$.
$b_i(t)$	inventory level of section buffer i at time t.
N_i	capacity of section buffer i.
$Ca_i(t)$	accumulated number of vehicles arrived at section buffer i before time t.
$Cd_i(t)$	accumulated number of vehicles departed from section buffer i before time t.
$y_d(t)$	state of driver d at time t. $y_d(t) = 0$, if driver d is idle; $y_d(t) = l$, if driver d is on the trip to supply lineside buffer l.
$a_d(t)$	action of driver d at time t. $a_d(t) = l$, if driver d decides to supply lineside buffer l; $a_d(t) = 0$, if no action is taken.
$Lb_l(t)$	inventory level of lineside buffer l at time t.

TP throughput of the GA line, (Job/hour).
WIP_i work-in process inventory of section buffer i.
H_l parts inventory of lineside buffer l.
U_d utilization of driver d.
$Te_d(u)$ the time driver d finishes the u-th trip and comes back to central docking area.
$To_d(u)$ the time driver d is sent out for the u-th supplying trip.
$Cs_d(t)$ accumulated number of trips of driver d before time t.
ξ a random sequence.
T total simulation time.
M_i total number of conveyor moves in section i within the given total simulation time T.
R_i total number of local simulation *rounds* of section i.
m_{ir} number of conveyor moves during the r-th round of local simulation of section i.
t_{ir} starting time of r-th round local simulation of section i.
e_{ir} ending time of r-th round local simulation of section i.
$Cr_l(t)$ accumulated number of parts replenished into lineside buffer l before time t.
$Cu_l(t)$ accumulated number of parts used at lineside buffer l before time t.
$P_l(k)$ number of parts consumed by the k-th vehicle at lineside buffer l.
$Ta_i(k)$ arrival time of vehicle k at section buffer i.
$Td_i(k)$ departure time of vehicle k at section buffer i.
$Tr_l(p)$ replenishment time of part p at lineside buffer l.
$Tc_i(m)$ time of the m-th conveyor-move in section i.
$Tf_{ij}(k)$ finish time for processing vehicle k at station j in section i.
$Tu_l(p)$ the time of part p being consumed at lineside buffer l.
$Ts_{ij}(k)$ starting time for processing vehicle k at station j in section i.
$Tp_{ij}(k)$ processing duration of vehicle k at station j in section i.
T_l round travel time from central docking area to lineside buffer l and then back to central docking area.
Q_l package size for lineside buffer l, i.e., the number of parts supplied to lineside buffer l in each delivery.

I. Introduction

Material Handling (MH) is an important element in vehicle production [4]. In modern automotive assembly plants, material handling system and assembly processes are closely related. Timely and accurate delivery of necessary material to the assembly line is essential to achieve high productivity and quality. It is claimed in [5] that 20-50% of the manufacturing costs may be related to material handling. Therefore, efficient and precise analysis of system performance in general assembly lines with material handling is necessary and important.

In automotive assembly plant, a general assembly line is typically structured as follows (see Figure 1) [1-3].

A *main production line* is composed of a series of *section*s, with each pair being separated by *in-process buffer*s (or, *section buffer*s). Typically, such sections include trims, chassis, door line, final assembly and inspection. Within each section, consecutive serial *station*s are connected through a paced-moving conveyor [2][3]. The sections are asynchronous in nature so that each is moving independently. However, the stations within each section are synchronized so that all stations in this section stop working if any of them stops. In addition, to increase productivity, some *component*s (e.g., doors, powertrain, suspension, etc.) may be pre-assembled in the *subassembly line*s before merged into the main line. The *vehicles* flow into the system on the moving conveyors, passing through all the stations. Each station has *lineside buffer*s supplying parts needed for assembly. At each station, operators load the required parts from the lineside buffers and assemble them on the vehicles. The parts to replenish the lineside buffers are delivered by the material handling (MH) workers from central docking area based on the production schedule and delivery policy using dollies, tuggers or fork-lifts.

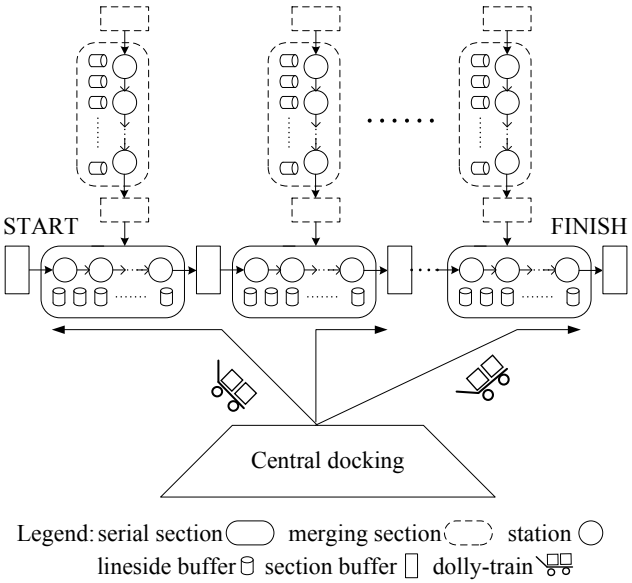

Fig. 1. Sketched General Assembly system with Material handling.

As one can see, besides machine breakdowns, a station could also suffer from blockages and starvations due to interactions among up- or downstream stations. In addition, a station will also be starved and stop moving if the necessary parts in the lineside buffers are not available. Clearly, larger inventories in the lineside buffers could keep production from being starved due to parts and make the analysis of the general assembly lines simple (by omitting the material handling). However, this is not pursuable due to cost, quality, and floor space constraints. Lean lineside inventories are more desirable. How to ensure a smooth production with material handling system in general assembly lines is an important question.

Analysis of production lines has attracted substantial attention from both research and practice during the last forty years and significant improvement has been achieved (see, for instance, review [9], monographs [1][6]). Most of these emphasize on serial lines and assembly systems. However, it is typically assumed that material handling is not an issue or has minimum impact so that the line always has sufficient materials

and will not be starved by parts. In some cases, the effect of part shortage is approximated by reducing station reliability. However, such approximation does not fully address the interactions between material delivery and vehicle assembly.

Including material handling into the analysis is challenging, since in addition to the traditional difficulties, such as randomness and nonlinearity of production, interactions between stations, asynchronous and synchronous natures between and within sections, etc., more difficulties arise due to the inconsistency and coupling between material handling and general assembly. Specifically, the inconsistency may include

1) Assembly (or production) process is carried out in the time scale of cycles (e.g., minutes), while material delivery is based on shipping schedule in the time scale of many cycles (e.g., hours).
2) Assembly process is implemented in the unit of individual vehicle, but material handling transports parts in batches (e.g., carts, boxes).
3) Moreover, in assembly process, each vehicle may need multiple types of parts with various quantities at each station, depending on the customer options.

The coupling between material handling and general assembly exists since

1) The lack of one particular part may result in the stoppage of the station and the whole section, and propagate to the sections up- and downstream.
2) The material handling worker may deliver parts to many stations (may not belong to a single section), which introduces global correlations among those stations and sections.
3) The change of control strategy in material delivery may interrupt the assembly process and introduce more transient phenomenon. Therefore, the traditional steady state analysis may not be able to address.

Due to the above complexities and coupling issues, analytical evaluation of system performance for general assembly line with material handling seems impossible. Therefore, simulation approach is pursued to analyze the system behavior. However, the traditional discrete-event simulation may suffer from long simulation time and computation intensity for large scales. Thus, developing an efficient simulation method to provide accurate and quick solutions of performance evaluation is of importance. The goal of this paper is intended to contribute to this end.

The main contribution of this paper is the development of a new efficient simulation method for general assembly lines with material handling in automotive manufacturing. Specifically, a method based on aggregated even-scheduling is proposed with the following salient features:

1) Dividing mechanism. By making use of partial decomposability, a novel dividing mechanism with boundary conditions is introduced to aggregate the simulation for each section into rounds.
2) Reduced event list. With a timing-focused simulation strategy based on max-plus algebra, many redundant events are omitted and only a small key-event list is kept in order to improve simulation efficiency.
3) Potential parallelism. It is possible to extend the new method for parallel and distributed simulation. Since the simulation framework is divided into two levels, individual sections of the GA line can be simulated in parallel and synchronized through the global system states and events.

With these salient features, the aggregated event-scheduling simulation method is possible to simulate actual production systems accurately enough with reasonable computational times. Numerical experiments and case studies suggest that such method results in efficient and effective analyses of general assembly line with material handling. Additionally, this method is also applicable to other manufacturing systems, such as the disassembly line, and systems with fork/join structures, such as parallel processing systems and distributed replicated database systems [48][49].

The remaining part of the paper is organized as follows. Section II presents literature reviews on analysis and simulation approaches for production systems and discrete event dynamic systems. Section III provides mathematical descriptions of the system. Section IV presents the aggregated event-scheduling simulation method. Numerical results are illustrated in Section V to demonstrate the effectiveness and efficiency of the new method and followed by Conclusions.

II. Literature Review

A General Assembly (GA) line is a typical type of manufacturing system. As one of the few ways that wealth is created, manufacturing system has attracted much research interest over the past two decades. These researches can be roughly categorized into two classes: the system design problem and the scheduling and real-time decision making problem, as surveyed in [1][6]. The MH system usually takes charge of the delivery of both raw parts and finished parts, and it consists of loading and unloading mechanisms, transfer mechanisms, and internal storage facilities, etc [4]. MH is widely required in various segments of manufacturing industries, and it is believed that the design of MH systems has great impact on the overall manufacturing process cost and efficiency [5]. The research of a material handling system can be similarly divided into two aspects: the design problem and the operation problem, as surveyed in [7].

Although there are plenty of researches on the GA lines and MH systems individually, it still lacks of analytical results for the problem combining these two systems together in a specific way. Since the GA line with MH system does not satisfy Markovian assumptions for processing time and routing probabilities in queuing network, it is not a product form queuing network which allows analytical solutions. Even more general Markov chain and Semi Markov Chain [8] models could not be applied directly to describe this system and obtain analytical solutions, since the time intervals between events (state changing) do not follow the exponential distribution and sections are asynchronous with different cycle times. Approximation and empirical approaches such as aggregations

[1][9] and decomposition [6], may not provide accurate and detailed information in all cases. Furthermore, when dealing with production control laws and material handling strategies, there are no analytical tools available. Thus, simulation is one of the most important approaches available in this case. To this end, this paper focuses on developing an efficient simulation approach for the GA line with MH illustrated in Fig. 1.

Simulation has been widely and successfully applied in the design and optimization of DEDS, especially manufacturing systems [8][10][12][13]. For example, [14]-[20] develop simulation models for different types of manufacturing systems to optimize the design and control strategies. [2][3][21]-[24] apply simulations for automotive production lines to investigate system parameters and test various hypotheses. [25]-[29] simulate material handling systems of flexible manufacturing systems, such as semiconductor fabs, to evaluate various designs and scheduling rules. [38] investigates optimal dispatching policies in material handling systems of general assembly lines based on the efficient simulation method developed in this paper. These successful applications demonstrate the advantages of simulations, i.e., flexibility, time compression, physical scaling, and risk avoidance [10][11].

Conceptually, all these discrete event simulation models mentioned above can be exactly formulated as the Generalized Semi-Markov Process (GSMP) mathematically. The evolution of a GSMP is governed by its state transition function. Different ways of doing simulation can be viewed as different ways of update system states. Since the state space is extremely large for the system under consideration, one has to avoid to explicitly write out and store the state transition function as a table for the GSMP as is done in event scheduling scheme [10][12], instead one has to define it implicitly by exploring the system structure so that efficient simulation can be done. Below we will focus on existing simulation approaches of DEDS and point out their limitations when applied directly to the system we concern.

Despite of great diversities of real-world systems, the *event scheduling scheme* (also known as the *next-event time-advance approach*) [10][12] provides a general simulation approach for DEDS [8]. In this approach, a simulation clock and an event list are introduced. A timing routine is invoked to determine which event in the event list will occur next. The simulation clock is advanced to the time when that event happens and an event routine is invoked to update the system state and to generate future events. The procedure is repeated until the stopping condition is eventually satisfied. Dozens of successful software packages have been developed based on this approach, such as Arena, AutoMod, and ProModel, etc. However, handling an event list with huge size makes this approach extremely difficult to be applied for large scale practical problems.

Unlike the event scheduling approach, the max-plus algebra (min-max algebra) introduces another approach to model DEDS [30]-[32]. It studies system behaviors through recursion formulas based on the timing-logic and can thus handle a large number of events efficiently. However, this approach is proper to deal with the systems with deterministic events together with the decision-free requirement. It is not applicable for the systems with random failures and state transitions depending on both upstream and downstream stations within a section. Therefore, the max-plus algebra approach could not be applied directly to simulate the GA line with MH.

Parallel and distributed simulation is another relevant research area. Four synchronization mechanisms, conservative, optimistic, hybrid, and adaptive, are widely investigated, such as in [33]-[35]; performance analysis and applications of these mechanism and corresponding protocols are studied, such as in [44]-[47]. An obvious benefit of parallel simulation is that computational times can be reduced by dividing a large simulation task into many sub-tasks that can be executed concurrently. For discrete event systems, many advances have been made (see e.g., [39][40] and [42]) which provide us many insights including cutting the system into pieces at buffers (queues), introduction of asynchronous protocols to develop our simulation strategy. To successfully harvest the benefit of parallel simulation, one has to carefully divide the simulation task so that the overhead caused by interaction among sub-tasks[1] is not a major problem. For discrete event systems with relatively simple structure, the division is not very hard: for example, for the FCFS queuing networks with infinite buffers, one can follow [40] to regard the simulation of each node (every queuing) as a sub-task and handle the interaction among nodes by inserting arrival events to the queue of every node properly.

However, our problem is more challenging. Comparing with existing parallel and distributed discrete event simulation problems [39][42][43], the simulation of the GA line with MH is much complicated, considering the facts that stations in each section are strongly coupled and synchronized by the paced-moving conveyor in the GA line; the MH system introduces global coupling among sections since drivers are shared by difference sections when supplying parts. These challenges make the division of the simulation task and the design of interaction mechanism a non-trivial problem. Detailed discussion can be found in Section IV-D.

To summarize, it is desirable to develop a new effective simulation method for the GA line with MH systems by investigating its structure characteristics. This research is with significant theoretical importance and extended application backgrounds.

III. System Descriptions

This section presents formally the general assembly line with material handling under consideration, in order to specify the system characteristics.

A. Overview of the System

The following notation is firstly introduced to denote the GA line with MH system. Sections in the main line are named as *serial sections*, and sections in the subassembly line are named as *merging sections*. Suppose the system consists of I serial sections and I merging sections,

[1] Since some of the events have to be synchronized at certain points.

$$\mathcal{I}_s \cup \mathcal{I}_m = \mathcal{I}, \mathcal{I}_s \cap \mathcal{I}_m = \phi, \quad (1)$$

$$|\mathcal{I}_s| = |\mathcal{I}_m| = I, \quad (2)$$

where $\mathcal{I}$ denotes the set of sections in the GA line; $\mathcal{I}_s$ and $\mathcal{I}_m$ denote the sets of serial sections and merging sections in the GA line, respectively. Hereafter, we suppose $\mathcal{I}_s = \{1, 2, \ldots, I\}$, $\mathcal{I}_m = \{1', 2', \ldots, I'\}$.

Based on the sketched layout of the entire system illustrated in Fig. 1, we redraw detailed structure of one section in the GA line in the figure below. As Fig. 2 shows, we use the following methods to indicate the connection relationships of the GA line: the *input buffer* of serial section *i*, i.e., the section buffer before serial section *i*, is denoted as buffer *i*-1; the *output buffer* of serial section *i*, i.e., the section buffer after serial section *i*, is denoted as buffer *i*; the merging section joining into serial section *i* is indexed as *i'*, which is also used to indicate the index of section buffers in between.

Section *i*, $\forall i \in \mathcal{I}$, consists of a paced-moving conveyor and a series of consecutive stations. Vehicles are transferred on the conveyor step by step, going through all the stations where various parts are assembled onto the vehicle. The total number of stations in section *i* is J_i, i.e.,

$$|\mathcal{J}_i| = J_i, \quad (3)$$

where $\mathcal{J}_i = \{1, 2, \cdots, J_i\}$ denotes the set of stations in section *i*. Note that we use $j = 1, 2, \ldots, J_i$ to indicate the consecutive order of stations in section *i*. For example, index 1 denotes the first station and index J_i is the last one in section *i*.

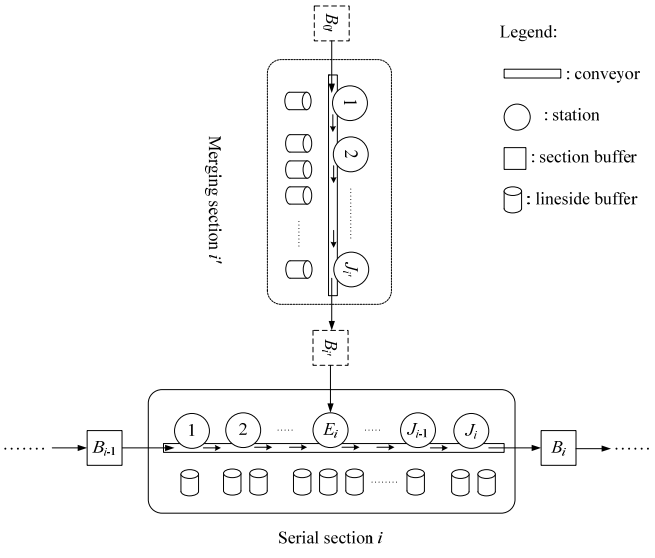

Fig. 2. Detailed structure of one section in the general assembly line.

Each driver in the MH system takes charge of a given set of lineside buffers and there are no overlaps,

$$\mathcal{L} = \bigcup_{i \in \mathcal{I}} \bigcup_{j \in \mathcal{J}_i} \mathcal{L}_{ij}, \quad (4)$$

$$\mathcal{L}(d) \cap \mathcal{L}(d') = \phi, \forall d \neq d', d, d' \in \mathcal{D}, \quad (5)$$

$$\bigcup_{d \in \mathcal{D}} \mathcal{L}(d) = \mathcal{L}, \quad (6)$$

where $\mathcal{L}$ denotes the set of lineside buffers in the entire system; $\mathcal{L}_{ij}$ is the set of lineside buffers belonging to station *j* in section *i*; $\mathcal{D}$ indicates the set of drivers of the MH system; $\mathcal{L}(d)$ is the set of lineside buffers supplied by driver *d*. (Note that it is the current setting in practical systems that responsibilities of drivers are fixed and distinguished. However, the simulation method developed in this paper could also handle the scenario if all drivers are pooled and supply all lineside buffers based on availability. Numerical test for this alternative setting is shown in Section V-C.)

To simplify the explanation, we make the following assumptions based on literature traditions and industry requirements.

1) Buffers have finite capacities.
2) All stations have random reliabilities. The time between failures and the time to repair for each station are exponentially distributed. The exponential reliabilities are widely assumed in studies of manufacturing systems [1][6]; the effectiveness of this assumption is evaluated by [36]. Note that the simulation method developed in this paper is also applicable to other distributions without memoryless property by augmenting the station state with accumulated working time and repairing time.
3) The *cycle time* of station *j* in section *i*, i.e., the time necessary to process a vehicle by a station, is deterministic and denoted as τ_{ij}. Stations in the same section have the same constant cycle time, i.e.
$$\tau_{ij} = \tau_{ij'} = \tau_i, \forall j, j' \in \mathcal{J}_i, \quad (7)$$
where τ_i is defined as the cycle time of section *i*. Sections have different cycle time in general, i.e.,
$$\tau_i \neq \tau_{i'}, \ \forall i, i' \in \mathcal{I}, i \neq i'. \quad (8)$$
Thus, dynamics of sections are asynchronous.
4) The conveyor will move as long as there is at least one vehicle in the section and also stations have no failures and the output buffer is not full.
5) The vehicle at the beginning buffer, B_0 and B_0's, are always ready and the finished good buffer B_I is infinite. The numbers of all parts in the central docking area are infinite.

B. Mathematical Description of Stations

For station *j* in section *i*, $\forall i \in \mathcal{I}, j \in \mathcal{J}_i$, it has three states at time *t*, *up*, *down* and *idle*, which is denoted as $s_{ij}(t)$. The station is *up*, if it is busy with a vehicle under assembling at time *t*, denoted as $s_{ij}(t) = 1$; the station is *down*, if it is broken down and in repair at time *t*, denoted as $s_{ij}(t) = -1$; otherwise the station is *idle*, denoted as $s_{ij}(t) = 0$. The idle state contains three cases: being empty, or holding a vehicle to output, or starved by parts at some lineside buffer belonging to it.

The time duration of the station being *up*, i.e., the accumulated working time, is defined as *uptime*; the time duration of the station in failure is defined as *downtime*. The stochastic reliability model for the stations in the system is depicted with exponential distributions of uptime and

downtime. The probability density functions of the up- and downtime of station j in section i at time t are given by:

$$f_{up}(t) = \lambda_{ij} e^{-\lambda_{ij} t},\ t \geq 0, \tag{9}$$

$$f_{down}(t) = \mu_{ij} e^{-\mu_{ij} t},\ t \geq 0, \tag{10}$$

where the reciprocal of rates λ_{ij} and μ_{ij} are referred as *Mean Time Between Failures* (MTBF) and *Mean Time To Repair* (MTTR) of the station in industries.

As a dynamical system, the transition diagram of the above exponential reliability model is: if up, the station j in section i may go down in each infinitesimal interval δt with probability $\lambda_{ij}\delta t$; if down, it may go up during δt with probability $\mu_{ij}\delta t$. It should be point out that we focus on the *operation-dependent failure* [1] in this paper, i.e., station breakdowns cannot occur while it is in idle state. Thus, the uptime discussed above does not contain the idle time.

C. Mathematical Description of Sections

For section i, $\forall i \in \mathcal{I}$, it has three states at time t, *up*, *down* and *idle*, which is denoted as $S_i(t)$. The section is *up*, if it is working on at least one vehicle at time t, denoted as $S_i(t) = 1$; the section is *down*, if at least one of its stations is in failure at time t, denoted as $S_i(t) = -1$; otherwise the section is *idle*, denoted as $S_i(t) = 0$. The state of section i is determined by the states of its stations as follows.

$$S_i(t) = \begin{cases} 1 & \text{up:} \quad \forall j \in \mathcal{J}_i, s_{ij}(t) \geq 0;\ \exists j' \in \mathcal{J}_i, s_{ij'}(t) = 1 \\ -1 & \text{down:} \ \exists j \in \mathcal{J}_i, s_{ij}(t) = -1 \\ 0 & \text{idle:} \quad \text{otherwise} \end{cases} \tag{11}$$

The state of section buffer i is depicted by its inventory level at time t, which is denoted as $b_i(t)$. We have

$$b_i(t) = Ca_i(t) - Cd_i(t). \tag{12}$$

where $Ca_i(t)$ denotes the accumulated number of vehicles arrived at section buffer i before time t; $Cd_i(t)$ indicates the accumulated number of vehicles departed from section buffer i before time t. Obviously, $Cd_i(t) \leq Ca_i(t)$.

Section i is in idle state due to *starvation* or *blockage* of vehicles from upstream or downstream, or *starvation* of components from merging section *i'*. In other words, section i is idle if the input buffer is empty, $b_{i\text{-}1}(t) = 0$, or the output buffer is full, $b_i(t) = N_i$, or the merging section buffer is empty, $b_{i'}(t) = 0$.

D. Mathematical Description of Drivers

The state of driver d at time t is denoted by $y_d(t)$, $d \in \mathcal{D}$, in the following sense: $y_d(t) = 0$ if driver d is idle at central docking area at time t; $y_d(t) = l$, $l \in \mathcal{L}(d)$, if driver d is on the trip to supply lineside buffer l at time t. The material handling strategy π determines the action for each driver at any time instance when the driver is idle, i.e.,

$$a_d(t) = \begin{cases} \pi_d(t, Lb_l(t), \forall l \in \mathcal{L}(d)), \text{if } y_d(t) = 0 \\ 0 \qquad\qquad\qquad\qquad\quad , \text{otherwise} \end{cases}. \tag{13}$$

where $\pi_d(t)$ is a mapping from the lineside buffer inventories to the driver action; $a_d(t) \in \mathcal{L}(d) \cup \{0\}$. Note that in each trip, a driver can supply only one lineside buffer and the delivery package size is predetermined. The action $a_d(t)$ for driver d at time t is defined as: $a_d(t) = l$, $l \in \mathcal{L}(d)$, if the driver decides to supply lineside buffer l at time t; $a_d(t) = 0$, if the driver decides to take no action at time t.

E. Performance Measures

Based on the above mathematical descriptions for the stations and sections, we want to evaluate the following performance measures through simulations.

1) *Throughput*. The average number of vehicles produced by the GA line per time unit in the steady state of system operation, which can be defined as the average arrival number of vehicles at section buffer I, i.e. the finished goods buffer at the end of the line [1]:

$$TP = \lim_{T\to\infty} \frac{1}{T} \cdot Ca_I(T). \tag{14}$$

2) *Work-in process inventory* of section buffer i. The average number of vehicles contained in section buffer i of the GA line in the steady state of its operation. It can be defined as:

$$WIP_i = \lim_{T\to\infty} \frac{1}{T} \int_{t=0}^{T} b_i(t)dt, \quad \forall i \in \mathcal{I}. \tag{15}$$

3) *Parts inventory* of lineside buffer l. The average number of parts contained in lineside buffer l of the GA line in the steady state of its operation. It can be defined as:

$$H_l = \lim_{T\to\infty} \frac{1}{T} \int_{t=0}^{T} Lb_l(t)dt, \quad \forall l \in \mathcal{L}, \tag{16}$$

where $Lb_l(t)$ denotes the inventory level of lineside buffer l at time t.

4) *Utilization* of driver d. The average ratio of working time over the total time for driver d in steady states. It can be defined as:

$$U_d = \lim_{T\to\infty} \frac{1}{T} \sum_{u=1}^{Cs_d(T)} \left[Te_d(u) - To_d(u) \right], \quad \forall d \in \mathcal{D}, \tag{17}$$

where $Te_d(u)$ indicates the time driver d ends the u-th trip; $To_d(u)$ denotes the time driver d is sent out for the u-th trip; $Cs_d(T)$ is the accumulated number of trips of driver d before time T.

As we mentioned in the introduction, there is a need to evaluate the above system performances in design and analysis of the GA line with MH systems. However, they could not be evaluated through closed form equations due to stochastic station reliabilities, complicated system dynamics, and various material handling strategies. Thus simulation is almost the only way to obtain the system performances under different designs.

To do efficient simulation is not an easy task. Traditional simulation methods, such as event-scheduling method, are not efficient to satisfy the requirement in practice due to large problem scales. The practical system we concern consists of multiple sections, hundreds of stations, hundreds of lineside buffers and dozens of drivers. Each item is related to several different types of events, which makes the total number of events extraordinarily large. Furthermore, the state space of the whole system could be so large that it is impossible to be described by traditional simulation approaches. For example,

each station has three states, up, down or idle. Combined together, the state space for a section with 20 stations will reach $3^{20} \approx 10^9$. Thus, it is desirable to develop a new method, i.e., the aggregated event-scheduling method, in this paper.

IV. Aggregated Event-Scheduling Method

To address the challenges of simulating the general assembly lines with material handling, this section presents a novel aggregated event-scheduling method, which combines the ideas of parallel simulation and event-scheduling together by developing a two-level simulation framework. A dividing mechanism with certain boundary conditions is introduced on the top-level to decouple the problem; a timing focused simulation strategy is applied on the bottom-level to further reduce the sizes of local event lists.

A. Two-Level Framework of Aggregated Event-Scheduling

Discrete-event simulation concerns the modeling of a system evolving over time by a representation in which system states change instantaneously at separate time points [10][12]. These time points denote the time when events happen, where an *event* is defined as an instantaneous occurrence that may change the system state. The purpose of discrete-event simulation is to find these time points for all the events with a given random sequence and initial states, i.e., to depict a simulation trajectory (sample path) of the system within given simulation time. For the GA line with MH system, a simulation *trajectory* is defined as follows:

$$\{\xi, S_i(t), b_i(t), s_{ij}(t), \forall i \in \mathcal{I}, j \in \mathcal{J}_i, \forall 0 \le t \le T\}, \quad (18)$$

where ξ denotes a random sequence; T is the total simulation time; $S_i(0)$, $s_{ij}(0)$, $b_i(0)$ are components of system state[2] (called state variables) with initial state variables $S_i(0)$, $s_{ij}(0)$, $b_i(0)$ being given. With a simulation trajectory sufficient long, we can evaluate system performances shown in (14) - (17).

Considering the challenge of large problem scale, it is not efficient to apply traditional event-scheduling method directly to simulate the GA line with MH system since it requires maintaining a comprehensive event list, the updates of which may depend on all state variables of the entire system. To address this difficulty, the paper employs the idea of divide-and-conquer and develops a novel aggregated event scheduling method. Although general parallel simulation cannot be applied directly due to the global correlations, there are still possibilities to decouple the system based on partial decomposability. Here the *decomposability* means that with a simulation technique, a complex system can be divided into several connected subsystems; under certain conditions, each subsystem can be simulated independently with only limited data exchange with others. The GA line with MH illustrated in Fig. 2 consists of connected sections, and yet they are separable for the purpose of simulation.

The aggregated event-scheduling method has a two-level framework as Fig. 3 illustrates.

[2] It should be clear that as in GSMP model, strictly speaking, these state variables are only *discrete* state variables and the lifetime of events are also state variables if we fully determine the state transition map.

The top-level simulation follows the same flow as the traditional event scheduling method and it consists of three subroutines: (1) an initialization routine which initializes the simulation clock, system states, statistical counters and event list; (2) a timing routine which determines the next event to happen and advances the simulation clock; (3) an event routine which updates system states. The difference is: the concept of a traditional event has been extended to a "round" of local simulation for a section. To be more specific, the top level introduces a concept of *round* and divides the simulation of each section into several rounds of local simulations. Traditional events are aggregated as rounds of local simulation, each of which can be carried out separately and independently under certain boundary conditions (details will be demonstrated in Section IV-B). Therefore, the timing routine here determines the occurrences of next round of local simulation; the event routine invokes a round of local simulation to update the system state. This cycle is repeated until the stopping condition is eventually satisfied.

On the bottom-level, one round of local simulation for a section or MH system is carried out as an independent module without interactions with other parts of the system. States of section buffers are updated in bottom-level simulation, and the boundary conditions introduced in Section IV-B determine when to stop local simulation and return to the top-level. Local event lists in bottom-level are further reduced with a timing-focused simulation strategy (the details will be explained in Section IV-C), so that simulation efficiency is further improved.

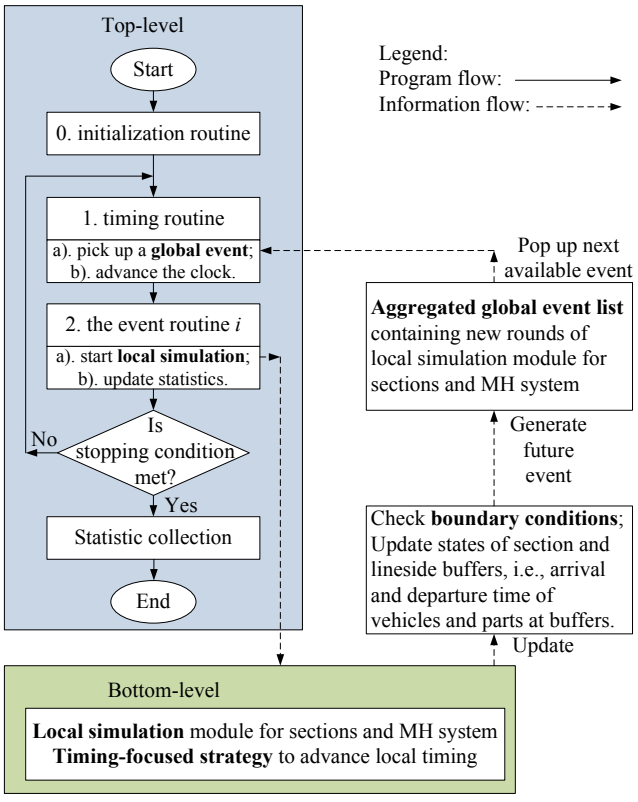

Fig. 3, Framework of the Aggregated Event Scheduling Method

To summarize, the aggregated event-scheduling method takes advantage of the divide-and-conquer idea and divides the simulation framework hierarchically into two levels based on the system partial decomposability. It has benefits as follows.

1) By dividing the simulation for each section into rounds and extending the concept of events as rounds of local simulation, the top-level of the framework keeps the size of the aggregated-event list in a reasonable scale so that the existing event scheduling concept can be extended to simulate large scale systems. A boundary condition is introduced to guarantee the equivalence of the aggregated simulation to the original problem.
2) Local event lists are further reduced with a timing-focused simulation strategy on the bottom-level, which derives timing relations of key events based on max-plus algebra and reduces redundant events.
3) The hierarchical architecture makes the simulation programming more easily and provides potentials to do simulations on parallel or distributed computers, although it is currently implemented on a serial computer.

With this new method it is possible to mimic real production problems fast and accurately within a reasonable computational time frame. Details of these features will be explained in the remaining part of this section.

B. Top-Level: Dividing Mechanism and Boundary Conditions

As the previous subsection explained, the purpose of the simulation for section i is to get its trajectory, which consists of M_i conveyor moves within the given simulation time T. Thus the key of the simulation is to get the time of the m-th conveyor-move in section i, which is denoted as $Tc_i(m)$, $m = 1, 2, \ldots, M_i$. Here we are facing a challenge to do efficient simulation since we cannot get all of the conveyor moves at the same time due to the correlations between different sections. However, the partial decomposability of the system provides a way to do local simulation of a section separately and independently within a short time-window, and obtain starting times of several conveyor moves. In other words, we need to introduce boundary conditions based on the states of section buffers and divide the top-level simulation of each section into rounds, so that simulation can be done alternatively among sections. Even though production of a section continues without stopping in the real situation, we have to stop a round of local simulation for the section if boundary conditions are not satisfied.

1) Notation

In the top-level dividing mechanism, we introduce the concept of *round* and divide the simulation of section i into R_i rounds; the r-th round contains m_{ir} conveyor moves, which can be simulated separately and independently. First we introduce the following notation to explain this mechanism.

M_i total number of conveyor moves in section i within the given total simulation time T.
R_i total number of local simulation *rounds* of section i.
m_{ir} number of conveyor moves during the r-th round of local simulation of section i; we have

$$M_i = \sum_{r=1}^{R_i} m_{ir} \ . \tag{19}$$

t_{ir} starting time of the r-th round local simulation of section i.
e_{ir} ending time of the r-th round local simulation of section i. e_{ir} can be determined by t_{ir} and m_{ir} during the r-th round local simulation of section i.

2) Boundary Conditions

Following similar ideas in our previous work [37], boundary conditions for the dividing mechanism in top-level simulation contain the following three parts. Suppose section i has finished r_i rounds of local simulation. The condition and the time to start the r_i+1-th round local simulation and the number of conveyor moves in the r_i+1-th round local simulation are determined as follows.

(P1). The condition to start the r_i+1-th round local simulation for serial section i, $\forall i \in \mathcal{I}_s$, is determined by the following (20) - (23):

$$Cm_{i1}(e_{ir_i}) + 1 \le Ca_{i-1}(e_{i-1,r_{i-1}}) \,, \tag{20}$$

$$Cm_{iJ_i}(e_{ir_i}) - N_i \le Cd_i(e_{i+1,r_{i+1}}) \,, \tag{21}$$

$$Cm_{iE_i}(e_{ir_i}) + 1 \le Ca_{i'}(e_{i'r_{i'}}) \,, \tag{22}$$

$$Cu_l(e_{ir_i}) + P_l(Cm_{ij}(e_{ir_i}) + 1) \le Cr_l(e_{ir_i}),\ \forall j \in \mathcal{J}_i,\ l \in \mathcal{L}_{ij} \,, \tag{23}$$

where the notation has the following meanings:

$Cm_{ij}(t)$ accumulated number of vehicles processed on station j in section i before time t.
E_i index of the station where merging section i' join into the serial section i. Here we assume that each serial section has one merging section at most.
$Cu_l(t)$ accumulated number of parts used at lineside buffer l before time t.
$P_l(k)$ number of parts used by the k-th vehicle at lineside buffer l .
$Cr_l(t)$ accumulated number of parts replenished at lineside buffer l before time t.

Hereafter, we let $N_I = \infty$ and $Ca_0(t) = \infty$ for all t according to assumption (5) in Section III-A.

The physical meaning of these conditions is intuitive: (20) and (21) guarantee we have sufficient information about the upstream and downstream section buffers of serial section i, i.e., its input buffer is not empty and output buffer is not full; (22) and (23) make sure that the component from the merging section and the parts from all lineside buffers are ready before we carry out the r_i +1-th round local simulation for serial section i.

The condition to start the r_i+1-th round local simulation for merging section i, $i \in \mathcal{I}_m$, is determined by (21)(23), since materials at the input buffers of merging sections are always available and there are no further merges in merging section in the current system we concerned.

(P2). The starting time of the r_i+1-th round local simulation for serial section i, $i \in \mathcal{I}_s$, is determined as follows:

if $s_{ij}(e_{ir_i}) = 0, \forall j \in \mathcal{J}_i$, then

$$t_{i,r_i+1} = \max\left\{e_{ir_i},\ Ta_{i-1}(Cm_{i1}(e_{ir_i}) + 1)\right\} \,;$$

otherwise,

$$t_{i,r_i+1} = \max\left\{\begin{array}{l} e_{ir_i}, Td_i(Cm_{iJ_i}(e_{ir_i}) - N_i), Ta_{i'}(Cm_{iE_i}(e_{ir_i}) + 1), \\ \{Tr_l(Cu_l(e_{ir_i}) + P_l(Cm_{ij}(e_{ir_i}) + 1)), \forall j \in \mathcal{J}_i, l \in \mathcal{L}_{ij}\} \end{array}\right\} \tag{24}$$

where the notation has the following meanings:

$Ta_i(k)$ arrival time of vehicle k at section buffer i.

$Td_i(k)$ departure time of vehicle k at section buffer i. We set

$$Td_i(k) = 0 \text{ if } k \le 0.$$

$Tr_l(p)$ replenishment time of part p at lineside buffer l.

The physical meaning of (24) is intuitively clear. There are two cases when we determine the starting time of the r_i+1-th round local simulation of serial section i. If all stations in section i are empty after r_i rounds of local simulation, i.e., $s_{ij}(e_{ir_i}) = 0, \forall j \in \mathcal{J}_i$, the starting time is determined by the finishing time of r_i rounds of simulation and the arrival time of vehicle $Cm_{i1}(e_{ir}) + 1$ at section buffer i-1. Otherwise, some station in section i is not empty, then the starting time of next round of local simulation is determined by the finishing time of r_i rounds of the simulation, departure time of vehicle $Cm_{iJ_i}(e_{ir}) - N_i$ at section buffer i, ready time of the next component for station E_i from merging section i', and arrival time of needed parts at all lineside buffers of serial section i.

Similarly, the starting time of the r_i+1-th round local simulation for merging section i, $i \in \mathcal{I}_m$, is determined by

$$t_{i,r_i+1} = \max\left\{\begin{array}{l} e_{ir_i}, Td_i(Cm_{iJ_i}(e_{ir_i}) - N_i), \\ \{Tr_l(Cu_l(e_{ir_i}) + P_l(Cm_{ij}(e_{ir_i}) + 1)), \forall j \in \mathcal{J}_i, l \in \mathcal{L}_{ij}\} \end{array}\right\} \tag{25}$$

(P3). The number of conveyor moves in the r_i+1-th round local simulation of serial section i, $i \in \mathcal{I}_s$, is determined by (26). The intuition behind (26) is: the r_i+1-th round local simulation of serial section i has to stop once information of input or output buffers, or the merging section, or any lineside buffer of serial section i is not available.

$$m_{i,r_i+1} = \min\left\{\begin{array}{l} Ca_{i-1}(e_{i-1,r_{i-1}}) - Cm_{i1}(e_{ir}), \\ Cd_i(e_{i+1,r_{i+1}}) + N_i - Cm_{iJ_i}(e_{ir_i}), \\ Ca_{i'}(e_{ir_i}) - Cm_{iE_i}(e_{ir_i}), \\ \forall j \in \mathcal{J}_i, l \in \mathcal{L}_{ij}: \\ \max\{K : Cr_l(e_{ir_i}) \ge Cu_l(e_{ir_i}) + \sum_{k=1}^{K} P_l(Cm_{ij}(e_{ir_i}) + k)\} \end{array}\right\} \tag{26}$$

Similarly, the number of conveyor moves in the r_i +1-th round local simulation of merging section i, $i \in \mathcal{I}_m$, is determined by

$$m_{i,r_i+1} = \min\left\{\begin{array}{l} Cd_i(e_{ir_i}) + N_i - Cm_{iJ_i}(e_{ir_i}), \forall j \in \mathcal{J}_i, l \in \mathcal{L}_{ij}: \\ \max\{K : Cr_l(e_{ir_i}) \ge Cu_l(e_{ir_i}) + \sum_{k=1}^{K} P_l(Cm_{ij}(e_{ir_i}) + k)\} \end{array}\right\} \tag{27}$$

(P1) - (P3) outline the boundary conditions for the dividing mechanism on top-level simulation, which plays as the guidance of the aggregated event-scheduling method. With the arrival and the departure time of vehicles at section buffers, and the replenishment time of parts at lineside buffers, we could obtain the time to start next round local simulation in (24)(25) and the number of conveyor moves in each round local simulation in (26)(27), which play the similar role as null messages used in [42][43][39]. This dividing mechanism improves the simulation efficiency significantly due to the following reason: with the condition in **(P1)** and the time given by **(P2)**, we can run a round of local simulation for a section with smaller scales of system states and event list, without communication with other sections, for a given number of conveyor moves pre-determined by **(P3)**.

3) Validations for the Boundary Conditions

This subsection validates the correctness of the boundary conditions of the dividing mechanism on top-level with the following property. The *correctness* here means that there is not deadlock according to the boundary conditions.

Property 1: There is no deadlock according to the boundary conditions of the top-level dividing mechanism. In other words, at any time there exists section i, $i \in \mathcal{I}$, satisfying conditions in **(P1)** to start another round of local simulation.

Proof: No deadlock would happen for the GA line without MH since there is no cycle in the line. Even though deadlock may happen when simulating the entire system with GA line and MH, it is avoidable due to the following reasons. The boundary conditions (20) - (27) in the top level dividing mechanism determine the time to start the next round local simulation and the number of conveyor moves in each round. These time and numbers of all sections are distributed in the top level simulation, similar to the null messages used in [42][43][39], so that the deadlock is avoidable here with the same reason as null message methods in [42][43][39].

The above property provides theoretical validations to the boundary conditions in the top-level dividing mechanism. Numerical validations will be provided in Section V.

4) Computation Complexity Analysis on Top-level

This section analyzes the computation complexity analysis of the top-level dividing mechanism with boundary conditions. The *computation complexity* here indicates the time complexity of the simulation method, i.e., the number of steps the method takes to simulate a given system as a function of the size of the system.

First, we estimate the computation budgets of the traditional event-scheduling method, if it is directly used to solve the problem. By *computation budget*, we mean the CPU time needed to simulate a given system as a function of the system size. For station j in section i, there are six types of events: starts or finishes an operation, fails or repairs, loads or unloads of vehicles. Let J denote the total number of stations in the production line, we have

$$J = \sum_{i \in \mathcal{I}} J_i \,. \tag{28}$$

The average length of the event list will be about $3J$. In traditional event scheduling method, the major computation task is to sort over the event list, which is needed every time an event happens. In the GA line with MH system, the production rate is so fast that conveyors will move in every minute on average. With each conveyor move, there will be four events occurring at each station. Therefore, the frequency of events in the system is very high, about $4J$ events every minute on average. Taking a practical system as an example with total simulation time as one week (about 10^4 minutes), we have $J = 144$, which implies that it is needed to do about $10^4 * 4J = 5*10^6$ sorting operations over an event list with length about $3J = 432$ on average. A numerical test for these sorting operations is done on a PC with 2.80GHz CPU and 2.00GB RAM. The result shows that the above sorting operations will need about 32 hours, which obviously cannot meet the practical requirement.

The computation budget of sorting operation over an event list with length n is proportional to ($n\log_2 n$). With the top-level dividing mechanism with boundary conditions, the aggregated event-scheduling method divides the whole event list into I local event lists, so that we only need to do sorting operations over aggregated event list with length n/I. Note that the total number of sorting operations is not changing. Thus, the computation budget of aggregated-event scheduling method is proportional to ($n/I \log_2(n/I)$), which is about $1/I$ of the traditional one. If we use an alternative simulation operation by maintaining a sorted event list and inserting new generated future events in appropriate places, the computation budget is proportional to n for inserting operation over an event list with length n. Based on the same argument as above, the computation budget of the inserting operation with the top-level dividing mechanism and boundary conditions is proportional to (n/I), which is also $1/I$ of the traditional one. Therefore, as long as we use the same operation (either inserting or sorting) in both the new simulation method and the traditional one, we could save $1/I$ computation budget with the top-level dividing mechanism. This analysis demonstrates the efficiency of the new method. Even though it is a limitation that the improvement of the new method depends on the number of sections, this result is still promising when dealing with large scale practical systems typically with plenty of sections.

C. Bottom-Level: Timing-focused Simulation Strategy

With the dividing mechanism with boundary conditions on top-level, simulations for sections are divided into rounds of local simulations, and they are carried out here on the bottom-level. To further improve the efficiency, local event lists are further reduced on the bottom-level by introducing a timing-focused simulation strategy, which focuses on the time points of key events and derives relations with other events.

1) Timing-Focused Local Simulation for Sections

For each serial section i, $i \in \mathcal{I}_s$, conveyor moves are the key events to determine the dynamic of the section. The timing-focused local simulation strategy focuses on the time of conveyor moves and derives time of other events iteratively. We use $Cc_i(t)$ to denote the accumulated number of conveyor moves in section i at time t. The time for the next conveyor movement is determined as follows. If all stations within the section is empty, the conveyor will move as soon as there are vehicles in the input buffer of section i, i.e., section buffer i-1; if some stations are not empty, the time for the next movement of the conveyor is determined by the following three aspects: 1), the time when all stations in section i finish their current operations; 2), the time when the output buffer of section i, i.e., section buffer i, gets ready to store the next vehicle; 3), the section buffer between merging section i' and serial section i, i.e., section buffer i', has content in it. To summarize, we have:

$$Tc_i(Cc_i(t)+1) = \begin{cases} Ta_{i-1}(Cm_{i1}(t)+1), \text{if } s_{ij}(t)=0, \forall j \in \mathcal{J}_i; \\ \max\{Tf_{ij}(Cm_{ij}(t), \forall j \in \mathcal{J}_i, Td_i(Cm_{i,J_i}(t)-N_i), , \\ \quad Ta_{i'}(Cm_{i,E_i}(t)+1)\}, \text{ otherwise.} \end{cases} \quad (29)$$

where the notation has the following meanings:

$Tc_i(m)$ time of the m-th conveyor-move in section i.

$Tf_{ij}(k)$ finish time for processing vehicle k at station j in section i.

When the conveyor of section i moves, we firstly update the clock to the time given by (29), i.e.,

$$t' = Tc_i(Cc_i(t)+1), \quad (30)$$

where t denotes the original clock before the conveyor moves and t' denotes the updated one. Then, we update time records as follows:

$$Ta_i(Ca_i(t)+s_{iJ_i}(t)) = t' \quad (31)$$

$$Td_{i-1}(Cd_{i-1}(t)+1) = t', \text{ if } b_{i-1}(t) \geq 1 \quad (32)$$

$$Td_{i'}(Cd_{i'}(t)+1) = t', \text{ if } s_{iE_i}(t') = 1. \quad (33)$$

$$Tu_l(p) = t', \forall p = Cu_l(t)+1, \cdots, Cu_l(t') \quad (34)$$

$$Ts_{ij}(k) = \max\{t', Tr_l(Cu_l(t')), \forall l \in \mathcal{L}_{ij}\} \quad (35)$$

$$Tf_{ij}(k) = Ts_{ij}(k) + Tp_{ij}(k), \quad (36)$$

where, $k = Cm_{ij}(t')$ shown later in (44), $\forall j \in \mathcal{J}_i$

where the notation has the following meanings:

$Tu_l(p)$ the time of part p used at lineside buffer l.

$Ts_{ij}(k)$ starting time for processing vehicle k at station j in section i.

$Tp_{ij}(k)$ processing duration of vehicle k at station j in section i.

(35) indicates that the operation for vehicle k can only start when all necessary parts are available at lineside buffers. Note that the processing duration in (36) contains the constant cycle time τ_{ij} for normal assembly operation and the repairing time if failure happens during the operation. The randomness is captured with the exponential reliability model in (9) and (10).

Secondly, the states of stations in section i will be updated as follows.

$$s_{i,j}(t') = s_{i,j-1}(t), \quad \forall j \in \mathcal{J}_i, j \neq 1, \quad (37)$$

$$s_{i,1}(t') = \begin{cases} 1, & \text{if } b_{i-1}(t) \geq 1 \\ 0, & \text{otherwise} \end{cases}, \quad (38)$$

$$s_{i,j}(\tau) = s_{i,j}(t), \forall t < \tau < t', \ \forall j \in \mathcal{J}_i. \quad (39)$$

Thirdly, we update the following counters of related section buffers, stations and lineside buffers.

$$Cd_{i-1}(t') = \begin{cases} Cd_{i-1}(t)+1, & \text{if } b_{i-1}(t) \geq 1 \\ Cd_{i-1}(t), & \text{otherwise} \end{cases} \quad (40)$$

$$Cd_{i-1}(\tau) = Cd_{i-1}(t),\ \forall t < \tau < t'$$

$$Ca_i(t') = Ca_i(t) + s_{iJ_i}(t)$$
$$Ca_i(\tau) = Ca_i(t),\ \forall t < \tau < t' \quad (41)$$

$$Cd_{i'}(t') = Cd_{i'}(t) + s_{iE_i}(t')$$
$$Cd_{i'}(\tau) = Cd_{i'}(t),\ \forall t < \tau < t' \quad (42)$$

$$Cc_i(t') = Cc_i(t) + 1$$
$$Cc_i(\tau) = Cc_i(t),\ \forall t < \tau < t' \quad (43)$$

$$Cm_{ij}(t') = Cm_{ij}(t) + s_{ij}(t'),\ \ \forall j \in \mathcal{J}_i$$
$$Cm_{ij}(\tau) = Cm_{ij}(t),\ \forall t < \tau < t',\ j \in \mathcal{J}_i \quad (44)$$

$$\forall j \in \mathcal{J}_i,\ l \in \mathcal{L}_{ij}:$$
$$Cu_l(t') = \begin{cases} Cu_l(t) + P_l(Cm_{ij}(t')), & \text{if } s_{ij}(t') = 1 \\ Cu_l(t), & \text{otherwise} \end{cases} \quad (45)$$
$$Cu_l(\tau) = Cu_l(t),\ \forall t < \tau < t'$$

Then with these updated counters, we can obtain the state of serial section i and inventory level of section buffers i, i-1, and i' according to (11) and (12).

The timing-focused simulation strategy for merging section i', $i' \in \mathcal{I}_m$, is similar to the case for serial sections. The only difference is the time for the next conveyor-move after time t is determined by (46) since we assume the input buffers of merging sections are never empty and there is no more merges in merge section.

$$Tc_{i'}(Cc_{i'}(t)+1) = \max\{Td_{i'}(Cm_{i'J_{i'}}(t) - N_{i'}), Tf_{i'j}(Cm_{i'j}(t)), \forall j \in \mathcal{J}_{i'}\} \quad (46)$$

2) Timing-Focused Local Simulation for MH System

The timing-focused local simulation for the MH system can be carried out similarly. Here we focus on the starting time of each trip, and derive time relations of other events iteratively. Supposing that at time t, driver d is idle at central docking area, $y_d(t) = 0$, $d \in \mathcal{D}$, having finished $Cs_d(t)$ trips. With a given dispatching policy (13), if the driver is sent out to supply buffer l at time t, i.e., $a_d(t) = l$, then the states of the system will be updated as follows:

$$To_d(Cs_d(t)+1) = t \quad (47)$$

$$Te_d(Cs_d(t)+1) = t + T_l \quad (48)$$

$$Cs_d(t+T_l) = Cs_d(t) + 1$$
$$Cs_d(\tau) = Cs_d(t),\ \forall t < \tau < t + T_l \quad (49)$$

$$Cr_l(t + \frac{1}{2}T_l) = Cr_l(t) + Q_l$$
$$Cr_l(\tau) = Cr_l(t),\ \forall t < \tau < t + \frac{1}{2}T_l \quad (50)$$

$$Tr_l(p) = t + \frac{1}{2}T_l,\ \forall p = Cr_l(t)+1, \cdots, Cr_l(t) + Q_l \quad (51)$$

$$Lb_l(t) = Cr_l(t) - Cu_l(t). \quad (52)$$

where the notation has the following meanings:

T_l round travel time from central docking area to lineside buffer l and then back to central docking area.

Q_l package size for lineside buffer l, i.e., the number of parts supplied to lineside buffer l in each delivery.

The dispatching policy currently used in practical systems is determined by the actual lineside buffer inventories. It is a threshold policy called "Reorder Point Policy". For lineside buffer l, we could obtain the estimated remaining lifetimes, $ERL_l(t)$, i.e., the length of time that lineside buffer l can maintain without replenishment before it goes empty,

$$ERL_l(t) = \tau_{ij} \cdot \max\left\{K : Lb_l(t) \geq \sum_{k=1}^{K} P_l(Cm_{ij}(t)+k)\right\}, \quad (53)$$

where we suppose lineside buffer l belong to station j of section i. The "reorder point policy" works as follows. If $ERL_l(t)$ is no more than a predetermined threshold RP_l, (reorder point), the driver responsible to lineside buffer l needs to start a trip supplying parts to it. If there are multiple lineside buffers need supply, the driver will supply the buffer with smallest estimated remaining life first. In other words, the "reorder point" dispatching policy is denoted as:

$$\pi_d^{RP}(t, Lb_l(t), \forall l \in \mathcal{L}(d)) = l'$$
$$\text{s.t., } ERL_{l'}(t) = \min\{ERL_l(t) : \forall l \in \mathcal{L}(d),\ ERL_l(t) \leq RP_l\} \quad (54)$$

where RP_l denotes the reorder point of lineside buffer l.

3) Validations for Timing-Focused Simulation Strategy

It is guaranteed by the following properties that with this timing-focused simulation strategy, the reduced event list is sufficient to regenerate the simulation trajectory.

Property 2: The trajectory generated in the local simulation of section i, $i \in \mathcal{I}$, is correct.

Proof: With Property 1, all the arrival and departure time series at the input buffer, the output buffer, merging buffer, and lineside buffers of section i are available during the (r+1)th round local simulation. Therefore, the trajectory generated by the local simulation of section i is correct.

Property 3: With the conveyor moving time $Tc_i(m)$, m = 1, 2, …, M_i, $i \in \mathcal{I}$, and trip starting time $To_d(u)$, u = 1, 2, …, U_d, $d \in \mathcal{D}$, (where M_i denotes the total number of conveyor moves in section i and U_d denotes the total number of trips of driver d, within given simulation time), we can regenerate the entire trajectory as (18) defines.

Proof: With properties 1 and 2, the entire trajectory can be regenerated based on the time relations in formulas (30)-(51).

4) Computation Complexity Analysis on Bottom-level

The traditional event scheduling scheme uses event list to generate trajectories [10][12]. Therefore it may suffer great computation complexity caused by the large event lists if it is directly used to do local simulations for the system. For the system concerned in this paper, the traditional event list contains:

(a) One type of event for section i, $\forall i \in \mathcal{I}$: the conveyor move;

(b) Two types of events for section buffer i: the arrival and departure of vehicles;

(c) Six types of events for station j of section i, $\forall i \in \mathcal{I}$, $j \in \mathcal{J}_i$: the start and finish of assembly processes, the load

and unload of vehicles, and the breakdown and repair of the station;

(d) Two types of events for lineside buffer l, $\forall l \in \mathcal{L}$: the arrival and departure of parts;

(e) Three types of events for driver d, $\forall d \in \mathcal{D}$: the start of a supplying trip, the arrival at destination lineside buffers and the end of the trip.

Therefore, the total number of events in traditional sense will rise to:

$$3I - 2 + 6J + 2|\mathcal{L}| + 3|\mathcal{D}|, \tag{55}$$

where I and J denote the total number of sections and stations, respectively. A typical system in practice may consist of about 10 sections, more than 150 stations, more than 300 lineside buffer, and more than 15 drivers. Thus, the total number of events is larger than 1500, which makes the traditional event scheduling scheme not effective enough to mimic the system in acceptable computation time.

Instead, with the timing-focused simulation strategies introduced in bottom-level, only the *key events* are recorded in a *reduced event list* and redundant events are cut off. The time of key events determine the dynamic of the entire system, while the time of redundant events can be deduced from the time of key events based on the timing relations derived above. The reduced event list only consists of key events, which involve the following two types of events:

(a) The move of the conveyor in section i, $\forall i \in \mathcal{I}$;

(b) The start of supplying trips for driver d, $\forall d \in \mathcal{D}$.

Therefore, the total number of key events is:

$$I + |\mathcal{D}|, \tag{56}$$

which is much smaller than the total number of traditional events as (55) shows. Still taking the typical practical system as an example, the number of key events in the reduced event list is 35, which is 37 times smaller than the number of traditional events.

This timing-focused simulation strategy is efficient since it takes advantage of the specific structures of this problem and only focuses on the time of key event, instead of time of all events. Based on these key timings, other timings related to the production process could be inferred through simple formulas, i.e., the entire trajectory is accurately regenerated. Therefore, this strategy presents an elegant performance and improves the simulation efficiency significantly.

D. Discussion on the Simulation Method

As we have pointed out in the literature review, the general principles for parallel and distributed discrete event simulation has provided us useful insights in designing the top-level simulation; however, non-trivial work has to be done to harvest benefits of parallel and distributed simulation for our entire system. Below are the major differences of our work compared with existing work from three aspects.

(1) The FCFS queuing network model studied in [40] does not provide an efficient way to decompose our system. The reason is the dynamic of sections in the GA line with MH is much more complicated and different from the queuing network studied in literatures. Specifically, the state transition of each station relies on the status of all other stations within one section since they are strongly coupled by the conveyor; whereas individual stations are different since they have different MTBF, MTTR, and various consumption rates for parts. Even worse, if with the queuing network formulation, each station would have one queue for semi-produced vehicles and queues for parts. Since there are more than 100 stations and more than 300 lineside buffers in our system, the size of the queuing network would be huge, which makes the traditional method inefficient.

(2) The appointment protocol with lookahead proposed in [40][42] cannot be directly used for our problem since otherwise we might end up with limited performance improvement due to the following reasons. (a) If we use the lookahead mentioned in [42], the lookahead time might be short since the minimum increment of a *Logical Process* (LP) for processing any event is very small in our problem, (in other words, the cycle time of the assembly is very small, about 1 minute). If we use the scheme proposed in [40], the lookahead mostly would equal to the cycle time due to the serial structure of the general assembly line. Thus, the speedup with such short lookahead is limited as it is pointed out in [40] that: the ability to "reduce synchronization overhead to acceptable levels clearly depends on the ability of provide lookahead". (b) The service time in our system (i.e., the assembly time for semi-produced vehicles on each station) is with high variation (each station has random failures and repair time; all stations in one section are highly coupled with one conveyor). As it is pointed out in [40], "under high variation of service time, very small lookahead values are possible, meaning that lookahead is computed more often, thereby incurring increased overhead".

(3) Although we borrowed the basic idea of asynchronous protocols in distributed simulation, it has to be extended to fit our problem. Instead of buffer level (queue length), we record the arrival and departure time of semi-produced vehicles at section buffers. With the information of the information of vehicle arrival and departure time, we can not only regenerate the buffer level according to (12), but also decompose the entire simulation into local simulation for sections as the top-level dividing mechanism shows, instead of the message channel [39][42] and the lookahead method [40][43]. Instead of the appointment protocol [40], we provide the condition and time to start and stop local simulation of each section (i.e., Logical processes) in equations (20)-(27), based on the information of vehicle arrival and departure time recorded at section buffers. Instead of small event lists in logical processes, we use max-plus algebra to capture the dynamics of the strongly coupled stations in each section in the local simulation.

To summarize entire Section IV, the aggregated event scheduling simulation method introduces a hierarchical architecture into simulations and designs a two-level framework to deal with the challenge of large problem scales. Based on the specific structure of the GA line with MHs, a dividing mechanism with boundary conditions is employed on the top level simulation, so that events are aggregated as rounds

of local simulations and the entire event list is divided into pieces. Additionally, a timing-focuses strategy based on max-plus algebra is applied in rounds of local simulation on bottom-level, so that local event lists are further reduced. The effectiveness and efficiency of this aggregated event-scheduling method are demonstrated theoretically through properties and computation complexity analysis. Numerical experiments will be shown in the next section.

V. Numerical Experiments and Insights

This section provides three experiments to demonstrate the effectiveness, efficiency, and the application capability of the aggregated event-scheduling method. Firstly, the method is compared with traditional event scheduling method and Arena on 32 pre-select test systems; secondly, we validate the new method on a practical system by comparing the simulated results with the real production data; thirdly, we apply the new method to simulate various scenarios in order to produce interesting results and managerial insight for the GA line with MH. All the tests are implemented with programming language C++ and tested on a PC with Pentium D 2.80GHz CPU, 2.00GB RAM, Windows system.

A. Effectiveness and Efficiency of the Simulation Method

For comparison purposes, the aggregated event-scheduling method and the traditional one are both implemented on a five-station assembly line shown in Fig. 4 without material handling.

Fig. 4. Layout of 5-station serial line.

We develop both the traditional event scheduling scheme [10][12] and the aggregated event-scheduling method in C++ program to simulate the system above. Additionally, an Arena simulation is carried out for the same systems.

We use 32 groups of test cases with different parameters, which are listed in Table I. The columns 2-5 show the capacity of buffers B_1, B_2, B_3, B_4; columns 6-10 list the MTBF (Mean Time Between Failures, i.e., the reciprocal of rate λ as (9) defines) of station 1, …, 5. All the five stations have the same MTTR (Mean Time To Repair, i.e., the reciprocal of rate μ as (10) defines) in each test case, which is shown in the last column of Table I. In each and every test case, all stations have the same cycle time, 1 minute.

We compare the simulation results and CPU time of the aggregated event-scheduling method, traditional event scheduling scheme, and Arena software with these 32 test cases. The simulation results for system throughput of these three different simulation approaches are shown in Table II. Column 2, 3, and 4 are the simulated throughput of 32 test cases with these three approaches respectively. Columns 5 and 6 are the relative differences between the aggregated event scheduling method versus Arena and traditional scheme. The last two rows of Table II show the maximum and the minimum of the absolute values of the relative differences comparing the new method with Arena and traditional event scheduling scheme.

TABLE I. Parameters of 32 Groups of Test Cases

#	Input									MTTR
	Section buffer capacity				Station MTBF					
	1st	2nd	3rd	4th	1st	2nd	3rd	4th	5th	
1	2	2	2	2	18	18	18	18	18	2
2	6	6	6	6	18	18	18	18	18	2
3	10	10	10	10	90	90	90	90	90	10
4	30	30	30	30	90	90	90	90	90	10
5	2	2	2	2	18	11.3	8	6	4.66	2
6	6	6	6	6	18	11.3	8	6	4.66	2
7	10	10	10	10	90	56.6	40	30	23.3	10
8	30	30	30	30	90	56.6	40	30	23.3	10
9	2	2	2	2	4.66	6	8	11.3	18	2
10	6	6	6	6	4.66	6	8	11.3	18	2
11	10	10	10	10	23.3	30	40	56.6	90	10
12	30	30	30	30	23.3	30	40	56.6	90	10
13	2	2	2	2	18	11.3	4.66	11.3	18	2
14	6	6	6	6	18	11.3	4.66	11.3	18	2
15	10	10	10	10	90	56.6	23.3	56.6	90	10
16	30	30	30	30	90	56.6	23.3	56.6	90	10
17	2	2	2	2	4.66	11.3	18	11.3	4.66	2
18	6	6	6	6	4.66	11.3	18	11.3	4.66	2
19	10	10	10	10	23.3	56.6	90	56.6	23.3	10
20	30	30	30	30	23.3	56.6	90	56.6	23.3	10
21	2	2	2	2	4.66	18	4.66	18	4.66	2
22	6	6	6	6	4.66	18	4.66	18	4.66	2
23	10	10	10	10	23.3	90	23.3	90	23.3	10
24	30	30	30	30	23.3	90	23.3	90	23.3	10
25	2	2	2	2	18	4.66	18	4.66	18	2
26	6	6	6	6	18	4.66	18	4.66	18	2
27	10	10	10	10	90	23.3	90	23.3	90	10
28	30	30	30	30	90	23.3	90	23.3	90	10
29	2	2	2	2	6	6	38	6	6	2
30	6	6	6	6	6	6	38	6	6	2
31	10	10	10	10	30	30	190	30	30	10
32	30	30	30	30	30	30	190	30	30	10

In Table I, the unit for MTBF and MTTR of stations is minute.

Table II illustrates the effectiveness of the aggregated event scheduling method, since it provides almost the same (less than 1% relative difference) simulation results as the traditional event scheduling scheme and software Arena, with only 0.01% and 0.22% relative differences on average, respectively. For each test case, we do simulation for 20 replications with both the aggregated and the traditional event scheduling methods. The average simulation results with these two methods are shown in columns 3 and 4 of Table II, respectively. We perform a *t*-test of the null hypothesis that the difference between simulated throughput with these two methods are a random sample from a normal distribution with mean 0 and unknown variance, against the alternative that the mean is not 0. All 32 test cases result in failures to reject the null hypothesis at the 1% significance level. The *p*-values are shown in column 7 of Table II. This experiment shows that the aggregated event scheduling method provides the same results as the traditional method statistically.

TABLE II. Effectiveness of Aggregated Event-Scheduling: Comparison of Simulation Results of Three Approaches

Case #	Arena result	Traditional event scheduling C++	Aggregated event scheduling C++	Diff. with Arena (%)	Diff. with tradi. (%)	*p*-value
1	46.21	46.02	46.05	0.34	-0.06	0.49
2	49.78	49.78	49.75	0.07	0.06	0.38
3	45.87	45.81	45.82	0.11	-0.02	0.86
4	49.68	49.78	49.72	-0.08	0.12	0.41
5	35.29	35.27	35.21	0.24	0.17	0.07
6	39.17	39.26	39.19	-0.05	0.18	0.06
7	35.05	34.74	34.75	0.86	-0.02	0.89
8	39.18	39.12	39.05	0.33	0.18	0.27
9	35.27	35.21	35.20	0.20	0.02	0.86
10	39.18	39.22	39.20	-0.05	0.04	0.68
11	34.98	34.64	34.74	0.70	-0.29	0.15
12	39.10	39.00	39.10	0.01	-0.25	0.27
13	37.65	37.51	37.53	0.32	-0.05	0.67
14	40.81	40.83	40.86	-0.12	-0.07	0.48
15	37.34	37.21	37.18	0.42	0.06	0.81
16	40.74	40.73	40.72	0.05	0.02	0.88
17	36.31	36.19	36.16	0.42	0.10	0.44
18	39.85	39.86	39.85	-0.01	0.00	0.98
19	35.91	35.60	35.69	0.62	-0.24	0.20
20	39.79	39.71	39.65	0.35	0.15	0.55
21	34.43	34.37	34.40	0.09	-0.08	0.41
22	38.43	38.38	38.40	0.08	-0.05	0.67
23	34.02	33.87	33.80	0.65	0.22	0.32
24	38.28	38.29	38.27	0.03	0.05	0.78
25	35.85	35.81	35.84	0.04	-0.07	0.47
26	39.28	39.33	39.31	-0.07	0.06	0.49
27	35.55	35.42	35.38	0.48	0.12	0.57
28	39.21	39.20	39.10	0.29	0.27	0.18
29	35.59	35.52	35.54	0.14	-0.05	0.72
30	39.91	39.86	39.86	0.13	0.00	0.98
31	35.19	35.01	35.04	0.43	-0.10	0.64
32	39.86	39.82	39.82	0.10	-0.01	0.95
Ave				0.22	0.01	
Max				0.86	0.29	
Min				0.01	0.00	

In Table II, columns 2, 3, and 4 show the simulated throughput in Job/hour. We perform the *t*-test at the 1% significance level. All these *t*-tests for 32 cases return $h = 0$, i.e., failure to reject the null hypothesis.

TABLE III. EFFICIENCY OF AGGREGATED EVENT-SCHEDULING: COMPARISON OF CPU TIME OF TWO APPROACHES

Approach	Arena	Traditional event scheduling in C++ (a)	Aggregated event scheduling in C++ (b)	Time saving vs. traditional (a)/(b)
Average CPU time for each replication	≈ 42s	8.098s	0.155s	52.24

Table III demonstrates the efficiency of the aggregated event scheduling method. Given the almost identical simulation results, these three approaches need various CPU time. Table III compares the average CPU time over the 32 groups of test cases. The total simulation time is 100,000 minutes, and no animation is included in Arena to make the comparison fair. From Table III, we can clearly see that aggregated event scheduling needs much less CPU time than software Arena and traditional event scheduling need. Arena needs long CPU time due to the difference of platform and the total simulation time (the setting with 100,000 minutes total simulation time is much longer than usual settings in Arena). With the same C++ implementation platform, the aggregated event scheduling method on average saves about 52 times computation budget comparing with the traditional one, which illustrates the tremendous improvement on simulation efficiency of the new method.

Note that the computation budget saved here is consistent with the computation complexity analysis in the previous section. In this example system, we have $I = J = 5$, $|\mathcal{D}| = |\mathcal{L}| = 0$. With the top-level dividing mechanism with boundary conditions, we save 1/5 computations according to the analysis in subsection IV-B-4). Moreover, with the bottom-level timing-focused simulation strategy, we reduce event list from average length 43 down to 5, according to (55) and (56) of the analysis in subsection IV-C-4). Therefore, a rough estimation of the total computation saves is about 1/5*5/43 = 1/43, which is consistent with the numerical testing results shown in Table III.

B. Validation with Production Data from a Practical System

We implement the aggregated event-scheduling simulation method on a real-world GA line with MH systems in automotive industry. The scale of the practical system is with several sections (including trim, chassis, door, final, etc.), more than 150 stations, more than 300 lineside buffers, and dozens of drivers. For confidential considerations, the parameters of this practical system cannot be released in public. The warm-up time, the total simulation time and the replication number are set to 5,000 minutes, 10,000 minutes, and 20, respectively, according to the following statistic tests shown in Table IV and Table V.

TABLE IV. STATISTICAL TESTS FOR DIFFERENT WARM-UP TIME

Measure	5,000 min. Warm-up time	10,000 min. Warm-up time	*p*-value	Measure	5,000 min. Warm-up time	10,000 min. Warm-up time	*p*-value
TP	29.79	29.75	0.27	U_9	0.62	0.62	0.43
U_1	0.53	0.53	0.47	U_{10}	0.80	0.80	0.10
U_2	0.51	0.50	0.03	U_{11}	0.69	0.69	0.13
U_3	0.69	0.69	0.14	U_{12}	0.69	0.68	0.38
U_4	0.61	0.61	0.53	U_{13}	0.58	0.58	0.69
U_5	0.65	0.65	0.26	U_{14}	0.56	0.56	0.53
U_6	0.68	0.67	0.05	U_{15}	0.51	0.51	0.81
U_7	0.63	0.63	0.73	U_{16}	0.43	0.43	0.51
U_8	0.68	0.68	0.93				

All the *t*-tests in Table IV are performed at the 1% significance level, and all of them return $h = 0$, i.e., failure to reject the null hypothesis. Columns 1 and 5 show the "Measure", where *TP* denotes the throughput; U_d is the utilization of driver d, $d = 1, \ldots, 16$. Note that simulation results are modified for the business confidential considerations, but the modification does not add any inaccuracy in this comparison. (The same as Table V does).

TABLE V. STATISTICAL TESTS FOR DIFFERENT REPLICATION NUMBERS

Measure	Replication # = 20	Replication # = 50	*p*-value	Measure	Replication # = 20	Replication # = 50	*p*-value
TP	29.79	29.80	0.69	U_9	0.62	0.62	0.69
U_1	0.53	0.53	0.82	U_{10}	0.80	0.80	0.29
U_2	0.51	0.51	0.61	U_{11}	0.69	0.69	0.92
U_3	0.69	0.69	0.93	U_{12}	0.69	0.69	0.70
U_4	0.61	0.61	0.59	U_{13}	0.58	0.58	0.97

U_5	0.65	0.65	0.58	U_{14}	0.56	0.56	0.39
U_6	0.68	0.68	0.59	U_{15}	0.51	0.51	0.88
U_7	0.63	0.63	0.35	U_{16}	0.43	0.43	0.61
U_8	0.68	0.68	0.54				

All the *t*-tests in Table V are performed at the 1% significance level, and all of them return $h = 0$, i.e., failure to reject the null hypothesis.

Table IV show the *t*-tests for two simulation setups: one is with 5000 minutes warm-up time and 10,000 minutes total simulation time; the other is with 10,000 minutes warm-up time and 15,000 minutes total simulation time. Both setups are with 20 replications. We perform *t*-tests for the throughput (*TP*) and individual utilizations of 16 drivers (U_d, $d = 1, \ldots, 16$) obtained from these two simulation setups. The null hypothesis of each test is that the differences of simulation results between these two setups are random samples from a normal distribution with mean 0, against the alternative that mean is not 0. All tests result in failures to reject the null hypothesis at the 1% significance level, with the *p*-values shown in columns 4 and 8 in Table IV. This result illustrates that the setup with 5000 minutes warm-up time and 10,000 minutes total simulation time is statistically sufficient. Similarly, Table V shows the *t*-tests for another two simulation setups: one is with 20 replications and the other is with 50 replications. Both of them are with 5000 minutes warm-up time and 10,000 minutes total simulation time. The null hypothesis of each test is that the differences of simulation results between these two setups are random samples from a normal distribution with mean 0, against the alternative that mean is not 0. All *t*-tests result in failures to reject the null hypothesis at the 1% significance level, with the *p*-values shown in columns 4 and 8 in Table V. This result demonstrates that the setup with 20 replications is statistically sufficient.

We compare the simulation results with the practical production data which is measured and collected for one month in a factory. The results are shown in Table VI and VII.

TABLE VI. DIFFERENCES OF SIMULATED AND MEASURED UTILIZATIONS

Driver index	Average Difference (%)	Confidence Interval with 90% Confidence Level (%)	
#1	2.42	2.40	2.44
#2	2.90	2.88	2.92
#3	2.45	2.43	2.46
#4	2.21	2.21	2.22
#5	2.40	2.39	2.42
#6	2.55	2.52	2.57
#7	2.36	2.35	2.38
#8	0.02	0.00	0.04
#9	2.44	2.42	2.45
#10	2.46	2.45	2.47
#11	1.91	1.90	1.92
#12	2.44	2.41	2.46
#13	1.27	1.26	1.28
#14	2.51	2.49	2.53
#15	2.55	2.53	2.58
#16	2.54	2.52	2.55
Mean	2.21	2.20	2.23

TABLE VII. DIFFERENCES OF SIMULATED AND MEASURED THROUGHPUT

	Average Difference (%)	Confidence Interval with 90% Confidence Level (%)	
System throughput	2.12	2.11	2.13

Table VI and Table VII demonstrate the effectiveness of the aggregated event scheduling method. Table VI shows that the simulated utilizations of 16 drivers are close to the measured ones (with 2.21% relative difference and [2.20%, 2.23%] confidence interval for 90% confidence level on average). Table VII shows that the simulated throughput is close to the measured one (with 2.12% relative difference and [2.11%, 2.13%] confidence interval for 90% confidence level). This discrepancy is due to some human factors which are simplified in simulation. This accuracy is good considering the fact that there is typically about 5% measure error in data collection.

Additionally, the above experiment also demonstrates the efficiency of the aggregated event-scheduling method. We should notice that the CPU time of our new simulation method is only 1.85 seconds per replication for the practical system with 10,000 minutes (about 7 days) total simulation time with good match for the measured throughput and utilization data. This simulation speed illustrates the efficiency of the new method.

C. *Application of the Aggregated Event Scheduling Method*

This section applies the aggregated event scheduling method to investigate the behavior of the GA line with MH systems. By varying the parameters and the settings of the practical system introduced in the previous subsection, we could answer what-if questions concerned in practice.

1) *Reallocation of lineside buffers*

The simulation result of the practical system introduced in previous subsection shows that the utilization varies a lot among drivers, from 0.428 to 0.796, with mean value 0.613 and variance 0.008 (as column 2 in Table VIII shows). The reason is each drive supplies a given set of lineside buffers and the allocation of buffers is imbalanced. By moving some lineside buffers from two drivers whose utilization is more than 0.6 to another two drivers whose utilization is less than 0.5, we could decrease the variance of driver utilization from 0.008 to 0.002, and increase the system throughput by 0.82% (shown in column 3 of Table VIII). This simulation result suggests managerial insight for industry engineers since it provides a way to improve the system performance without any further investment.

2) *Putting all drivers into one group*

Each driver supplies a given and fixed set of lineside buffers and the replenishment is not exchangeable in current system. This section we simulate the scenario by putting all drivers into one group and letting them supply all the lineside buffers based on availability. The simulation result is shown in column 4 of Table VIII, and it turns out that all drivers have almost identical utilizations in this setting (with variance about $4.49*10^{-7}$), and the system throughput increase by 1.01% comparing with the original setting shown in column 2. This experiment suggests a new managerial concept for the automotive industry.

TABLE VIII. VARIOUS ALLOCATIONS OF LINESIDE BUFFERS

	Original	Reallocated	One group
Throughput	29.80	30.04	30.10
Mean utilization	0.613	0.619	0.620
Variance utilization	0.008	0.002	0.000
Max. utilization	0.796	0.694	0.621
Min. utilization	0.428	0.514	0.618

Row 2 shows the throughput in Job/hour. Rows 3-6 show the mean, variance, maximum and minimum of the utilization of 16 drivers. Note that all results are modified for the business confidential considerations, but the modification does not add any inaccuracy in the comparison. (The same as the following Table IX and Table X do).

3) The necessary number of drivers

This section obtains the minimum number of drivers needed in the MH system by simulating systems with various numbers of drivers. Here we still use the previous setting that all drivers are put into one group. With the minimum throughput required and maximum driver utilization allowed, say 30.0 and 0.72, we could obtain the necessary number of drivers is 14 based on the simulation results in Table IX.

TABLE IX. VARIOUS NUMBERS OF DRIVERS

#Driver	16	15	14	13	12
Throughput	30.10	30.10	30.11	30.11	29.09
Mean utilization	0.620	0.661	0.708	0.763	0.799

Row 3 shows the mean utilization of all the drivers. Since all drivers are put in one group, the utilization of all drivers is identical with variance less than 10^{-6}.

4) Parameter sensitivity

In this section we analyze the sensitivity of the reorder point of the dispatching policy defined in (54). Simulation result with various reorder points in Table X shows that larger reorder point implies higher throughput and driver utilization on average. In this experiment we use the original allocation of lineside buffers as in subsection V-B.

TABLE X. VARIOUS REORDER POINT

Reorder point (minute)	15	30	60
Throughput (job/hour)	28.58	29.80	29.90
Mean utilization	0.588	0.613	0.616

The simulation method developed in this paper is helpful to answer "what-if" questions and produce managerial insight on the behavior of the GA line with MH system, since this method can simulate large scale systems within acceptable computation time. Additionally, the simulation method can also be applied to do simulation-based optimization and analysis for the GA line with MH system, which are shown in our recent work [38][41].

VI. CONCLUSIONS

This paper presents a novel simulation method for a GA line with MH systems to meet the challenges of large problem size and correlated system dynamics. Different from the traditional event scheduling based methods, it combines ideas of max-plus algebra, event scheduling and decoupled simulation. Making use of the partial system decomposability, we introduce a two-level simulation framework. A dividing mechanism with boundary conditions is employed on top-level to divide the global event list into small sizes. A timing-focused strategy based on max-plus algebra is applied on bottom-level local simulation to further reduce local event lists. With this new method it is possible to mimic real production systems fast and accurately within a reasonable computational time frame. Numerical testing results of a practical production line show that the new method is efficient and effective. Additionally, although this simulation method is developed for the GA line with MH systems, it should be pointed out that this method is also applicable to simulate other manufacturing systems (such as disassembly lines), and systems with fork/join structures (such as parallel processing systems and distributed replicated database systems).

This simulation method can be applied to investigate various scenarios and analyze parameter sensitivity. More importantly, engineers and plant managers could make real time decisions via this fast and accurate model. Furthermore, simulation based optimization is being conducted systemically and will be reported in our future work.

ACKNOWLEDGMENT

The authors would like to thank Prof. Yu-Chi Ho, Prof. Xi-Ren Cao, Prof. Leyuan Shi, Dr. Qing-Shan Jia and Mr. Tao Sun for their valuable comments and suggestions. The authors also wish to thank the anonymous reviewers for their insightful comments.

REFERENCES

[1] J. Li and S. M. Meerkov, *Production systems engineering*, Springer, New York, 2008.

[2] E. J. Williams and H. Celik, "Analysis of convey or systems within automotive final assembly," In *Proceedings of the 1998 Winter Simulation Conference*, pp. 915-920. D. J. Medeiros, E. F. Watson, J. S. Carson and M. S. Manivannan, eds.

[3] S. K. Bhattacharyya, R. Roy, and M. J. Low, "A computer simulation system for the evaluation of man assignments on car assembly tracks," *Simulation*, vol. 61(2), pp. 124-133, 1993.

[4] Y.-L. Chang, R. S. Sullivan, and J. R. Wtlson, "Using SLAM to design the material handling system of a flexible manufacturing system," *International Journal of Production Research*, vol. 24(1), pp. 15-26, 1986.

[5] A.-V. Ardavan, and L. Gilbert, "Loop based facility planning and material handling," *European Journal of Operational Research*, vol. 164, pp. 1-11, 2005.

[6] S. B. Gershwin, *Manufacturing systems engineering* / Stanley B. Gershwin. Englewood Cliffs, N.J. : PTR Prentice Hall, 1994.

[7] B. M. Beamon, Performance, reliability, and performability of material handling systems, *International Journal of Production Research*, vol. 36(2), pp. 377- 393, 1998.

[8] C. G. Cassandras and S. Lafortune, *Introduction to discrete event systems*. Boston: Kluwer Academic, 1999.

[9] J. Li, D. E. Blumenfeld, N. Huang, and J. M. Alden, "Throughput analysis of production systems: recent advances and future topics," *International Journal of Production Research*, to appear, 2008.

[10] J. Banks, J. S. Carson II, B. L. Nelson, and D. M. Nicol, *Discrete-event system simulation*. Upper Saddle River, NJ : Prentice Hall, 2005.

[11] J. A. Harding and K. Popplewell, "Simulation: an application of factory design process methodology," *The Journal of the Operational Research Society*, vol. 51(4), pp. 440-448, 2000.

[12] A. M. Law and W. D. Kelton, *Simulation modeling and analysis*. New York: McGraw-Hill, 2000.

[13] J. S. Smith, "Survey on the use of simulation for manufacturing system design and operation," *Journal of Manufacturing Systems*, vol. 22(2), 2003.

[14] G. Weigert, S. Horn and S. Werner, "Optimization of manufacturing processes by distributed simulation," *International Journal of Production Research*, vol. 44(18–19), pp. 3677–3692, 2006.

[15] W. Wang and R. Bell, "A knowledge based multi-level modelling system for the design of flexible machining fadlities," *International Journal of Production Research*, vol. 30(1), pp. 13-34, 1992.

[16] A. Grosfeld-Nir, M. Magazine, and A. Vanberkel, "Push and pull strategies for controlling multistage production systems," *International Journal of Production Research*, vol. 38(11), pp. 2361- 2375, 2000.

[17] A. Grosfeld-Nir, M. Magazine, "A simulation study of pull systems with ascending/descending buffers and stochastic processing times," *International Journal of Production Research*, vol. 43(17), pp. 3529-3541, 2005.

[18] A. W. Booth, "Object-Oriented Modeling for Flexible Manufacturing Systems," *The International Journal of Flexible Manufacturing Systems*, vol. 10, pp. 301–314, 1998.

[19] L. Rabelo, M. Helal, A. Jones and H.-S. Min, "Enterprise simulation: a hybrid system approach," *International Journal of Computer Integrated Manufacturing*, vol. 18(6), pp. 498 – 508, 2005.

[20] S. Kumar, D. A. Nottestad, "Capacity design: an application using discrete-event simulation and designed experiments," *IIE Transactions*, vol. 38, pp. 729–736, 2006.

[21] R. A. Lindau, T. Kanflo and K. R. Lumsden, "Impact of Real-time Information for Scheduling a Car-body Shop - A Simulation Study," *International Journal of Operations & Production Management*, vol. 14(3), pp. 114-125, 1994.

[22] F. T.S. Chan, "Using simulation to predict system performance: a case study of an electro-phoretic deposition plant," *Integrated Manufacturing Systems*, vol. 6(5), pp. 27-38, 1995.

[23] A. Jayaraman, R. Narayanaswamy, A. K. Gunal, "A sortation system model," In *Proceedings of the 1997 Winter Simulation Conference*, ed. S. Andrado'ttir, K. J. Healy, D. H. Withers, and B. L. Nelson.

[24] G. M. Buxey and D. Sadjadi, "Simulation studies of conveyor-paced assembly lines with buffer capacity," *International Journal of Production Research*, vol.14(5), 607-624, 1976.

[25] A. Prakash, M. Chen, "A simulation study of flexible manufacturing systems," *Computers ind. Engng*, vol. 28(I), pp. 191-199, 1995.

[26] S. N. Kumar, R. Sridharan, "Simulation modeling and analysis of tool sharing and part scheduling decisions in single-stage multimachine flexible manufacturing systems," *Robotics and Computer-Integrated Manufacturing*, vol 23, pp. 361–370, 2007.

[27] K. S. El-Kilany, P. Yong, M. A. El Baradie, Generic tool for modelling and simulation of semiconductor intrabay material handling system. *Journal of Materials Processing Technology*, vol. 155-156, pp. 1927-1934, 2004.

[28] S. H. Kong, "Two-step simulation method for automatic material handling system of semiconductor fab," *Robotics and Computer-Integrated Manufacturing*, vol. 23 pp. 409-420, 2007.

[29] Y. Han, D. Park, S. Chae, C. Lee, "Full fabrication simulation of 300mm wafer focused on AMHS (Automated Material Handling Systems)," *Lecture Notes In Computer Science*, vol. 3398, pp. 514-520, 2005.

[30] X.-R. Cao and Y.-C. Ho, "Models of discrete event dynamic systems", *IEEE Control Systems Magazine*, vol. 10(4), pp. 69-76, 1990.

[31] D.-Z. Zheng and Q. Zhao, *Discrete event dynamic systems*, (in Chinese). Tsinghua University Press, 2001.

[32] G. Cohen, D. Dubois, J. P. Quadrat, and M. Viot, "A linear-system-theoretic view of discrete-event processes and its use for performance evaluation in manufacturing," *IEEE Transactions on Automatic Control*, vol. 30(3), pp. 210-220, 1985.

[33] S. R. Das, "Adaptive protocols for parallel discrete event simulation," *The Journal of the Operational Research Society*, vol. 51(4), pp. 385-394, 2000.

[34] R. M. Fujimoto, "Parallel and distributed simulation systems," In *Proceedings of the 2001 Winter Simulation Conference*, pp. 147-157, B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W Rohrer, eds.

[35] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: numerical methods*. Athena Scientific,1997.

[36] R.R. Inman, "Empirical evaluation of exponential and independence assumptions in queuing models of manufacturing systems," *Production and Operations Management*, vol. 8(4), 409-432, 1999.

[37] Y. Zhao, C. Yan, Q. Zhao, N. Huang, J. Li and X. Guan, "Efficient Simulation for Serial Production lines based on Aggregated Event-Scheduling," In: *Proceedings of the 4th IEEE Conference on Automation Science and Engineering*, pp. 406-411, 2008.

[38] Y. Zhao, Q. Zhao, Q.-S. Jia, X. Guan, X.-R. Cao, "Event-Based Optimization for Dispatching Policies in Material Handling Systems of General Assembly Lines", In: *Proceeding of the 47th IEEE Conference on Decision and Control*, pp. 2173-2178, 2008.

[39] A. Ferscha and S.K. Tripathi, "Parallel and distributed simulation of discrete event systems", *Handbook of Parallel and Distributed Computing*, McGraw Hill, 1995.

[40] D.M. Nicol, "Parallel discrete-event simulation of FCFS stochastic queueing networks", In: *Proceedings of the ACM/SIGPLAN conference on Parallel programming: experience with applications, languages and systems*, pp. 124-137, 1988.

[41] C.-B. Yan, Q. Zhao, N. Huang, G. Xiao, and J. Li, "Line-side Buffer Assignment in General Assembly Line Systems with Material Handling", the 13th *IFAC Symposium on Information Control Problems in Manufacturing (INCOM'09)*, accept, 2009.

[42] V.-Y. Vee, W.-J. Hsu, "Parallel Discrete Event Simulation: A Survey", Centre for Advanced Information Systems. Report from Nanyang Technical University, 1998.

[43] D.M. Nicol, "Principles of Conservative Parallel Simulation", In: *Proceedings of the 1996 Winter Simulation Conference*, pp. 128-135, 1996.

[44] A. Park, R.M. Fujimoto and K.S. Perumalla, "Conservative Synchronization of Large-Scale Network Simulations", *Proceedings of the 18th Workshop on Parallel and Distributed Simulation*, pp. 153-161, 2004.

[45] A. Falcon, P. Faraboschi and D. Ortega, "An Adaptive Synchronization Technique for Parallel Simulation of Networked Clusters", *IEEE International Symposium on Performance Analysis of Systems and Software*, pp. 22-31, 2008.

[46] M.J. Chung and J. Xu, "And Overhead Reducing Technique for Time Warp", *Journal of Parallel and Distributed Computing*, vol. 65, pp. 65-73, 2005.

[47] D.W. Bauer Jr. and E.H. Page, "Optimistic Paralle Discrete Event Simulation of the Event-based Transmission Line Matrix Method", *Proceedings of the 2007 Winter Simulation Conference*, pp. 676-684, 2007.

[48] R. Nelson and A.N. Tantawi, "Approximate Analysis of Fork/Join Synchronization in Parallel Queues", *IEEE Transactions on Computers*, vol. 37(6), pp. 739-743, 1988.

[49] H.W. Maalouf and M.K. Gurcan, "Minimisation of the Update Response Time in a Distributed Database System", *Performance Evaluation*, vol. 50, pp. 245-266, 2002.