Dynamic Control of A Closed Two-stage Queueing Network for Outfitting Process in Shipbuilding

F. Dong^a, J.R. Deglise-Hawkinson^a, M.P. Van Oyen^a, D.J. Singer^b

^aUniversity of Michigan, Department of Industrial and Operations Engineering, 1205 Beal Ave., Ann Arbor, Michigan 48109, U.S.A

^bUniversity of Michigan, Department of Naval Architecture and Marine Engineering, 2600 Draper Road, Ann Arbor, Michigan 48109, U.S.A

Abstract

The U.S. Naval shipbuilding industry faces significant challenges to build ships on-time and within budgeted cost. To achieve greater efficiency and timeliness in shipbuilding, we developed a flexible two-stage queueing model under a CONWIP job release policy to enhance the planning and control of the outfitting process, one of the key processes in shipbuilding. The model is formulated using Markov Decision Processes which can provide (1) the optimal dynamic control policy, and (2) the optimal cost. The numerical results showed that the optimal control policy is a state dependent threshold type policy and very complex to analyze. Therefore, we developed a static model to simplify the dynamic model and used Mean Value Analysis to gain insights. Using both data from the dynamic model and the static model, we developed a regression model to calculate a threshold policy heuristic. Testing reveals that the performance of this heuristic is very close to the optimal.

Keywords: CONWIP, Flexible Queueing Networks, Shipbuilding, Outfitting, Markov Decision Processes, Regression, Threshold Policy, Mean Value Analysis

1. Introduction

“U.S. shipbuilders produce the finest warships in the world, but cost growth is eroding the purchasing power of the Navy”, which is warned in [1] by the Office of the Deputy Under Secretary of Defense. Over the last decade, the U.S. shipbuilding industry has improved significantly on productivity as a result of Navy and industry initiatives and investment. However, there are still large technology gaps in some U.S. shipyards that present opportunities to make further substantial improvements, particularly in the preproduction functions which include design, production engineering, and planning according to [1]. Research presented in this paper focuses on developing a flexible queueing model of ship outfitting processes and using Markov Decision Processes to develop a more efficient dynamic control in order to reduce the ship construction time and cost. At the same time, shipbuilding also motivates us to identify, design, and solve new problems in the control of flexible queueing networks.

1.1. Ship Outfitting Process and Obstacles to Improved Performance

Shipbuilding is a unique industry that uses a wide variety of manufactured components and requires a large number of workers possessing various skills as well as specialized facilities. There are two major processes in shipbuilding: hull construction and outfitting. Hull construction includes all the activities associated with fabricating and assembling the hull, while outfitting refers to the process of fabrication and installation of nonstructural components. Hull construction normally uses the block construction method, which is also called *Modularization*. This construction method has been used in ship production since World War II. The ship is divided geometrically into blocks. Depending on the ship size and type, a typical ship may consist of hundreds of blocks. The main hull construction work flow

contains the processes of part assembly, block assembly, grand block assembly, and final hull construction, as pictured below in Figure 1. Typically, ship blocks are assembled in a block assembly area and then joined together to form grand blocks in a nearby area, these grand blocks are lifted by large cranes to the drydock for the final erection.

Figure 1. Hull Construction Processes

Outfitting in shipbuilding includes painting, pumps, piping systems, main propulsion system, electrical system, air conditioning (HVAC), etc. It represents as much as 50% of the cost of the ship and up to 50% of ship construction time in many instances (see [2]). The scheduling of outfitting greatly depends on the schedule of the hull construction and therefore it is a highly integrated assembly process. Outfitting activities can almost always be performed during or after any stages of the hull construction. It is nearly always more efficient to process the outfitting at an early stage due to better ergonomics with easier access and greater feasibility to employ large machinery.

Research presented in this paper focused on the ship outfitting process. Taking the sequence in which blocks are assembled as an input, our model and methodology contribute toward an effective dynamic control policy for the best stage at which to perform a block's outfitting: block assembly or grand block assembly.

The challenges of controlling the best stage at which to perform outfitting arises in part from the time and cost differences of processing the same outfitting work at different stages. The outfitting process can delay the entire ship production system due to unexpected delays, system variations, capacity limitations, and technological constraints. Contemporary outfitting planning relies heavily on the experience and judgment of key personnel. The combinatorial and stochastic nature of outfitting planning and the lack of prior efforts to incorporate control methodologies suggests that there is an opportunity to create new models for ship outfitting processes to enhance overall shipbuilding performance. Although our model makes simplifying assumptions, our goal is to contribute a stochastic model which can capture the variability of ship outfitting in order to provide a more effective and accurate outfitting planning decision and control approach.

1.2. Literature on Ship Outfitting Planning

In the past, there has been little operations research literature in the area of ship outfitting. Some early production research on ship outfitting includes [2], [3], and [4]. [3] introduces the basic outfitting processes and the Zone Outfitting Method. [4] describes the planning and execution of pre-outfitting in structural assemblies. [2] clearly discusses and analyzes current outfitting problems, and uses a deterministic activity network model to formulate the outfitting planning problem. A mixed integer programming model of outfitting with sequence constraints was developed to address this outfitting planning problem. More recent research on the scheduling of the outfitting process has been conducted in [5], which provides a methodology that can automatically generate an outfitting sequence and planning.

All these studies only consider the deterministic processing time of the outfitting process and provide static outfitting planning. Approaches such as these will fail when dealing with common problems in shipbuilding, such as demand change during production and high variability in processing times. Therefore, we must investigate the outfitting problem with variability and develop a control policy that can change dynamically according to the stage of the shipbuilding system. Our previous research in ship outfitting in paper [6] presented a static two-stage $G/G/1$ queueing model using Kingman's approximation to provide the outfitting work distribution policy at strategic level. The model presented in this paper is a dynamic queueing model using Markov Decision Processes to analyze the optimal control of the outfitting activities based system dynamics. To our knowledge, this is a rare model for which Mean Value Analysis (MVA), a static model, is a fairly effective approximation for the dynamic control problem. This MVA approximation is then used to provide an improved heuristic by incorporating the MDP results over a training test suite to yield an effective regression-based heuristic (which is another approach that has not received much attention).

The remainder of this paper is organized as follows. In Section 2, we develop a two-stage closed queueing network for the outfitting process and formulate the problem using Markov Decision Processes (MDP). The numerical

examples from the MDP illustrate the structure of the optimal control policy, which extensive numerical analysis suggests to be of threshold type. The optimal control policy is difficult to strongly characterize by a theoretical analysis of the MDP model, as structural properties for this problem are not very effective in defining a heuristic. Therefore, we develop a static queueing model in Section 3 and use MVA to gain more insight into this model. In Section 4, a heuristic is developed using a regression model and information from both the dynamic and static models, which is more effective in controlling the outfitting activities and also simpler to implement. Finally, we discuss conclusions and directions for future research in Section 5.

2. The Model

A dynamic queueing model is developed to search for an efficient control policy which can allocate the outfitting work according to the system states. Figure 2 illustrates the structure of the model. There are two processing stages in this model: stage 1 (the second element illustrated in Figure 1) represents the general assembly process associated with the block construction, while stage 2 represents the grand block construction. This model presents a simpler approximate representation of practice, but it is still a challenging model to analyze, and it is expected to provide value in advancing practice.

Figure 2. Two Stage Outfitting Model

There are three tasks in this model: block sub-assembly work (S) at stage 1, grand block assembly work (A) at stage 2, and outfitting work (O). Jobs (blocks) are released at stage 1 (block assembly), upon a completion from stage 2 (grand block assembly), thereby maintaining a CONWIP system with K jobs/blocks in the system at all times (see [7]). Task S can only be processed at stage 1 and task A can be only processed at stage 2, while task O can either be processed at stage 1 or stage 2. Based on the observed state of the system, a control action is exercised for every job/block at stage 1 when it reaches the head of the line: whether the outfitting activity should be processed at stage 1 or stage 2. The capacity of the workshop at stage 1 is one, meaning that at most 1 block sub-assembly can be built at a time. This is a consequence of the fixtures, infrastructure, and partial automation that is employed at stage 1. On the other hand, the capacity of the workshop at stage 2 is equal to the CONWIP level K. Up to K grand block assemblies may be constructed at a time in stage 2, because shipbuilding is typically a labor intensive process after the structural work of block assembly is finished, and the capacity of human labor applied is quite flexible at stage 2. Obviously, the maximum number of blocks at stage 2 is limited to the maximum WIP level, K. Transfer times between stages 1 and 2 are taken to be negligible. Station 1 can starve for lack of work if all K jobs at stage 2. We take the job/block sequencing decisions to have been made in advance (based on the hull construction master schedule); therefore we do not model the block scheduling problem. Because the hull construction sequence is not modeled (a complicated problem in its own right). In our model, the released blocks are statistically identical with respect to processing times for assembly and outfitting upon release to the input of stage 1 (an assumption which will fit better for some ships than others based on the variations across blocks). To study sophisticated dynamic policies, the model allows all of the system state information to be used when deciding at which stage the outfitting activity should be processed.

The processing times for all the tasks are modeled as exponentially distributed random variables. As is standard in MDP models, the ability to optimize dynamic control usually requires the use of exponentially distributed processing times (as was the case in our prior paper [8]). In practice, U.S. shipyards have complicated planning and scheduling systems for procurement and the entire production process; however, to the best of our knowledge stochastic and dynamic decision/control modeling for outfitting to enhance production performance is new. During construction, ship blocks are complex three-dimensional structures and defects require real time rework. Therefore, design changes during ship production (especially for navy ships which are vulnerable to shifting capability requirements or the introduction of new technology) and defects cause a lot of delay due to rescheduling and rework. An example from [1] is the USS San Antonio (LPAD 17), the lead ship of her class of amphibious transport docks or landing platform docks, for which the originally scheduled delivery of July 2002 was severely delayed, with the final commissioning over three years late.

The processing rates of task S and task A are μ_S and μ_A respectively. The processing rate of sub-assembly work plus outfitting work at stage 1 is μ_{SO} , and the processing rate of grand block assembly work plus outfitting work at stage 2 is μ_{AO} . C_{SO} is the lump sum cost of the outfitting work being processed in the first stage, and C_{AO} is the lump sum cost of the outfitting work being processed in the second stage. γ is the lump sum reward rate of the finished block. The entire system is under a Constant Work In Process (CONWIP) release policy with CONWIP level K . The objective is to effectively allocate the outfitting work to minimize the long run average cost.

2.1. Literature Related to Modeling and Analysis

Having reviewed literature on ship outfitting planning in Section 1.2 above, we now focus on the technical approach. This CONWIP flexible queueing model is designed for the ship outfitting problem. However, a growing body of recent research has studied the control of flexible queueing networks. A similar model has been studied in [9], which considered a problem with optimal work sharing between two adjacent workers, each of whom processes a fixed task in addition to their shared tasks. It uses a MDP model that develops a threshold worksharing policy to improve the system throughput and lower the Work in Process (WIP) level. Our model has cost associated with each task and the objective is to minimize the performance matrix associated with both cost and system throughput. [10] studied an optimal stochastic scheduling model of a two-stage tandem queue with two parallel servers (unlike our serial model), and the objective was to minimize the total holding cost. However, they assumed no external arrivals to the system (static model) and allowed preemption.

Much work has been done in the area of CONWIP systems (see [7]). In [11] they considered two-station lines with exponential processing times and use Markov chain models to demonstrate the effectiveness of the Shortest Processing Time first (SPT) type of policies. In [12] they studied a CONWIP system with two cross-trained workers and characterize an optimal worker-to-task assignment policy, which is a Fixed Before Shared (FBS) principle prioritizing serving the fixed task. [13] introduced a new canonical model of zone-based cross-training workers and develops a methodology to employ it in U-shaped CONWIP lines. [14] considered a two-stage dynamic line balancing model under partial cross-training and define the trade-off between investing in Work-In-Process (WIP) and investing in workers.

In contrast, our work has to determine when to perform outfitting work is agile to perform at either the first or the second stage (a form of flexibility), but with different processing times and costs. The objective also considers the throughput reward which complicates the tradeoff that must balance the cost of production. It is challenging to design an effective control policy to balance the production cost and the system throughput at the same time. In the next section, a Markov Decision Process model is developed to analyze the system and explore the structure of an optimal control policy over many problem instances.

2.2. Markov Decision Process Formulation

For the Markov Decision Process formulation, the state variables are $\{n_S, n_{SO}\}$, where n_S is the number of blocks which only have steel work being processed at the first stage (i.e., those that still have outfitting and assembly work to be processed at the second stage), and n_{SO} is the number of blocks which have had both steel work and outfitting work processed at the first stage (i.e., these only have assembly work left at the second stage). The total number of blocks in the system is the CONWIP level K . Therefore, given $\{n_S, n_{SO}\}$, the number of blocks at stage 1 is $K - n_S - n_{SO}$. Let Λ denote the uniformization factor and $\Lambda = \mu_S + K\mu_A$. The objective is to minimize the long run average cost per stage.

Let $J_k(n_S, n_{SO})$ denote the optimal k-stage cost to go function, and set the terminal cost function, $J_0(n_S, n_{SO}) = 0$. To maximize insight, we write the Bellman equation in terms of a standard value iteration recursion (see Appendix A for further details).

$$J_{k+1}(n_S, n_{SO}) = \frac{1}{\Lambda} [\mathbb{I}_{\{n_S \geq 1\}} n_S \mu_{AO} (-\gamma + J_k(n_S - 1, n_{SO})) + \mathbb{I}_{\{n_{SO} \geq 1\}} n_{SO} \mu_A (-\gamma + J_k(n_S, n_{SO} - 1)) + \mathbb{I}_{\{n_S + n_{SO} < K\}} \min\{\mu_{SO}[C_{SO} + J_k(n_S, n_{SO} + 1)] + (\mu_S - \mu_{SO})J_k(n_S, n_{SO}), \mu_S[C_{AO} + J_k(n_S + 1, n_{SO})]\} + (\Lambda - \mathbb{I}_{\{n_S \geq 1\}} n_S \mu_{AO} - \mathbb{I}_{\{n_{SO} \geq 1\}} n_{SO} \mu_A - \mathbb{I}_{\{n_S + n_{SO} < K\}} \mu_S) J_k(n_S, n_{SO})]. \quad (1)$$

In the value iteration (A.3), the first term and second term denote the job throughput (or departure rate) from the second stage. The indicator function $\mathbb{I}_{\{n_S \geq 1\}}$ identifies that there is at least one block without outfitting at the second stage, and that the probability of finishing one block without outfitting is $n_S \mu_{AO} / \Lambda$. The $\mathbb{I}_{\{n_{SO} \geq 1\}}$ indicator function indicates that there is at least one block with outfitting at the second stage, and that the probability of finishing the block is $n_{SO} \mu_A / \Lambda$. γ is the lump sum reward of the finished block. In (A.3), the term $-\gamma$ is negative since the objective is to minimize the cost. The third term in this function is the decision term, which determines whether to process outfitting at stage 1 or stage 2, to minimize the cost per stage. The indicator function $\mathbb{I}_{\{n_S + n_{SO} < K\}}$ provides the constraint that there is at least one block at stage 1 since $n_S + n_{SO}$ is the total number of blocks at stage 2. The last term is the uniformization term.

2.3. Numerical Examples

To gain insight into the structure of the optimal control policy, we provide a set of illustrative numerical examples that is computed via the MDP formulation presented in Table 1. For numerical stability, we use the relative value iteration algorithm to generate the numerical results (see Appendix A).

Cases	1	2	3	4	5	6	7	8
K	10	10	10	10	10	10	10	10
C_{AO}	1	1	1	1	2	2	2	1
C_{SO}	1	1	1	1	2	2	1	2
γ	2	2	2	2	1	1	1	1
μ_S	5	5	5	5	5	5	5	5
μ_{SO}	2.5	3.5	2.5	2.5	3.75	3.75	3	3
μ_A	1	1	1	1	0.5	0.5	1	1
μ_{AO}	0.1	0.1	0.01	0.75	0.375	0.35	0.5	0.5

Table 1. Numerical Example Test Cases

Cases 1 through 4 study $\gamma > \min\{C_{SO}, C_{AO}\}$, for which the system has a larger reward for throughput than the offsetting cost of outfitting at either stage. Thus, the optimal control policy aims to improve the system's efficiency. All the test cases satisfy the constraint $(1/\mu_{AO} - 1/\mu_A) > (1/\mu_{SO} - 1/\mu_S)$, i.e. the outfitting at the second stage takes more time.

Figure 3 illustrates the numerical results of test cases 1 and 2. The x-axis is n_S , the number of blocks without outfitting at the second stage, and the y-axis is n_{SO} , the number of blocks with outfitting at the second stage. In this figure, “1” denotes that the optimal action at that particular state is processing outfitting at stage 1, while “2” denotes that the optimal action at that particular state is completing outfitting at stage 2. Only decisions for states with $n_S + n_{SO} \leq K$ are indicted due to the WIP constraint of K from CONWIP control policy. When $n_S + n_{SO} = K$, no decision is made since there is no block in stage 1.

Figure 3 (a) shows that when n_S and n_{SO} are smaller than a certain threshold, outfitting should be completed at stage 2. In all other cases, it is better to complete outfitting at stage 1. Logically, when n_S and n_{SO} are small, there

Figure 3. Outfitting MDP Numerical Example 1: $\gamma > \min\{C_{SO}, C_{AO}\}$

are not many blocks at the second stage, and it is not efficient to idle the workshops (or workers) at the second stage. Therefore, it is optimal to process outfitting at the second stage when n_S and n_{SO} are small. When there is a sufficient amount of jobs at the second stage, it is favorable to process outfitting at the first stage since the single workshop at the first stage is more efficient with outfitting work. For case 2, μ_{SO} increases from 2.5 to 3.5, which indicates the process of outfitting at first stage is more efficient, and as expected the region of states in which “1” is optimal increases. Figure 3 (b) shows that the optimal control will assign more outfitting work to stage 1 and less outfitting work to stage 2, but one cannot predict exactly which states will be changed.

Figure 4. Outfitting MDP Numerical Example 2: $\gamma > \min\{C_{SO}, C_{AO}\}$

In case 3, the processing rate of outfitting at the second stage, μ_{AO} , is decreased from 0.1 to 0.01. Figure 4 (a) shows the scenario where outfitting is only processed at the second stage, when $n_S = 0$ and n_{SO} is smaller than a threshold. Most outfitting work should be processed at the first stage. In case 4, μ_{AO} is increased from 0.1 to 0.75 relative to case 1. Figure 4 (b) demonstrates a scenario where all outfitting processing occurs at the second stage. Although processing outfitting at the second stage is less efficient than processing it at the first stage, it is still optimal to complete all of the outfitting at the second stage because there are K workshops at the second stage and the capacity

is much larger than stage 1. Therefore, to maximize the system throughput (for reasons mentioned above), it is optimal to use all the capacity at the second stage.

Figure 5. Outfitting MDP Numerical Example 3: $\gamma < \min\{C_{SO}, C_{AO}\}$

Cases 5 and 6 are the cases where $\gamma < \min\{C_{SO}, C_{AO}\}$, i.e. the throughput reward is smaller than the cost for each block. Therefore, the optimal control policy will focus on minimizing the cost instead of maximizing the throughput. Case 5 is depicted in Figure 5 (a) and shows an “opposite” threshold control policy as compared to case 1: when n_S and n_{SO} are smaller than a threshold, outfitting should be processed at the first stage, and when n_S and n_{SO} are greater than a threshold, outfitting should be processed at the second stage. In case 6, the processing rate of outfitting at the second stage, μ_{AO} , decreases from 0.375 to 0.35. Figure 5 (b) shows that there is more outfitting being processed at the second stage. This seems counterintuitive since processing outfitting at the second stage is less efficient than in case 5, there should be less outfitting work processed at the second stage. However, because $\gamma < \min\{C_{SO}, C_{AO}\}$, the optimal control policy from the MDP will slow down the system’s throughput and minimize the cost from C_{SO} and C_{AO} .

Figure 6. Outfitting MDP Numerical Example 4: Change of C_{SO} and C_{AO}

Cases 7 and 8 exemplify the impact of changing C_{SO} and C_{AO} on the optimal control policy. In case 7, $C_{AO} = 2$,

$C_{SO} = 1$, and $\gamma = 1$, which indicates that it costs more to process outfitting at the second stage. Figure 6 (a) shows that it is optimal to process all the outfitting work at stage 1. In case 8, $C_{AO} = 1$ and $C_{SO} = 2$, which indicates it costs more to process outfitting at first stage (which is in the actual shipyard, but this case is used solely to show how cost can impact the optimal control policy). Figure 6 (b) shows that it is optimal to process all the outfitting work at stage 2. Case 7 and 8 show that C_{SO} and C_{AO} play a very important role in the control of allocating outfitting work to the most beneficial stages.

The numerical study suggests that the optimal control policy is a threshold type policy, but there is no strong structure (e.g. a priority rule) to be found via analytical methods. Proving the optimality of a threshold type policy will not yield a useful heuristic, nor will it yield additional deep insights, so we do not pursue that path. On the other hand, we have discovered a simplified static model that is surprisingly effective at yielding insight and a highly effective and fast MVA algorithm that is useful in our heuristic approach. Thus we set our focus on methods to develop the desired heuristic policy as well as insights to be gained through approximations and computational analysis.

3. A Static Model Using Mean Value Analysis

Traditional CONWIP models without dynamic control can be analyzed by using a technique known as Mean Value Analysis (MVA). MVA requires a clearly defined static server discipline, so it is impossible to solve the dynamic queueing model using MVA. MVA calculates the mean queue size, mean waiting times, and throughput in a closed queueing network by exploiting a product-form solution [15]. MVA is specifically used here to provide an approximate model for further understanding of the system behavior under different circumstances. One difference from the dynamic model is that instead of controlling the outfitting activities at the first stage, the approximation is a static policy that generates an independent and identically distributed (i.i.d.) sequence of Bernoulli(α) random variables (r.v.'s) to randomly select the outfitting work to be processed at stage 1 or stage 2. On average, there is $\alpha\%$ of outfitting work being processed at stage 1, and $(1 - \alpha)\%$ of outfitting work being processed at stage 2. Figure 7 illustrates this static queueing model using Bernoulli(α) r.v.'s to select the outfitting work. The decision variable is the Bernoulli parameter α and the goal is to find the optimal α^* which minimizes the system average cost including outfitting cost and throughput reward.

Figure 7. Mean Value Analysis Model

First, several parameters are calculated in order to develop the MVA recursive function. π_1 and π_2 are used to demonstrate the un-normalized equilibrium distribution of a job being at stage 1 and stage 2. $\pi_1 = \pi_2 = 1$ since this is a tandem, two stage closed network that requires all jobs going through stage 1 to also go through stage 2. μ_1 and μ_2 are the average processing rates at stage 1 and stage 2 when $K = 1$, where $\frac{1}{\mu_1} = \alpha \frac{1}{\mu_{SO}} + (1 - \alpha) \frac{1}{\mu_{AO}}$ and $\frac{1}{\mu_2} = \alpha \frac{1}{\mu_A} + (1 - \alpha) \frac{1}{\mu_{AO}}$. $W_i(n)$ is the average job waiting time at stage i with n jobs in the system. $L_i(n)$ is the average queue length at stage i with n jobs in the system. $\lambda_i(n)$ is the throughput rate at stage i with n jobs in the system.

3.1. MVA Recursive Function and Iteration

For clarity, we justify the recursion. When $K = 1$, the average waiting time at stage i is:

$$W_i(1) = \frac{1}{\mu_i}, i = 1, 2 \quad (2)$$

The average throughput rate at stage i is:

$$\lambda_i(1) = \frac{1}{\pi_1 W_1(n) + \pi_2 W_2(n)}, i = 1, 2 \quad (3)$$

Based on Little's Law, the average queue length is as follows:

$$L_i(1) = \lambda_i(1)W_i(1), i = 1, 2 \quad (4)$$

For $K = n$, the recursive equations are as follows:

$$W_1(n) = (L_1(n-1) + 1) \frac{1}{\mu_1} \quad (5)$$

$$W_2(n) = \frac{1}{\mu_2} \quad (6)$$

$$\lambda_i(1) = \frac{n}{\pi_1 W_1(n) + \pi_2 W_2(n)}, i = 1, 2 \quad (7)$$

$$L_i(n) = \lambda_i(n)W_i(n), i = 1, 2 \quad (8)$$

The cost function with variable α is:

$$(\alpha C_{SO} + (1 - \alpha)C_{AO} - \gamma)\lambda(\alpha, K), \quad (9)$$

where $\lambda(\alpha, K)$ is the system throughput which can be calculated using the MVA recursive functions. The system throughput only depends on the value of α and the CONWIP level K .

The goal is to find the optimal α^* that minimizes the cost function (9). In the next section, some numerical examples will be presented to evaluate the performance of this MVA model.

3.2. Numerical Example

The test-suite detailed in Table 2 is designed to provide insight into how the change of CONWIP level K and various system dynamics may impact the optimal α^* .

K	3, 5, 10
C_{AO}	1
C_{SO}	1
γ	3
μ_S	5
μ_{SO}	3
μ_A	5
μ_{AO}	2.5 ~ 0.1

Table 2. MVA Test-Suite

Figure 8 represents the numerical example of Table 2. The y axis in this graph is the value of α^* , the optimal fraction of outfitting work to be processed at stage 1, discretized over $[0,1]$. The x axis is the value of μ_{AO} . μ_{AO} decreases from 2.5 to 0.1, indicating that it will take longer to process outfitting work at stage 2. When μ_{AO} is large, $\alpha^* = 0$, which shows that it is optimal to process all the outfitting work at stage 2. When μ_{AO} is small, $\alpha^* = 1$, which means it is optimal to process all the outfitting work at stage 1 when processing outfitting at stage 2 takes too long. α^*

increases when μ_{AO} decreases indicating that when the process of outfitting work at stage 2 is becoming less efficient, more outfitting work should be processed at stage 1.

The different curves show that when K increases, less outfitting work should be processed at stage 1 and more outfitting work will be processed at stage 2. It is because the capacity of stage 1 is one and the capacity of stage 2 is K . Therefore, when the CONWIP level K increases, more outfitting work assigned to stage 2 can reduce the congestion at stage 1, while maintaining a high system throughput.

Figure 8. MVA Numerical Example

The numerical example shows a very reasonable behavior of the static model analyzed by MVA. In addition to quantifying the value of a dynamic policy, the purpose of developing the MVA model is to use the information provided by this MVA model and design an effective policy to control the dynamic model. Therefore, it is necessary to compare the performance between the MVA and MDP models and analyze the difference in system behavior.

3.3. MVA vs MDP

To compare the performance between the dynamic and static models, the absolute percentage optimality gap is applied as the performance metric. Let $Z(\alpha^*)$ denote the optimal cost of MVA, and λ^* denote the optimal cost from the MDP. The absolute percentage optimality gap $|G|$ is defined as follows:

$$|G| = \left| \frac{Z(\alpha^*) - \lambda^*}{\lambda^*} \right| \cdot 100\%. \quad (10)$$

Cases	1	2	3
K	3	3	10
C_{AO}	1	1	1
C_{SO}	1	1	1
γ	3	3	3
μ_S	5	5	5
μ_{SO}	3	3 ~ 0.5	3
μ_A	5	5	5
μ_{AO}	2.5 ~ 0.1	0.5	0.5

Table 3. MVA vs MDP Test Suite

Table 3 presents a test-suite to compare the optimal cost from both the MVA and MDP models.

Figure 9. MVA vs MDP: Case 1

Figure 9 illustrates the optimal cost of MVA and MDP in case 1. It shows that the optimal cost from the MDP model is always smaller or equal to the optimal cost from the MVA model in case 1, which reflects the fact that the dynamic model cannot have worse performance. The graph also illustrates that when $\alpha^* \approx 0$ or $\alpha^* \approx 1$, the absolute percentage optimality gap $|G|$ is close to 0. $\alpha^* \approx 0$ indicates that it is optimal to process nearly all of the outfitting work at stage 2, while $\alpha^* \approx 1$ indicates that it is optimal to process nearly all outfitting work at stage 1. In such cases, the optimal control policy from the MDP model will be close to the output from the MVA model. When $0 < \alpha^* < 1$, the MDP model has a better performance than the MVA model. The largest absolute percentage optimality gap $|G|$ is 5.53% in the case where $\alpha^* \approx 0.5$. When $0 < \alpha^* < 1$, the system is under the most complex circumstances and it is necessary to utilize the dynamic model to provide an optimal control policy based on the system dynamics.

Figure 10. MVA vs MDP: Case 2

Figure 10 illustrates the cost of the MVA model and the MDP model in case 2 when the μ_{SO} decreases from 3 to 0.5. The CONWIP level is $K = 3$. It shows a similar result to case 1: the optimal cost from the MDP model is always smaller or equal to the optimal cost from the MVA model. The difference is that when μ_{SO} is large, $\alpha^* = 1$, which means processing all the outfitting at stage 1 when it is efficient. When μ_{SO} decreases, α^* also decrease from 1 to 0. When μ_{SO} is too small, no outfitting will be processed at stage 1.

Figure 11. MVA vs MDP: Case 3

Figure 11 illustrates the cost of the MVA model and the MDP model where the CONWIP level is $K = 10$. The absolute percentage optimality gap $|G|$ decreases when the CONWIP level K increases. When there are more jobs in the system, it is less sensitive to the “mistake” made by the random decision from the Bernoulli trial of the MVA model, since more jobs are in the systems, there is less chance to idle the workshops at both stage 1 and stage 2. Therefore, the performance of the MVA model approaches the efficiency of the MDP model when the number of jobs K increases in the system.

The numerical study above shows that the MVA model is a surprisingly good approximation of the MDP model, and the information provided by the MVA model should be valuable for designing the heuristic to control the dynamic model. In the next section, a regression based threshold policy is designed using both information from the MVA model and the MDP model to achieve the best performance we could get with an algorithm that would be more computationally efficient in real time use (at the expense of requiring significant pre-processing effort). This approach also has value in providing new insights into the control problem.

4. A Regression Based Threshold Policy

4.1. Conditions for Threshold Policy Type

Based on a computational search over a large test suite (see Table 4 for details) to identify the properties of an optimal policy, the structure of the optimal control policy was found to be a threshold policy in every case; however, there exist the following two types of threshold policies. Let $n_S^*(n_{SO})$ and $n_{SO}^*(n_S)$ denote the optimal threshold for n_S (when n_{SO} is fixed) and n_{SO} (when n_S is fixed). The type 1 threshold policy, shown in Figure 12 (a), illustrates that when $n_S \leq n_S^*(n_{SO})$ and $n_{SO} \leq n_{SO}^*(n_S)$, it is best to process outfitting at stage 2; otherwise, process outfitting at stage 1. The type 2 threshold policy in Figure 12 (b) indicates that when $n_S \leq n_S^*(n_{SO})$ and $n_{SO} \leq n_{SO}^*(n_S)$, outfitting should be completed at stage 1; otherwise, process outfitting at stage 2.

The type 1 and type 2 threshold policies show completely opposite policy structures, and the question remains to discover what the parametric conditions for each type of policy are. Based on the results of a large test-suite which includes more than 10,000 cases, we are able to identify numerically-established observations, including the conditions for type 1 and type 2 policies that were identified, as follows:

Condition 1: When $\gamma = \min\{C_{SO}, C_{AO}\}$, the optimal cost of the MDP model, λ^* , is in the neighborhood of 0, and the optimal cost of the MVA model has $Z(\alpha^*) = 0$ for most cases as well. The optimal control policies were all found to be strict priority policies that process outfitting at stage 1 when $C_{AO} \geq C_{SO}$, and perform outfitting at stage 2 otherwise.

Figure 12. Threshold Policy Types

Condition 2: When $\gamma > \min\{C_{SO}, C_{AO}\}$, the optimal cost of the MDP model, λ^* , is negative (which means that the production reward exceeds the holding cost). In all cases, the optimal control policy was found to be of type 1 threshold policy (which includes the degenerate cases of strict priority policies).

Condition 3: When $\gamma < \min\{C_{SO}, C_{AO}\}$, the optimal cost of the MDP model, λ^* , is positive, and in all cases the optimal control policies are type 2 threshold policy (which includes strict priority policies).

Condition 2, $\gamma > \min\{C_{SO}, C_{AO}\}$, indicates that the reward for each block is larger than the outfitting cost for each block. This condition is the only relevant one in practice, because the production line should be profitable (otherwise there would be no incentive to produce). Therefore, the focus in the remainder of the paper is to design a threshold policy under the condition $\gamma > \min\{C_{SO}, C_{AO}\}$, which is the regime in which type 1 threshold policies are optimal.

4.2. The Regression Model

While one could in practice compute $n_S^*(n_{SO})$ and $n_{SO}^*(n_S)$ for each value of n_S and n_{SO} , there are not many people in industry who are able to easily understand and perform these computations. It is also computationally expensive to optimize the problem directly. An effective heuristic will not only be more computationally tractable, but it will be designed to capture the key parameters and concepts that drive the performance of the optimal threshold policy so as to generate insight into these problems. Our previous research in the control of flexible queueing networks in paper [8] was the first, to our knowledge, to introduce the idea of using regression for heuristic policies for MDPs. In particular, that paper showed the effectiveness of using a regression model as a heuristic for controlling the “N” queueing network. Therefore, we now create a regression model to approximate the optimal policy via a linear threshold-type policy.

Figure 13 shows the structure of this heuristic threshold type policy, which has two variables n_S^* and n_{SO}^* , which are the intercepts from which the threshold line is linearly interpolated. When n_S is smaller than the threshold line (precisely, the projection of the line onto the x axis for the state n_{SO}), outfitting is processed at stage 2; otherwise, outfitting is processed at stage 1. This will usually differ from the optimal threshold policy, which may be nonlinear. While it is good for the shape of the heuristic policy to match that of the optimal policy, the quality of the heuristic is only judged through its cost criterion relative to the optimal cost. Let $\tilde{n}_S^*$ and $\tilde{n}_{SO}^*$ denote the threshold intercepts approximated by the regression model.

Figure 14 shows that the regression model uses the information from both the MVA model and MDP model. MDP solutions are generated and stored offline. After intensive pre-processing, the regression threshold heuristic is computationally efficient, with the bulk of the computations being the MVA recursion which is linear in K . There are four predictors from the MVA model: (1) α^* , the optimal percentage of outfitting work processed at stage 1 and the following three which are all computed under the optimal mix α^* ; (2) $\bar{n}_S$, the average number of blocks without

Figure 13. Type 1 Threshold Policy Regression Model

Figure 14. Regression Model

outfitting at stage 2; (3) $\bar{n}_{SO}$, the average number of blocks with outfitting at stage 2; and (4) λ , the system throughput of the static model. $K, C_{AO}, C_{SO}, \gamma, \mu_S, \mu_{SO}, \mu_A, \mu_{AO}$ are the system parameters from the MDP model. Some second-order terms of system dynamics are also included, such as $\ln(K)$, $\sqrt{K}$, $C_{AO} \cdot \mu_S$, $C_{SO} \cdot \mu_{SO}$, $\gamma \cdot \mu_A$, $\gamma \cdot \mu_{AO}$. Forward Stepwise Selection was applied to select the significant predictors, which yielded the 16 predictors.

4.2.1. Test Suite for the Regression Threshold Model to Define Problem Instance for Training and Testing

Table 4 represents the design of our test-suite of problem instances that will be randomly divided into training problem instances to generate the regression model as well as a set for testing this heuristic. This test-suite provides 13,824 test cases; however, these must be checked for appropriateness.

From this, only the test cases that satisfy the following four conditions were selected: (1) $\mu_S > \mu_{SO}$ (so outfitting makes stage 1 longer), (2) $\mu_A > \mu_{AO}$ (so that grand block assembly takes longer when outfitting is included), (3) $1/\mu_{AO} - 1/\mu_A > 1/\mu_{SO} - 1/\mu_S$, the outfitting at second stage takes longer time; (4) $\gamma > \min\{C_{SO}, C_{AO}\}$, reward for each block is larger than the minimum cost. Randomly selecting half of these cases for the regression training data set (with the remainder for testing), we had over 4,000 cases in the training set and over 4,000 in the testing set of instances. Clearly, this ensures that our measure of performance for any regression-based heuristic will not incorporate any data used to create the model.

K	5, 6, 7, 8, 9, 10
C_{SO}	5
C_{AO}	$(0.5, 1, 2, 4) \times C_{SO}$
γ	$(1, 2, 4) \times C_{SO}$
μ_S	1, 2, 5
μ_{SO}	$(0.75, 0.5, 0.1, 0.01) \times \mu_S$
μ_A	$(1.5, 1, 0.5, 0.1) \times \mu_S$
μ_{AO}	$(0.75, 0.5, 0.1, 0.01) \times \mu_A$

Table 4. Regression Test-Suite

4.2.2. Derivation of Regression Equations

First, we randomly selected half of the useable test cases from Table 4 and ran these problem instances through both the MDP model and the MVA model. The MVA model provides the value of α^* , $\bar{n}_S$, $\bar{n}_{SO}$, and λ for each test case. The MDP model provides the optimal threshold n_S^* and n_{SO}^* . $K, C_{AO}, C_{SO}, \gamma, \mu_S, \mu_{SO}, \mu_A, \mu_{AO}$ are the system parameters from the test-suite. Some second-order terms of system dynamics are also included, such as $\ln(K)$, $\sqrt{K}$, $C_{AO} \cdot \mu_S$, $C_{SO} \cdot \mu_{SO}$, $\gamma \cdot \mu_A$, $\gamma \cdot \mu_{AO}$. We use Forward Stepwise Selection to select the significant predictors and the regression equations of $\bar{n}_S^*$ and $\bar{n}_{SO}^*$ for $K \in 5, 6, 7, 8, 9, 10$ are as follows:

$$\begin{aligned} \bar{n}_S^* &= 3.982 - 4.614\alpha^* + 0.6761K - 0.0829C_{AO} + 0.1627\bar{n}_S + 0.00308\gamma\mu_A - 0.313\mu_{SO} \\ &\quad + 0.1579\bar{n}_{SO} + 0.126\mu_S + 0.015\gamma - 1.69\ln(K) + 0.062\lambda \\ \bar{n}_{SO}^* &= 2.201 - 4.246\alpha^* + 0.5103K - 0.1421C_{AO} + 0.0418\gamma + 0.01404\gamma\mu_A - 0.049\gamma\mu_{AO} \\ &\quad - 0.309\mu_{SO} + 0.212\bar{n}_{SO} + 0.133\bar{n}_S + 0.422\mu_{AO} + 0.1\mu_A - 0.0022C_{AO}\mu_S \end{aligned} \quad (11)$$

Forward Stepwise Selection selected 11 predictors for the regression equation of $\bar{n}_S^*$ with an R^2 value of 88.19%, while 12 predictors have been selected for the regression equation of $\bar{n}_{SO}^*$ with an R^2 value of 81.05%. The most significant factor for both equations is α^* . This is an interesting result, since α^* is calculated by the static model using Mean Value Analysis; however, it is able to capture the feature of the nonlinear queueing dynamics effectively despite being a static randomized policy. If α^* is the only variable used as the predictor, the R^2 value is 76.98% for $\bar{n}_S^*$ and 63.29% for $\bar{n}_{SO}^*$. K is the second most significant factor in the regression equations, which is intuitive because the thresholds should scale in an approximately linear manner (as confirmed by $\ln(K)$ and $\sqrt{K}$ being less predictive).

It may be preferable in application to use some simpler regression equations with fewer predictors. If α^* and K are the only predictors in the regression, the R^2 value is 84.21% for $\bar{n}_S^*$ and 71.10% for $\bar{n}_{SO}^*$. The equations are as follows:

$$\begin{aligned} \bar{n}_S^* &= 2.897 - 5.973\alpha^* + 0.4945K \\ \bar{n}_{SO}^* &= 2.811 - 5.792\alpha^* + 0.5514K \end{aligned} \quad (12)$$

Equations (12) are much simpler than equations (11) in terms of number of predictors. The simpler regression equations provide alternatives when it is applied in the real world given different circumstances. In this paper, the equations in (11) are applied to obtain the thresholds, since they are the most accurate approximations.

4.2.3. Performance of the Regression Model

The regression model's performance can be obtained via simulation; however, it is computationally superior to modify the MDP model by incorporating the regression model's policy as a fixed (yet dynamic in the sense of being state dependent) policy, which reduces the model to a policy evaluation algorithm. An example to illustrate the performance of regression policy is designed in Table 5.

Figure 15 shows the optimal costs of the MVA model, the optimal costs of the MDP model, and the costs of the MDP model under control of the regression threshold policy. The y axis in this figure is the cost and the x axis is the serving rate of assembly plus outfitting, μ_{AO} . The performance of the MDP model under control of the regression threshold policy is always better or equal to the performance of the MVA model. When μ_{AO} is large, where $\alpha^* = 0$, the

K	6
C_{AO}	5
C_{SO}	5
γ	10
μ_S	5
μ_{SO}	3
μ_A	5
μ_{AO}	2.5 ~ 0.1

Table 5. Regression Performance Example

performance of the optimal MDP model, MVA model, and the dynamic model under control of regression threshold policy is the same (or very close). When μ_{AO} decreases, the cost difference between the model and MVA model increases, which is also the circumstance where the regression threshold policy performs better than the MVA model.

Figure 15. MVA, MDP, and Regression Comparison: $K = 6$ (Note that MVA is on Average Better than the Regression Heuristic)

4.3. Design of the Final Heuristic

The half randomly selected test cases from Table 4 shows that the performance of the MVA model is very close to the performance of the MDP model when $\alpha^* = 0$ or $\alpha^* = 1$. When $\alpha^* = 0$, the average absolute percentage optimality gap $|G|$ is 0.053%; When $\alpha^* = 1$, the average absolute percentage optimality gap $|G|$ is 0.0048%. The results also shows that the regression threshold policy greatly improves the performance of the MVA model when $0 < \alpha^* < 1$. Therefore, a final heuristic is designed to improve the efficiency and accuracy, which combines both the MVA model and Regression Threshold policy as in Figure 16.

The algorithm of this final heuristic is:

- (1) Run the MVA model and calculate the value of α^* .
- (2) If $\alpha^* = 1$, process all the outfitting at stage 1, while if $\alpha^* = 0$, process all the outfitting at stage 2.
- (3) If $0 < \alpha^* < 1$, apply Regression Threshold Policy using α^* as an input (as well as $\bar{n}_S$, $\bar{n}_{SO}$, and λ from the MVA model).

We use the absolute percentage optimality gap again as the performance matrix. Let Z^H denote the cost of the dynamic system under control of the heuristic, and λ^* denote the optimal cost from the MDP. The absolute percentage

Figure 16. Final Heuristic Design

optimality gap $|G|$ is defined as follows:

$$|G| = \left| \frac{Z^H - \lambda^*}{\lambda^*} \right| \cdot 100\%. \quad (13)$$

Table 6 presents the absolute percentage error of the performance of the Regression Threshold Policy, the policy from Mean Value Analysis, and the Final Heuristic (which judiciously combines the Regression Threshold Policy and Mean Value Analysis). As expected, α^* has a significant impact on the performance of different policies. Therefore, we present the results according to three ranges for the value of α^* . When $\alpha^* = 0$, the average absolute percentage error of Mean Value Analysis model is only 0.08%; however, for the Regression Policy is a seemingly poor 21.79%. The reasons for the poor performance of the regression threshold go beyond the observation that it is difficult for regression equations to reach a highly accuracy under extreme cases (when $\alpha^* = 0$ or 1). In fact, when $\alpha^* = 0$ or 1, the optimal performance, λ^* , is very close to 0. Therefore, taking the percentage gap yields very high percentage errors even when there is only a small absolute error. When $0 < \alpha^* < 1$, the Regression Threshold Policy improves the performance of MVA by 2.79%. The Final Heuristic which combines Mean Value Analysis and Regression Threshold has the best performance, and it is very close to optimal.

Policy	$\alpha^* = 0$	$0 < \alpha^* < 1$	$\alpha^* = 1$
Mean Value Analysis	0.08%	5.89%	0.006%
Regression Threshold	21.79%	3.11%	15.07%
Final Heuristic	0.08%	3.11%	0.006%

Table 6. Final Heuristic, Regression Threshold, and Mean Value Analysis Comparison in Percentage Error

5. Conclusion and Future Work

In this paper, we developed a flexible two-stage queueing model under a CONWIP job release policy to enhance the planning and control of the outfitting process in shipbuilding. Ship outfitting has the flexibility to be processed at different stages. It is also one of the key components in ship production to determine the ship cost and completion time. The outfitting processes is often delayed due to system variation, capacity limitations, and delays in hull construction. The model presented in this paper is able to account for exponential distributions on processing times and provides more efficient dynamic control policies for the outfitting activities. While it is not our focus to quantify the value in practice of optimized throughput control, our study of control policies and performance gives insight into the large value obtained (i.e., consider the impact of an inflexible system that only allows $\alpha = 0$ or $\alpha = 1$ in Figures 10, 11, 12, and 16).

To analyze the two-stage queueing model, we first formulated an MDP model to capture the essential features of the problem. The numerical results showed that the optimal control policy is a state-dependent threshold type policy. The structure of an optimal policy is one of two types of threshold policies, and the nonlinear switching curves

vary in complex ways on the system parameters/dynamics and costs. Therefore, we developed a static two-stage queueing model to gain more insight into the dynamic system. Instead of dynamic control, we used Bernoulli trials with parameter (α) in this model as a static approximation and used Mean Value Analysis (MVA) to analyze the static model. This model maintains the same system dynamics and cost. The main objective is to find the optimal α^* that minimizes the average cost. Mean Value Analysis can provide the information of the optimal α^* , and also other system metrics, including the average waiting time, the average queue length, and the throughput at each station.

To compare performance between the dynamic and static models, we used the optimal cost from the MDP as a benchmark to measure the performance of the static model. The numerical results showed that the performance of the dynamic model is never worse than the static model. When $\alpha^* = 0$ or $\alpha^* = 1$, the static model performed the same as the dynamic model in most cases and the average optimal percentage optimality gap is smaller than 0.1%. When $0 < \alpha^* < 1$, the dynamic model has better performance than the static model, since the dynamic model can provide more flexible decisions based on the state of the system.

We used the information provided by the static model and also the value of system parameters to search a regression based threshold policy. The regression model has better performance when $0 < \alpha^* < 1$, and the static model can provide more accurate conditions for strict priority rule (when $\alpha^* = 0$ or 1). Therefore, the final heuristic is designed to combine both regression model's policy and Mean Value Analysis.

In addition to generating insights into an optimal policy and generating a very effective heuristic policy, our MVA analysis shows that a static randomized policy may be very effective in some cases, which is a rather surprising result. While our current model is small enough to solve the MDP (unless the WIP level is very large), our approach for developing the final heuristic policy is an interesting approach in which a static model is analyzed using MVA. This static model yields both insights into the optimal dynamic control, and provides a fast algorithm that is used in the final heuristic policy to achieve excellent performance. In applications, the practical considerations often require the use of heuristic policies and methods to speed up computation times.

It remains as future research to introduce the due date of outfitting work to the model and develop a dynamic policy integrated with the information on the due date. Another practical problem in outfitting is the risk of rework after the outfitting work is performed, and that rework may need to be done at the final stage if it is done in the first stage (more generally, an earlier one).

6. Acknowledgements

This work was supported in part by the National Science Foundation under Grant No. CMMI-1233095, and Office of Naval Research (ONR) N00014-08-1-0579.

Appendix A. Relative Value Iteration

The relative value iteration (see [16]) uses a corresponding differential cost vector h^* . The same constant is subtracted from the value function for all states $J_k(n_S, n_{SO})$, so that the difference vector h remains bounded. The differential cost vector, $h_k(n_S, n_{SO})$, is defined as follows:

$$h_k(n_S, n_{SO}) = J_k(n_S, n_{SO}) - J_k(0, 0) \quad \forall \mathbf{k}, \mathbf{x}, \mathbf{s}, \quad (\text{A.1})$$

The state $\{0, 0\}$ is chosen as the fixed state. The relative value function is:

$$\begin{aligned}
h_{k+1}(n_S, n_{SO}) = & \frac{1}{\Lambda} [\mathbb{I}_{\{n_S \geq 1\}} n_S \mu_{AO} (-\gamma + h_k(n_S - 1, n_{SO})) \\
& + \mathbb{I}_{\{n_{SO} \geq 1\}} n_{SO} \mu_A (-\gamma + h_k(n_S, n_{SO} - 1)) \\
& + \mathbb{I}_{\{n_S + n_{SO} < K\}} \min\{\mu_{SO}[C_{SO} + h_k(n_S, n_{SO} + 1)] + (\mu_S - \mu_{SO})h_k(n_S, n_{SO}), \\
& \mu_S[C_{AO} + h_k(n_S + 1, n_{SO})]\} \\
& + (\Lambda - \mathbb{I}_{\{n_S \geq 1\}} n_S \mu_{AO} - \mathbb{I}_{\{n_{SO} \geq 1\}} n_{SO} \mu_A - \mathbb{I}_{\{n_S + n_{SO} < K\}} \mu_S) h_k(n_S, n_{SO}) \\
& - (\min\{\mu_{SO}(C_{SO} + h_k(0, 1)) + (\mu_S - \mu_{SO})h_k(0, 0), \mu_S(C_{AO} + h_k(1, 0))\} \\
& + (\Lambda - \mu_S)h_k(0, 0)].
\end{aligned} \tag{A.2}$$

This method is mathematically equivalent to the value iteration equation in (A.3) and computationally stable for solving the infinite horizon problem. The relative value iteration is guaranteed to converge (within any prespecified accuracy level) to some vector $h_k(n_S, n_{SO})$, with the optimal long run average holding cost calculated as:

$$\lambda^* = \min\{\mu_{SO}(C_{SO} + h_k(0, 1)) + (\mu_S - \mu_{SO})h_k(0, 0), \mu_S(C_{AO} + h_k(1, 0))\} + (\Lambda - \mu_S)h_k(0, 0). \tag{A.3}$$

References

- [1] C. Zimmerman, J. Bissell, Global shipbuilding industrial base benchmarking study part I: Major shipyards, Office of the Deputy Under Secretary of Defense, 2005, <http://www.acq.osd.mil/ip>.
- [2] R. Graves, L. McGinnis, The outfit planning problem: Production planning in shipbuilding, *Naval Research Logistics Quarterly* 29 (2) (1982) 357–384.
- [3] R. Storch, C. Hammon, H. Bunch, Ship production, Cornell Maritime/Tidewater Pub., 1988.
- [4] R. Goldbach, Application of preoutfitting during construction of ammunition ships ae 32-35, *Marine Technology Society Journal* 10 (1).
- [5] Y. Wei, U. Nienhuis, E. Moredo, Two approaches to scheduling outfitting processes in shipbuilding, *Journal of Ship Production and Design* 26 (1) (2010) 20–28.
- [6] F. Dong, J. Deglise-Hawkinson, M. P. Van Oyen, D. J. Singer, Analytical approach to a two-stage queuing network for the planning of outfitting processes in shipbuilding, *Journal of Ship Production and Design* 29 (3) (2013) 136–141.
- [7] W. J. Hopp, M. L. Spearman, *Factory physics*, 3rd ed., Waveland Press, 2011.
- [8] F. Dong, M. P. Van Oyen, D. J. Singer, Dynamic control of the N queueing network with application to shipbuilding, *International Journal of Production Research* (ahead-of-print) (2013) 1–18.
- [9] E. Gel, W. Hopp, M. Van Oyen, Factors affecting opportunity of worksharing as a dynamic line balancing mechanism, *IIE Transactions* 34 (10) (2002) 847–863.
- [10] H. Ahn, I. Duenyas, R. Zhang, Optimal stochastic scheduling of a two-stage tandem queue with parallel servers, *Advances in Applied Probability* 31 (4) (1999) 1095–1117.
- [11] J. Ostolaza, J. Thomas, J. O. McClain, The use of dynamic (state-dependent) assembly-line balancing to improve throughput, *Journal of Manufacturing Operations Management* 3 (1990) 105–133.
- [12] E. S. Gel, W. J. Hopp, M. P. Van Oyen, Hierarchical cross-training in work-in-process-constrained systems, *IIE Transactions* 39 (2) (2007) 125–143.
- [13] H. Parvin, M. P. Van Oyen, D. G. Pandelis, D. P. Williams, J. Lee, Fixed task zone chaining: worker coordination and zone design for inexpensive cross-training in serial conwip lines, *IIE Transactions* 44 (10) (2012) 894–914.
- [14] R. G. Askin, J. Chen, Dynamic task assignment for throughput maximization with worksharing, *European Journal of Operational Research* 168 (3) (2006) 853–869.
- [15] M. Reiser, S. S. Lavenberg, Mean-value analysis of closed multichain queueing networks, *Journal of the ACM (JACM)* 27 (2) (1980) 313–322.
- [16] D. Bertsekas, *Dynamic programming and optimal control*, Vol. 1, 3rd ed., Athena Scientific, Belmont, MA, 2005.