Continuous Simulation

Keerthi Krishna PARVATHANENI

October 7, 2024

1 Abstract

Simulation has been used to analyze, optimize the performance of complex manufacturing systems. Since simulations are more flexible than the real world system, it can be a very sophisticated choice for generating data that can be used in data hungry methodologies like machine learning (ML). Discrete event simulation (DES) is a standard computational method to evaluate the state space of a system, which requires a complete description of all the events (low and high frequency operations / events) on the shop floor that were executed in series. This results in an algorithm that is inherently slow. On the other hand, approximating a discrete event system to a continuous model will allow us to jump from state space to the next without simulating all the operations. In this work we developed a robust continuous simulation model that can handle transfer lines and trees. This study also contains a rigorous comparison between the developed model against the results from discrete events simulation and analytical model. During this we studied the effect of convergence criterion on the steady state prediction of the model.

Keywords: Manufacturing systems, Simulation, Optimization, Discrete event simulation, Continuous simulation.

Version FR

La simulation a été utilisée pour analyser et optimiser les performances de systèmes de fabrication complexes. Comme les simulations sont plus flexibles que le système réel, elles peuvent constituer un choix très sophistiqué pour générer des données qui peuvent être utilisées dans des méthodologies avides de données telles que l'apprentissage automatique (ML). La simulation d'événements discrets (DES) est une méthode de calcul standard pour évaluer l'espace d'état d'un système, qui nécessite une description complète de tous les événements (opérations/événements à basse et haute fréquence) de l'atelier qui ont été exécutés en série. Il en résulte un algorithme intrinsèquement lent. D'autre part, l'approximation d'un système à événements discrets par un modèle continu nous permet de passer d'un espace d'état à l'autre sans simuler toutes les opérations. Dans ce travail, nous avons développé un modèle de simulation continu robuste qui peut gérer les lignes de transfert et les arbres. Cette étude contient également une comparaison rigoureuse entre le modèle développé et les résultats de la simulation d'événements discrets et du modèle analytique. Au cours de cette étude, nous avons étudié l'effet du critère de convergence sur la prédiction de l'état stable du modèle.

Mots-clés : Systèmes de fabrication, Simulation, Optimisation, Simulation d'événements discrets, Simulation continue.

2 Introduction

Tasks such as predicting the future state of a manufacturing line, analyzing the effect of the line parameters and optimization of the buffer need a model that can provide the key indicators such as throughput of a line and average buffer levels. Analytical models such as decomposition were popular and the fastest way to analyze the effect of the line parameters on these key indicators. However, they come with a lot of assumptions and this will lead to simplification of the line, this renders an inaccurate prediction in case of complex lines. This gap has been filled by using simulation models, unlike analytical model simulations can replicate the exact behavior until to the scale of user need. Discrete event simulation is one such method to simulate a system. Depending on the scale and parameters in the simulation, DES can be painfully slow. This makes it a difficult choice during optimization or any other application where we need to perform many simulations. The slowness of a DES simulation is inherited from the level of abstraction at which the events/operation are defined in simulation. For example, consider a transfer line where machines are separated by a finite sized buffer. One can extract the events to be simulated into two sets.

- High frequency operation (operation on product, exit and entry of parts at machines).
- Low frequency operation (machine failure and repairs, blocking or starvation).

A DES model is designed to simulate both these event groups, thus making the algorithm slower. Other than simulating the part as a discrete element, one can approximate the flow of the material to be continuous. This will help us not to visit the state where a part enters a machine and leaves a machine. This is the idea of a continuous simulation (CS). We approximate a discrete part system as a continuous material system by only simulating the low frequency high time period events such as machine failure, repair, blocking and starvation. This idea can be found in literature, the most recent advancements for CS are presented in the work of Salvatore Scrivano et al [1]. They presented a generalized CS model that can handle from transfer lines to loops, they reported CS in general is about 15 times faster than DES. However, the proposed model was only for single part-time and is not flexible enough to apply a user control policy. Also, it lacks some robust testing methodology to compare the results acquired from CS with respect to DES. Moreover, in an algorithm during a starvation or blocking the processing rates of the upstream and downstream machines are modified according to a parameter called characteristics' length that is evaluated as the longest chain length in the given graph. This results in approximations on the processing rates of the machines. In our article we replace this method and evaluate the processing rates of the machines based on the upstream and downstream of the machines. This will help us find the exact processing rates for the machines.

The conclusions and observations drawn from this literature review, helped us to formulate a problem statement that will help us to develop a generalized continuous simulation model.

- Build a generalized model to handle lines, trees and loops.
- Evaluate the exact state of the simulation using the information from the graph.
- Handle the complex flow systems such as multiple part types.

In this report is a first approach from Dillygence to address the statements that were mentioned above. We will present a CS algorithm to solve the transfer lines. The later part of the document you will see the introduction to transfer lines and nomenclature, followed by a continuous simulation algorithm and finally ends with the results and conclusions.

2.1 Nomenclature

A transfer line is a production system where the machines in series may or not be separated by buffers, to form a unidirectional graph. An example for a production graph can be found in figure 1. The machines were indicated by a square and a buffer with a circle and their corresponding parameters are listed in table 1 and table 2.

$$\begin{tikzcd} \boxed{M_1} \arrow[r] & B_1 \arrow[r] & \boxed{M_2} \arrow[r] & B_2 \arrow[r] & \boxed{M_3} \end{tikzcd}$$

Figure 1: Three machine transfer line

Processing rate (μ_m): The amout of material can be operated on the machine in a unit time.

Meantime to fail (MTTF): As the name defines, it's a statistical representation of a distribution that can be used to generate the random variable that gives the operation time before its next failure.

Meantime to repair (MTTR): Similar to MTTF, it's a statistical representation of a distribution that gives a number to indicate the time required to repair a machine.

Buffer capacity ($N\{b\}$): It's a limit to number of parts can be stored in buffer at a given time.

□	Machine or module to carry operation.
$\mu_m = 1/cycletime$	Processing rate
$MTTF$	Mean Time To Fail
$MTTR$	Mean Time To Repair

Table 1: Module parameters

○	Buffer to store parts in between operations.
$N\{b\}$	Maximum buffer capacity
$L\{b\}$	Minimum buffer capacity

Table 2: Buffer Parameters

3 Algorithm

To construct an algorithm we must set the required conditions and assumptions. The material flow is continuous. A machine can fail in a regular interval of time and can be repaired immediately without waiting for an operator. The mean time to failure and repair rates of the machines are known before starting the simulation. The buffers can be finite or infinite and can never fail. The simulation starts at time zero and progresses forward with a small-time increment Δt. These small-time increments are called events, and they are the low frequency activities such as failure, repair and blocking and starvation. The time at the next event is $t(k+1)$, from the present time t_k can be measured by equation 1. Then events can be categorized into two types.

1. Module events.
2. Buffer events.

$$t(k+1) = t(k) + \Delta t \tag{1}$$

3.1 Module events

For each machine $M\{m\}$, the remaining machine time $RM\{m\}$ is time before the occurrence of the next event related to a change of the machine state. At time t_k, the machine $M\{m\}$ is in state one referees that it is operating and zero if it is undergoing repair. From the definition of $RM\{m\}$, it is necessary to know the time period to the occurrence of a transition from one to zero or vice versa. For a line with multiple machines, the next possible machine event will be minimum of $RM\{m\}$. To measure this we will use a time to failure matrix $FM\{m\}$ of size 2 $\times$ k. Where, each column corresponding to "k" is mode of failure and the first element in the column is time to failure and second element on the corresponding column is time or repair.

Hence, we can decompose $FM\{m\}$ into two parts $FM\{m, ttf\}$ contains the time to next failure and $FM\{m, ttr\}$ that has time to repair. Time to fail (ttf) and time to repair are random variables generated during the simulation from the distribution parameters MTTF and MTTR. When the simulation starts at time $t\{k\} = 0$, the value of each entry in $FM\{m\}$ acquired is from an exponential distribution.

The failure matrix should be recomputed at the end of every event in the system. Since, ttf is operation dependent, one should know if the machine is slowed down because of blocking or starvation. In such cases, we can have ideal processing rate as μ_m^{ideal} and update processing rate $\mu_m^{updated}$ because of blocking or starvation. Now once can update the failure matrix $FM\{m\}$ using equation 2 and 3.

$$FM\{m, ttf\} = (FM\{m, ttf\} \times \frac{\mu_m^{updated}}{\mu_m^{ideal}}) - \Delta t \tag{2}$$

$$FM\{m, ttr\} = \begin{cases} FM\{m, ttr\} - \Delta t, & \text{if } FM\{m, ttf\} = 0 \\ FM\{m, ttr\}, & \text{else} \end{cases} \tag{3}$$

Now remaining machine time can be measured as $RM\{m\} = min(FM\{m\})$.

3.2 Buffer events

The remaining buffer time $RB\{b\}$ of a buffer b is defined as the time to its fulfillment or depletion of $B\{b\}$. To get $RB\{b\}$, the vector of size 1×2 is introduced such that the first entry defines the time to fill buffer and the second entry is time to deplete buffer $B\{b\}$. These values depend on the current buffer level $x^b(t_{k+1}) = x^b(t) + (\mu_u - \mu_d) * \Delta t$ and on the effective processing rates μ_m of the immediately upstream and downstream machines $M\{b-1\}$, $M\{b+1\}$ respectively. For example consider a two machine transfer line as shown in figure 2. Rate of material accumulating in buffer can be measure by $flux = \mu_u - \mu_d$. Time to

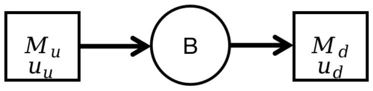


Figure 2: Material accumulating in buffer

fulfillment $RB\{b, tf\}$ and time to depletion $RB\{b, td\}$ can be measure by equation 4 and 5.

$$RB\{b, tf\} = \frac{N\{b\} - x\{b\}}{\mu_u - \mu_d} \tag{4}$$

$$RB\{b, td\} = \frac{L\{b\} - x\{b\}}{\mu_u - \mu_d} \tag{5}$$

For a long transfer lines more than once buffer can be filled or emptied this results in accelerating the blocking and starvation of modules, this phenomenon can be addressed by propagation.

3.3 Propagation

In figure 3 we can see a long line, the color of the module is related the processing speed of machine or module. The dark green represents a higher processing rate and lighter color indicates a lower processing rate respectively. In case of *blocking* buffer $B3$ gets filled first (represented in dark blue color) and this forces module M_3 to operate at lower processing than its ideal processing rate. Since the flow should accept the continuity of material the update processing rate of module three μ_3^{update} should equal to μ_4^{ideal}. If we assume all the modules will not fail in near future this will force $B2$ to fill, hence blocking M_2. Applying the continuity condition the processing rate for second modules should be updated as $\mu_2^{update} = \mu_3^{updated}$. This phenomenon can be termed as propagation. Similarly, the propagation of starvation is also showed in figure 3 on right side. The red color represents the buffer getting empty.

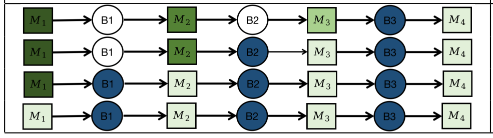
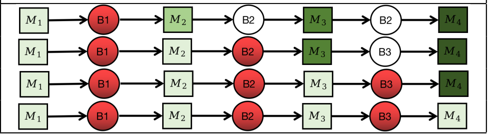
Blocking	Starvation
	

Table 3: Propagation of blocking and starvation in a transfer line.

In case of blocking, after certain time all the buffers will be filled forcing all the upstream modules to operate with the processing rates of the downstream modules. For Starvation the modules down the stream are forced affected due to the lower processing rates of upstream machines. In our algorithm, we address this phenomenon by changing the processing rates of the modules based on the buffer level on upstream and downstream to machine.

Now we have all the pieces to stitch, the algorithm for building a continuous simulation is presented in algorithm 1. The simulation starts at current time $t_k = 0$ the next event in the system can be obtained as $\Delta t = min(RM\{m\}, RB\{b\})$. The next step is to update the state space of the system by this Δt. The update step involves computing the buffer levels, updating the failure matrix and propagation in case of blocking or starvation.

Algorithm 1: Algorithm for continuous simulation

```
Data: Production graph with parameters for modules and buffers.
$run\ time = t_{simulation}$
Result: Average buffer levels and throughput
initialization;
$t_k = 0$ ;
while $t_k < t_{simulation}$ do
    compute events
    $RM\{m\}$ = compute module events ;
    $RB\{m\}$ = compute buffer events ;
      $t_{k+1} = t_k + \min(RM\{m\}, RB\{m\})$ ;
    update state
    compute buffer levels
      $x^b(t_{k+1}) = x^b(t) + (\mu_u - \mu_d) * \Delta t$
    update module state
      $\mu_m$ - Processing rates
      $FM_m$ - Failure matrix
    Propagation to update processing rates.
end
```

4 Results and conclusions

In this section we present the comparisons of throughput and average buffer levels between continuous and in house developed discrete event simulation. In general two models are compared based on their results at steady state. Theoretically, a system is said to be at steady state, if all the state probabilities of the system approach a constant. For a small machine line with finite buffers the total state spaces can be equal to or more than gas molecules in a room. For present computational capabilities it will take forever to visit each of this state more than once, this condition cannot be fulfilled. However, when we run the simulations long enough certain line performance indicator such as throughput and average buffer levels approaches a constant value and this state is now mentioned as steady state.

4.1 Results

We will start by comparing the results for a given simulation time $t_{simulation}$. The line parameters were generated by using the in house case generation method developed by Stan Gershwin. We simulated the same line for the constant simulated time $t_{simulation} = 10days$, but we repeated the simulation for $10, 100, 1000$ times. The relative error in the throughput estimate in for all the three cases did not exceed 0.806% indicating that $t_{simulation} = 10days$ is long enough to achieve a constant throughput or a steady state in terms of throughput. In case of average buffer levels ($\bar{N}$) as presented in figure 4 from the first row, we observe that continuous and discrete event simulation approach to same mean. However, their standard deviations are much higher, which indicated that buffer are not yet approaching to steady state. In the second row we show the buffer average occupancy calculated as $\bar{N}/N$. The buffer occupancy shows both the buffer varies from being empty to fulfill. The buffer distribution approach a normal distribution with increase in number of repetitions, this is due to rule of large numbers as we are simulating more number of random events during high repetitions. The mean of such normal distribution should indicate the steady state, but one cannot be sure until we reduce the variance of the distribution. Hence, we decided to formulate a criterion to determine the run time for simulation such that average buffer levels approach a steady state.

Convergence criterion: Instead of terminating the simulation at a fixed run time, at end of simulation we re-run the simulation for three times to check the standard deviation of the throughput and average buffer levels over the three runs. If the standard deviation is less than $\epsilon = 10^{-5}$ we will terminate the simulation else we repeat the process by doubling the $t_{simulation}$. The results for a four and eight machine line are presented in figure 5. The distributions have lower variation when compared to previous figure 4, providing more confidence on the steady state for the average buffer levels.

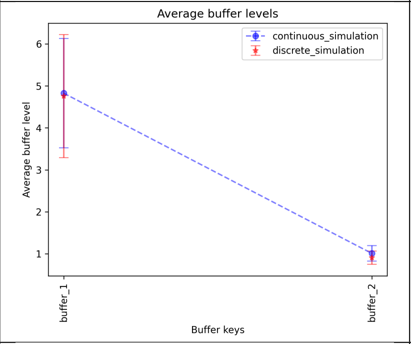
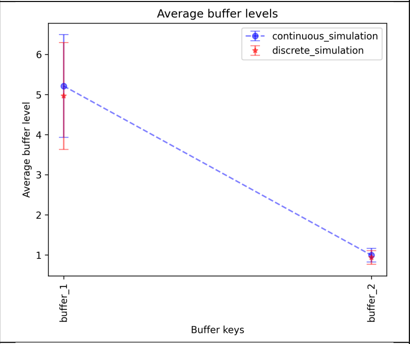
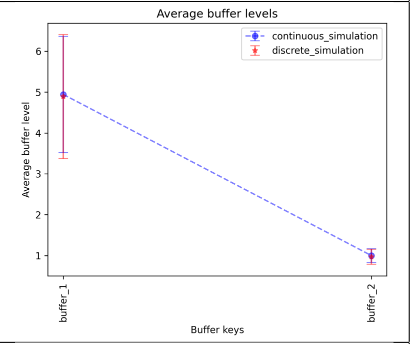
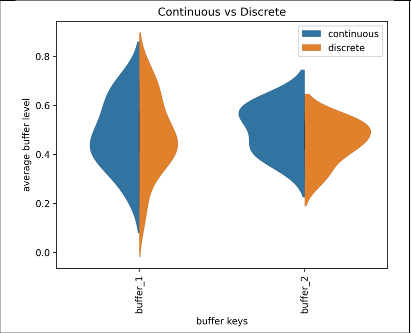
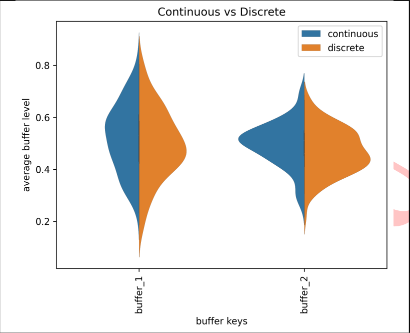
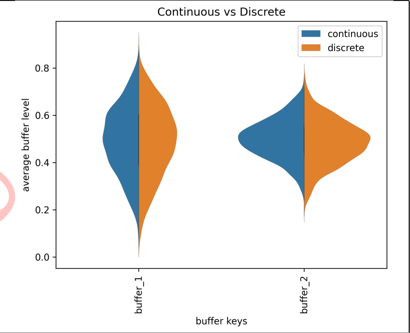
$t_{simulation} = 10days$	$t_{simulation} = 100days$	$t_{simulation} = 1000days$
		
		

Table 4: Distribution of average buffer levels with increase in number of repetition in simulation.

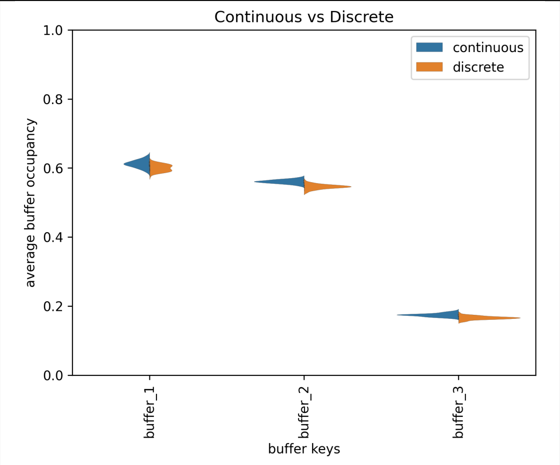
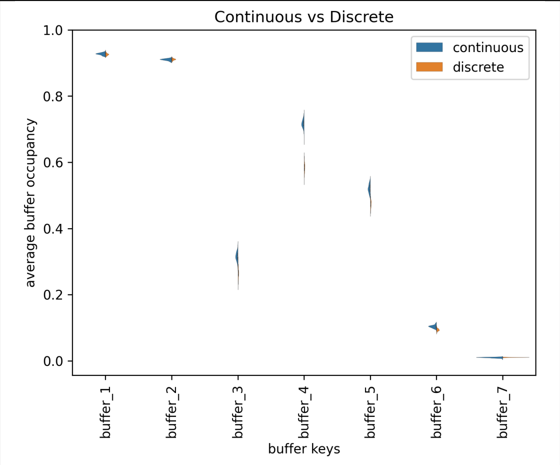
Four machine line	Eight machine line
	

Table 5: Distribution of average buffer levels for long lines.

4.2 Conclusions

In this study we presented an algorithm for continuous simulation model that can predict the average buffer levels and throughput. The predictions have minimum error when compared to discrete event simulation. Since, continuous and discrete simulations were written in different programming languages, the quantification of its computational efficiency is yet to be conducted. This work will be instrumental in developing later improvements to handle.

- Multiple part type.
- Assembly, disassembly lines and loops.

This work also provided a deep insight into mechanics of line and raised question on the evaluation of the steady state.

References

[1] Salvatore Scrivano, Barış Tan, and Tullio Tolio. Continuous-flow simulation of manufacturing systems with assembly/disassembly machines, multiple loops and general layout. *Journal of Manufacturing Systems*, 69:103–118, 2023.