An MDP-based Method for Dynamic Workforce Allocation in Bernoulli Serial Production Lines

Jiachen Tu and Liang Zhang

2023 IEEE 19th International Conference on Automation Science and Engineering (CASE) | 979-8-3503-2069-5/23/$31.00 ©2023 IEEE | DOI: 10.1109/CASE56687.2023.10260500

***Abstract*—With the development of Industry 4.0 in manufacturing, new technologies such as big data analytics, artificial intelligence, and cloud computing, have been widely deployed to production practice for performance evaluation and analysis. These technologies enabled fast and accurate production decision-making on the factory floor. To further support these activities, it's essential to develop real-time automated decision-making mechanisms backed by rigorous control and optimization analytics. The focus of this paper is on enhancing the throughput of production processes through the control and optimization of the floating workforce resources on the factory floor. Specifically, we consider serial production lines with finite buffers and machines characterized by the Bernoulli reliability model. Additionally, we assume that multiple shared workforce units can be allocated dynamically during production via real-time production bottleneck identification and mitigation in order to improve steady-state throughput. This paper addresses this problem for three and four-machine line cases, where two shared workforce units are dynamically allocated among the machines. An optimal control policy is derived based on the Markov decision process (MDP) approach. Numerical experiments are carried out to demonstrate the efficacy of the proposed method.**

I. Introduction

With the rapid development of Industry 4.0, some smart manufacturing technologies (for example, real-time sensing, big data, cloud computing) have been widely applied to the production system [1], [2]. Vast amount of data from the production process can be collected, transmitted, and analyzed. This facilitates the improvement of decision making, management, and production control for manufacturers. In a production system, bottleneck is typically defined as the operation that has the strongest effect on system performance [3]. In traditional production systems research, bottleneck are commonly considered as a static characteristic with respect to the system's steady state performance (see, for instance, [4]–[7]). However, this steady state-based approach forces the identification and mitigation of production bottlenecks into a relatively low-frequency activity. On the other hand, it is more desirable to be able to react timely to different production conditions and states and identify those impactful operations on-the-fly by taking advantage of the real-time data made available through the Industry 4.0 technologies. To achieve this goal, the notion of real-time bottleneck (RTBN) is introduced in [8]. Based on this study, the location of RTBN may be dynamically shifting among workstations as the system state evolves over time and RTBN is the one, whose improvement should be prioritized over all other machines under the current system state. Clearly, identification of RTBN is essentially a dynamic control problems and the goal is to obtain an optimal control policy that maximizes the efficiency of the production system. A similar study on machine switch-on/off control for energy efficient production is reported in [9].

In production systems research, Markov process models are commonly used to characterize the parts flow in a manufacturing process and have been successfully applied to numerous practical case studies (see, for instance, [3], [10], [11]). Under these Markovian models, the dynamic control problem in production systems can be viewed as decision-making optimization in Markov process, which is known as Markov decision process (MDP). Indeed, MDP has been widely applied in manufacturing and production analysis and control [12]–[14]. Traditionally, MDP is solved via dynamic programming (DP), which evaluates the value function of the control policy through the knowledge of the underlying system model. Based on the value function, the optimal policy is searched for by solving the sub-problems contained in Bellman equation. On the other hand, DP requires exact knowledge of the state transition probabilities of the system, which may face several challenges itself. First of all, the accuracy of the transition probabilities are highly dependent upon the system mathematical model and the identified parameters, which, in some practical applications, may be subject to various sources of uncertainties such as uncertainty of customer demands, operation uncertainty, production lead time uncertainty, quality uncertainty, machine reliability, etc. (see [15]–[17]). Moreover, even if a high-fidelity mathematical model is constructed, handling the full-scale mathematical model with all transition probabilities considered could be a computationally intractable for system with a large number of states (see [18], [19]). To tackle these challenges in traditional MDP problems, the reinforcement learning (RL) method has been developed, aiming at learning an optimal policy through an agent moving in an environment (see [20]–[22]). The common feature in RL scenarios is that the states of the system are determined by the RL control policies or actions. Once the policy is defined, as well as the current state, the next system state is fixed. As a result, during the computation of RL algorithms, knowing transition

This work was supported in part by the U.S. National Science Foundation (NSF), under Grant Number FM-2134367.

Jiachen Tu and Liang Zhang are with Department of Electrical and Computer Engineering, University of Connecticut, Storrs, CT, 06269, USA. Email: jiachen.tu@uconn.edu, liang.zhang@uconn.edu

probability is unnecessary and the policy dictates the state transitions directly.

In this paper, we study the Bernoulli serial line production system model, which is characterized by a Markov chain and model the dynamic workforce allocation problem using the MDP approach. The remaining of the paper is organized as follows: Section II introduces the mathematical model of the production systems considered and defines the problem addressed. Section III describes the solution to the problem for three- and four-machine lines with two shared helpers through an MDP policy iteration algorithm. Numerical experiments are carried out in Section IV to verify the performance of the proposed method. Finally, conclusions and future work are provided in Section V.

II. System model and problem formulation

A. Modeling

Consider a serial production line model shown in Figure 1 defined by the following assumptions:

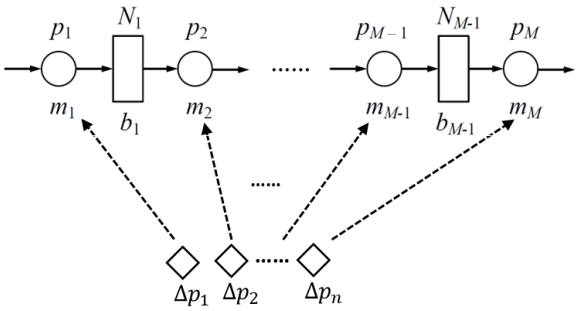


Fig. 1. Serial production line with Bernoulli machines and shared helpers

(i) The serial line has a total of M machines (indicated by the circles in Figure 1), $m_1, \ldots, m_M$, and $M-1$ buffers (indicated by the rectangles), $b_1, \ldots, b_{M-1}$.

(ii) The buffer b_i, $i = 1, \ldots, M-1$, has finite capacity N_i, i.e., it can accommodate at most N_i jobs.

(iii) The cycle times of all machines are identical and equal to τ. The system operates in discrete time slots. The duration of each time slot is τ.

(iv) The machines are unreliable and subject to random failures according to the Bernoulli reliability model. Specifically, during each time slot, the status of the machines, $\{1 = \text{up}, 0 = \text{down}\}$, are independent Bernoulli random variables. For machine m_i, $i = 1, \ldots, M$, it is up in a time slot with probability p_i and down with probability $1-p_i$. Bernoulli parameter p_i is also referred to as the *efficiency* of machine m_i.

(v) A machine is *starved* if it is up and its upstream buffer is empty at the beginning of this time slot. Machine m_1 is never starved for raw material.

(vi) A machine is *blocked* if it is up, its downstream buffer is full at the beginning of this time slot, and its immediate downstream machine is either down or blocked during this time slot. Machine m_M is never blocked.

(vii) If machine m_i is up and neither blocked nor starved, then it processes one job during this time slot, i.e., it picks up the job from its upstream buffer at the beginning of the time slot and places it in the downstream buffer at the end of the time slot.

(viii) A total of n *shared helpers*, $\eta_1, \ldots, \eta_n$, are available to be allocated to any machine during system operations. We assume that $n < M$ and that during one cycle time, each machine can only get one helper. In addition, assume that if helper η_j is allocated to machine m_i during a time slot, then during this time slot, machine m_i's efficiency is (temporarily) elevated to $p_i + \Delta p_j$. The efficiency of the machines not receiving a helper during this time slot stays at their original values. At the end of the time slot, the helpers leave the machines that they were allocated to, reset the machines efficiency back to p_i's, and await the new allocation for the next time slot. Moreover, it is assumed that

$$\max_i p_i + \max_j \Delta p_j \leq 1. \tag{1}$$

Remarks: In this paper, we assume that switching a helper from one machine to another occurs instantaneously in between cycles. Moreover, it is assumed that the efficiency elevation resulted from a given helper is fixed (i.e., operation-independent). More complicated scenarios will be studied in future work.

B. Problem formulation

A Bernoulli serial line defined by assumptions (i)-(vii) (i.e., without shared helpers) is characterized by a Markov chain with the states being the occupancy of the buffers. Let h_i, $i = 1, \ldots, M-1$, denote the number of jobs in buffer b_i and let $\boldsymbol{h} = [h_1, \ldots, h_{M-1}]$ represent the state of the underlying Markov chain. Clearly, $h_i \in H_i = \{0, 1, \ldots, N_i\}$ and the set of all states of this Markov chain is $\mathbf{H} = H_1 \times H_2 \times \cdots \times H_{M-1}$. Therefore, a state-based control policy of helpers allocation for systems defined by assumptions (i)-(viii) can be expressed as a mapping from $\mathbf{H}$ to $\{[i_1, i_2, \ldots, i_n] | i_j = 1, \ldots, M, j = 1, \ldots, n\}$, where i_j indicates the allocated machine of helper η_j:

$$\boldsymbol{\pi} : \mathbf{H} \rightarrow \{[i_1, i_2, \ldots, i_n] | i_j = 1, \ldots, M, j = 1, \ldots, n\}. \tag{2}$$

Let the control objective of the system considered be the maximization of steady state throughput (i.e., long term productivity). In the framework of the system model at hand, this amounts to the production rate, PR, of the system (expected number of jobs produced by m_M per time slot in steady state). Among all control policies $\boldsymbol{\pi}$ defined in (2), let $\boldsymbol{\pi}^*$ denote the optimal control that leads to the

maximal PR. Then, a machine is referred to as the *real-time bottleneck* (RTBN) under system state $\boldsymbol{h}$, $\boldsymbol{h} \in \mathbf{H}$, if a helper is allocated to this machine under $\boldsymbol{\pi}^*$ for state $\mathbf{h}$. Moreover, the machine receiving the largest efficiency elevation from the $\boldsymbol{\pi}^*$-allocated helper under state $\boldsymbol{h}$ is referred to as the *primary* RTBN under state $\boldsymbol{h}$.

Note that Bernoulli serial lines with only one shared helper in the system (and thus only one RTBN for each state), i.e., $n = 1$, have been studied in our previous work [8] and [23]). Specifically, the study in [8] shows that the threshold-based policy is optimal for the two-machine case and develops a procedure based on a set of heuristic rules to search for the optimal control policy in three-machine lines. Then, using the three-machine line result as a building block, a heuristic method for control policy optimization in longer production lines is derived in [8], which decomposes and represents the system into a group of virtual three-machine lines and determines the optimal control policy for each individual three-machine lines. Finally, the control policies for the individual virtual lines are combined to form the one for the overall system via a voting mechanism. In [23], an aggregation-based analytical algorithm is developed to calculate the transient and steady state performance metrics of the Bernoulli production system with one RTBN and develops a particle swarm optimization (PSO)-inspired approach for control policy optimization. However, due to the dramatic increase of the control policy space, production lines with multiple helpers are much more complicated and the approach for single-helper system cannot be directly applied. Therefore, the goal of this paper is to develop a new method for control policy optimization for in Bernoulli lines with multiple shared helpers. In particular, an MDP-based policy iteration algorithm is developed for Bernoulli lines with two shared helpers. Extension of the results to larger-sized systems will be conducted in future work.

III. MDP Model and Algorithm for Allocation of Shared Helpers in Bernoulli serial lines

In this paper, we consider the cases of Bernoulli lines with two shared helpers. That is, during each time slot, two machines in the system will be identified as RTBNs and allocated with the helpers η_1 and η_2. Note that for two machine lines with two shared helpers, it is possible to convert the problem to the single-helper problem addressed in [8]. For example, consider a two-machine Bernoulli line with parameters $p_1 = 0.7$, $p_2 = 0.8$, $N_1 = 5$, and $\Delta p_1 = 0.1$, $\Delta p_2 = 0.15$. Clearly, during any cycle time, each machine will get one helper and have its efficiency elevated by at least 0.1. The primary RTBN among the two will be allocated with $\Delta p_2 = 0.15$ leading to an addition improvement of 0.05 in machine efficiency. Thus, we can view the system being equivalent to a two-machine line with $p_1 = 0.8$, $p_2 = 0.9$ and a single helper with $\Delta p = 0.05$. Therefore, in this paper, we directly consider the Bernoulli lines with $M > 2$ machines.

In the case of an M-machine Bernoulli serial line, the total number of system states are $\prod_{i=1}^{M-1}(N_i + 1)$. For each system state, the number of possible allocation of n shared helpers is equal to the number of the n-permutations of M machines (assuming the helpers are different), i.e., $M!/(M-n)!$. Clearly, with the growth of M and n, the dimension of the solution space increases exponentially and it will become computationally intractable for traditional discrete optimization method to solve the problem with such a large solution space. Therefore, in this paper, we adopt the approach of MDP to conduct the optimization of the control policy for the underlying Markov system. Specifically, a policy iteration algorithm is developed that searches for the globally optimized control policy, leading to improved steady state PR. Details of the optimization problem formulation and the implementation of the algorithm for the systems considered are described below.

A. Mathematical model

In this subsection, we formulate the Markov decision process (MDP) model for the shared helper allocation problem in Bernoulli serial line production systems. Specifically, define 4-tuple, $(\boldsymbol{H}, \boldsymbol{A}, \boldsymbol{P_a}, \boldsymbol{R_a})$ to represent the Markov process model. Each term is illustrated below:

- $\boldsymbol{H}$: System state space. As discussed in Section II-B, the system state is defined as the number of jobs in each buffer.
- $\boldsymbol{A}$: Action space. For the problem addressed in this paper, the control action is the allocation of the shared helpers $\eta_1, \ldots, \eta_n$. The state-based decision-making of the control action is defined as control policy $\boldsymbol{\pi}$ expressed in equation (2). For convenience, let $\boldsymbol{\pi}_h^{(i)}$ denote the index of the machine that helper η_i is allocated to under system state $\boldsymbol{h}$.
- $\boldsymbol{P_a(h, h')}$: The transition probability from system state h to h' under action a.
- $\boldsymbol{R_a(h, h')}$: Immediate reward after system transitioning from state h to h' under action a. Since the goal of the problem is to maximize the steady state PR under the control policy, we define the reward as the expected production from the control action, i.e.,

$$R_a(\boldsymbol{h}, \boldsymbol{h'}) = \begin{cases} p_M + \Delta_j, \text{ if } h_{M-1} > 0 \text{ and } M = \boldsymbol{\pi}_{\boldsymbol{h}}^{(j)}, \\ p_M, \text{ if } h_{M-1} > 0 \text{ and } M \neq \boldsymbol{\pi}_{\boldsymbol{h}}^{(j)} \ \forall j, \\ 0, \text{ if } h_{M-1} = 0. \end{cases} \tag{3}$$

The optimization objective is to maximize the cumulative reward expressed as

$$E\left[\sum_{t=0}^{\infty} \gamma R_{a_t}(\boldsymbol{h}_t, \boldsymbol{h}_{t+1})\right], \tag{4}$$

where γ is the discount factor satisfying $0 \leq \gamma < 1$, a_t is the control action at time t, and $\boldsymbol{h}_t$ denotes the system state (buffers occupancy) at time t.

B. MDP solution algorithm

The general approach to solving MDP problem is using dynamic programming (DP). Define $V(\boldsymbol{h})$ as the value function of current state $\boldsymbol{h}$, which is the discounted sum of the rewards by following the control policy. Then, the DP algorithm can expressed using the following two equations:

$$V(\boldsymbol{h}) = \sum_{\boldsymbol{h'}} P_{\boldsymbol{\pi_h}}(\boldsymbol{h}, \boldsymbol{h'})[R_{\boldsymbol{\pi_h}}(\boldsymbol{h}, \boldsymbol{h'}) + \gamma V(\boldsymbol{h'})], \tag{5}$$

$$\boldsymbol{\pi_h} = \arg\max_{a} \sum_{\boldsymbol{h'}} P_{\boldsymbol{\pi_h}}(\boldsymbol{h}, \boldsymbol{h'})[R_{\boldsymbol{\pi_h}}(\boldsymbol{h}, \boldsymbol{h'}) + \gamma V(\boldsymbol{h'})]. \tag{6}$$

By iterating these two equations for all system states, the control policy will eventually converge to the optimal policy [24]. This method is called policy iteration. It is an iterative algorithm that alternates between policy evaluation and policy improvement until it converges to the optimal policy. The algorithm typically starts with a random or predefined initial policy. Then, at each iteration, the algorithm evaluates the policy using equation (5) and updates the optimal policy based on equation (6). It repeats these two steps until the policy converges. Detailed steps of the algorithm is given in the pseudo code below.

Algorithm 1 MDP control optimization by policy iteration

Initialization: Set $V(\boldsymbol{h}) = 0$, $V'(\boldsymbol{h}) = V(\boldsymbol{h})$, $\boldsymbol{\pi_h} = [1, 2, ...n]$, and $\boldsymbol{\pi'_h} = [n, n-1, ..., 1]$ for all $\boldsymbol{h}$, where n is the number of shared helpers, and set $\Delta_1 = \Delta_2 = 1$
while $\Delta_1 > 10^{-10}$ **do**
  Policy Evaluation:
  while $\Delta_2 > 10^{-10}$ **do**
    for each $\boldsymbol{h} \in \boldsymbol{H}$ **do**
      $V(\boldsymbol{h}) = \sum_{\boldsymbol{h'}} P_{\boldsymbol{\pi_h}}(\boldsymbol{h}, \boldsymbol{h'})[R_{\boldsymbol{\pi_h}}(\boldsymbol{h}, \boldsymbol{h'}) + \gamma V(\boldsymbol{h'})]$
    end for
    $\Delta_2 = \sqrt{\sum_{\boldsymbol{h}} (V'(\boldsymbol{h}) - V(\boldsymbol{h}))^2}$
    $V'(\boldsymbol{h}) = V(\boldsymbol{h})$
  end while
  Policy Improvement:
  for each $\boldsymbol{h} \in \boldsymbol{H}$ **do**
    $\boldsymbol{\pi_h} = \arg\max_a \sum_{\boldsymbol{h'}} P_{\boldsymbol{\pi_h}}(\boldsymbol{h}, \boldsymbol{h'})[R_{\boldsymbol{\pi_h}}(\boldsymbol{h}, \boldsymbol{h'}) + \gamma V(\boldsymbol{h'})]$
  end for
  $\Delta_1 = \sqrt{\sum_{\boldsymbol{h}} (\boldsymbol{\pi'_h} - \boldsymbol{\pi_h})^2}$
  $\boldsymbol{\pi'_h} = \boldsymbol{\pi_h}$
end while

C. Illustrative example

To illustrate the algorithm and the behavior of the control policy, an example is given below. Consider a three-machine Bernoulli serial line with parameters $p_i = 0.8$, $i = 1, 2, 3$, $N_1 = N_2 = 5$, and $\Delta p_1 = 0.1, \Delta p_2 = 0.15$. The MDP policy iteration algorithm described above is used to find the optimal policy. Note that the discount rate γ shows the degree, to which the agent cares about the rewards in the distant future. Therefore, in an infinite Markov chain problem, to evaluate the steady state performance metrics, γ should be selected close to 1. On the other hand, if γ is selected very close to 1, the iteration time needed to achieve convergence tends to be infinite. Here, we choose the discount rate γ as 0.999 and 0.9999 to illustrate the efficacy of the algorithm. The resulting control policies are shown in Fig. 2 for $\gamma = 0.999$ and in Fig. 3 for $\gamma = 0.9999$. Note that the color shading and the numbers in the blocks represent the indices of machines ($\boxed{1}$ for machine 1, $\boxed{2}$ for machine 2, and $\boxed{3}$ for machine 3) that is allocated with helper η_1 or η_2 under system state (h_1, h_2) by the control policy obtained.

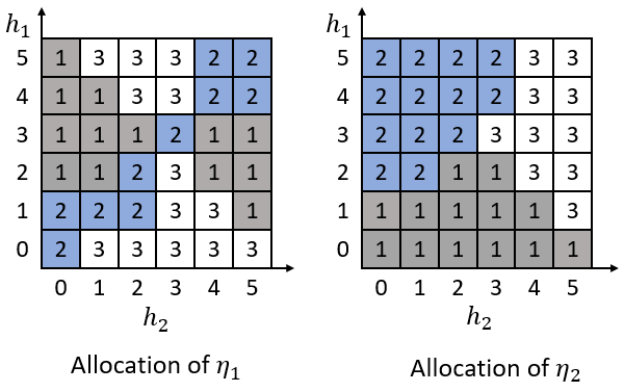


Fig. 2. Optimal control policy under $\gamma = 0.999$

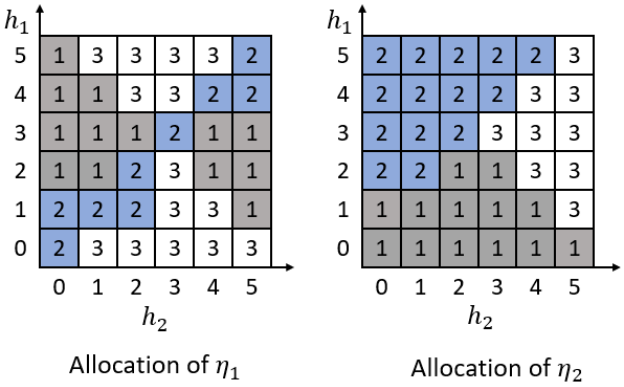


Fig. 3. Optimal control policy under $\gamma = 0.9999$

For this example, since $\Delta p_2 > \Delta p_1$, the machine receiving Δp_2 is viewed as the primary RTBN, which has the most impact on the long-term system performance at the current state. From Fig. 2 and Fig. 3, we can also see that when the second buffer is empty ($h_2 = 0$), with the increasing of the number of jobs in b_1, the primary RTBN (the one allocated with η_2) will switched to m_2 and relieve the load of buffer b_1. Similarly, with the increasing of the number of jobs in b_2, machine m_3 tends to be the primary RTBN to alleviate the burden of b_2. Comparing the optimization results summarized in these two figures, we can see that the control policies obtained under $\gamma = 0.999$ and $\gamma = 0.9999$ are quite

similar: They have the exact same helpers allocation for all system states except one. For the state that the allocation does differ ($\boldsymbol{h} = [5, 4]$), the two control policies still agree on the combination of machines to be allocated with helpers, with the only discrepancy being which particular machine receives η_1 and which receives η_2. Under these two control policies, the system's steady state production rates (evaluated using simulation) are $PR = 0.8796$ (for $\gamma = 0.999$) and $PR = 0.8797$ (for $\gamma = 0.9999$), respectively.

The steady state probability distributions of the buffers occupancy for the system are illustrated by the heat maps in Fig. 4 for both $\gamma = 0.999$ and $\gamma = 0.9999$ and also for the case where no shared helper is present. In the figure, the numbers in each block indicates the steady state probability of the corresponding buffer occupancy pair. As one can see, when no helper is present (Fig. 4(a)) the system spends significant amount of time (41.7%) in states where starvation and/or blockage are occurring. These states are the ones in the outer ring of the state maps, i.e., those with $h_i = 0$ or 5, $i = 1, 2$. However, under the optimal control policies obtained, the system only spends about 9% of time in those states and, instead, spend over 50% of time in *safer* states (those with $h_i = 2$ or 3). This also explains the efficacy of the control policies obtained.

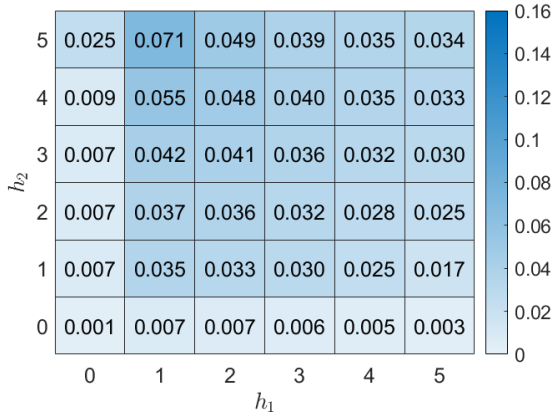
It should be noted that, with the increase of discount factor γ towards 1, the computing time of the policy iteration algorithm increases drastically (from about 10s to 70s on a PC with Intel® Core™ i7-10875H 2.3GHz processor and 16GB RAM with the algorithm implemented in C++). Thus, from the perspective of both system performance (i.e., PR) and computational time, it is not of essential importance to obtain the optimal policy for every single state system, as long as high quality control actions are obtained for the most frequently visited states.

IV. Numerical Experiment

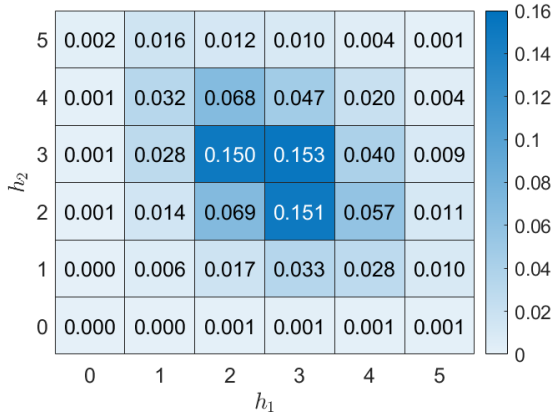
To understand the efficacy of the MDP approach for solving the helpers allocation problem, numerical experiments are carried out. Moreover, two heuristic-based policies are used to compare with the proposed MDP approach:

- *Upstream-first approach* (UPF): Sort Δp_j's in descending order. Allocate the helper with the largest Δp to the furthest upstream machine as long as its downstream buffer is not full at the begining of the present time slot, and repeat this until all helpers are allocated.
- *Downstream-first approach* (DNF): Sort Δp_j's in descending order. Allocate the helper with the largest Δp to the furthest downstream machine with a non-empty buffer immediately in front of it, and repeat this until all helpers are allocated.

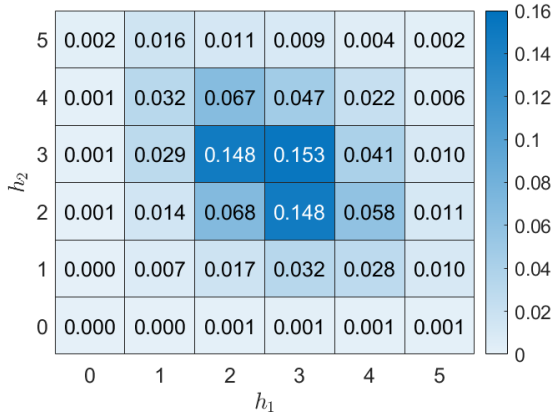
In this experiment, a total of 2,000 Bernoulli serial lines are random selected, with 1,000 for each $M \in \{3, 4\}$.



(a) With no shared helper



(b) Under optimal control policy for $\gamma = 0.999$



(c) Under optimal control policy for $\gamma = 0.9999$

Fig. 4. Steady state probability distributions of the buffers occupancy

The parameters of each line are randomly and uniformly generated from:

$$p_i \in (0.7, 0.9), \quad N_i \in \{2, 3, 4, 5\}, \\ \Delta p_j \in (0.1, 1 - \max_i p_i). \tag{7}$$

The steady state production rate PR is calculated by a C++ simulation program based on the corresponding control policy. In particular, for each line, the simulation program runs for 20 replications with the first 40,000 cycle times

in each replication being the warm-up time and the next 400,000 cycle times being the results collection time. Then, the mean values of the 20 replications are computed to obtain the steady state values of PR.

To quantify the efficacy of our method, the DNF method is used as the baseline for comparison and we define

$$\epsilon_{method} = \frac{PR^{method} - PR^{DNF}}{PR^{DNF}} \cdot 100\%, \qquad (8)$$

where $method \in \{MDP, UPF\}$. The results are shown in Table I.

TABLE I
AVERAGE IMPROVEMENT OF METHODS OVER DNF

	MDP	UPF
$M = 3$	10.96%	4.04%
$M = 4$	13.57%	5.68%

As one can see, the MDP-based approach performs significantly better than other two methods. The control policy resulted from the MDP approach can improve the steady state PR by over 10% compared with the DNP heuristic.

V. CONCLUSION

This paper studies the control policy of the allocation of multiple workforce units in Bernoulli serial lines. By using the Markov decision process approach and an policy iteration algorithm, we can obtain the optimal control policy for allocating the shared helpers in real time. The results of numerical experiments and examples of both three and four-machine cases indicate that this approach provides high quality control policies compared with two heuristic methods.

However, it should be noted that the computational complexity of the MDP policy iteration algorithm grows exponentially as the system size increases and it may become very time-consuming or even practically infeasible for even a moderate-sized system. Thus, our immediate future work is to develop alternative methods to extend the research to systems with more machines and shared workforce units (helpers). In addition, while the research of this paper focuses on steady state performance of production systems, we also intend to extend the results to systems operating in transients (e.g., systems with finite production runs).

REFERENCES

[1] H. S. Kang, J. Y. Lee, S. Choi, H. Kim, J. H. Park, J. Y. Son, B. H. Kim, and S. D. Noh, "Smart manufacturing: Past research, present findings, and future directions," *International Journal of Precision Engineering and Manufacturing-Green Technology*, vol. 3, pp. 111–128, 2016.

[2] P. Alavian, Y. Eun, S. M. Meerkov, and L. Zhang, "Smart production systems: automating decision-making in manufacturing environment," *International Journal of Production Research*, vol. 58, no. 3, pp. 828–845, 2020.

[3] J. Li and S. M. Meerkov, *Production Systems Engineering*. Springer, 2009.

[4] Z. Jia, L. Zhang, J. Arinez, and G. Xiao, "Finite production run-based serial lines with Bernoulli machines: Performance analysis, bottleneck, and case study," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 1, pp. 134–148, 2016.

[5] Z. Jia and L. Zhang, "Serial production lines with geometric machines and finite production runs: Performance analysis and system-theoretic properties," *International Journal of Production Research*, vol. 57, no. 8, pp. 2247–2262, 2019.

[6] Q. Chang, J. Ni, P. Bandyopadhyay, S. Biller, and G. Xiao, "Supervisory factory control based on real-time production feedback," *J. Manuf. Sci. Eng.*, vol. 129, no. 3, pp. 653–660, 2006.

[7] S. Biller, J. Li, S. P. Marin, S. M. Meerkov, and L. Zhang, "Bottlenecks in bernoulli serial lines with rework," *IEEE Transactions on Automation Science and Engineering*, vol. 7, no. 2, pp. 208–217, 2010.

[8] J. Tu, Y. Bai, M. Yang, L. Zhang, and P. Denno, "Real-time bottleneck in serial production lines with bernoulli machines: Theory and case study," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 4, pp. 1822–1834, 2021.

[9] Z. Jia, L. Zhang, J. Arinez, and G. Xiao, "Performance analysis for serial production lines with bernoulli machines and real-time wip-based machine switch-on/off control," *International Journal of Production Research*, vol. 54, no. 21, pp. 6285–6301, 2016.

[10] S. Ambani, S. M. Meerkov, and L. Zhang, "Feasibility and optimization of preventive maintenance in exponential machines and serial lines," *IIE transactions*, vol. 42, no. 10, pp. 766–777, 2010.

[11] Y. Bai and L. Zhang, "Recursive decomposition/aggregation algorithms for performance metrics calculation in multi-level assembly/disassembly production systems with exponential reliability machines," *International Journal of Production Research*, vol. 0, no. 0, pp. 1–26, 2023. [Online]. Available: https://doi.org/10.1080/00207543.2023.2166622

[12] C. Li, Y. Chen, and Y. Shang, "A review of industrial big data for decision making in intelligent manufacturing," *Engineering Science and Technology, an International Journal*, vol. 29, p. 101021, 2022.

[13] A. Bousdekis, K. Lepenioti, D. Apostolou, and G. Mentzas, "A review of data-driven decision-making methods for industry 4.0 maintenance applications," *Electronics*, vol. 10, no. 7, 2021.

[14] E. Ruschel, E. A. P. Santos, and E. de Freitas Rocha Loures, "Industrial maintenance decision-making: A systematic literature review," *Journal of Manufacturing Systems*, vol. 45, pp. 180–194, 2017.

[15] C. Li, F. Liu, H. Cao, and Q. Wang, "A stochastic dynamic programming based model for uncertain production planning of re-manufacturing system," *International Journal of Production Research*, vol. 47, no. 13, pp. 3657–3668, 2009.

[16] F. Pan and R. Nagi, "Robust supply chain design under uncertain demand in agile manufacturing," *Computers & Operations Research*, vol. 37, no. 4, pp. 668–683, 2010.

[17] G. P. Lechuga, F. V. Martínez, and E. P. Ramírez, *Handbook of Research on Modern Optimization Algorithms and Applications in Engineering and Economics*. IGI Global, Hershey, Pennsylvania, 2016.

[18] N. L. Zhang and W. Zhang, "Speeding up the convergence of value iteration in partially observable markov decision processes," *Journal of Articial Intelligence Research*, vol. 14, pp. 29–51, 2001.

[19] P. Poupart, "Exploiting structure to efficiently solve large scale partially observable markov decision processes," Ph.D. dissertation, University of Toronto, Department of Computer Science, 2005.

[20] T. Wuest, D. Weimer, C. Irgens, and K.-D. Thoben, "Machine learning in manufacturing: advantages, challenges, and applications," *Production & Manufacturing Research*, vol. 4, no. 1, pp. 23–45, 2016.

[21] P. D. Paraschos, G. K. Koulinas, and D. E. Koulouriotis, "Reinforcement learning for combined production-maintenance and quality control of a manufacturing system with deterioration failures," *Journal of Manufacturing Systems*, vol. 56, pp. 470–483, 2020.

[22] M. Panzer and B. Bender, "Deep reinforcement learning in production systems: a systematic literature review," *International Journal of Production Research*, vol. 60, no. 13, pp. 4316–4341, 2022.

[23] J. Tu and L. Zhang, "Performance analysis and optimisation of bernoulli serial production lines with dynamic real-time bottleneck identification and mitigation," *International Journal of Production Research*, vol. 60, no. 13, pp. 3989–4005, 2022.

[24] M. S. Santos and J. Rust, "Convergence properties of policy iteration," *SIAM Journal on Control and Optimization*, vol. 42, no. 6, pp. 2094–2115, 2004.