A State-Dependent Riccati Equation Index Policy for Dynamic Production Sequencing in Compounding Pharmacies

Kraig Delana
IE Business School kraig.delana@ie.edu

Christopher Chen
Kelley School of Business, Indiana University cch3@iu.edu

Xiaoshan Peng
Kelley School of Business, Indiana University xp1@iu.edu

When patients require personalized medications or medications that are otherwise unavailable, e.g., due to patients' allergies or drug shortages, retail pharmacies must rely on compounding pharmacies that produce medications to order. Inspired by discussions with the management team of a compounding pharmacy, this project aims to improve timely access to compounded medications through the development of a dynamic production control policy designed to minimize the time between the prescription request and when the medication is ready. We propose and evaluate a novel and theory-driven heuristic policy based on a fluid model of production dynamics paired with a State-Dependent Riccati Equation controller, lightly modified for use as a value function approximation technique, which serves as the basis of an easily implementable index policy. Numerical studies suggest that for large systems this heuristic is within 11% of a lower bound on waiting cost per unit time within a representative fluid setting demonstrating strong theoretical performance. Within a data-driven simulation with more than 1000 unique medications where arrival rates are dynamic and must be estimated, a hybrid policy combining an age-limit with the proposed heuristic can reduce the average cost of waiting time by 32.5% compared to the status-quo oldest request sequencing policy without increasing worst case delays. The proposed heuristic thus contributes a novel and easily implementable approach to production sequencing within compounding pharmacies and a generalizable approach to challenging multi-product production management problems where optimal policies are intractable.

Key words: Compounding Pharmacy, Production Control, State-dependent Riccati Equation
History: December 14, 2025

---

[1]This project has received funding from the EU's Horizon 2023 research and innovation program under the Marie Skłodowska-Curie European Postdoctoral Fellowship Grant agreement No 101150714.

1. Introduction

Approximately 90% of medications are industrially produced and distributed through retail pharmacies (Minghetti et al. 2014). However, when patients require personalized medications, e.g., due to allergies, or medications that are otherwise unavailable due to shortages, retail pharmacies must rely on compounding pharmacies to produce these medications to order. Driven by a combination of scientific advancements, growing demand for personalized medications, and increasingly common drug shortages, the global compounding market is projected to nearly double from $16.9 billion (in 2024) to $31.8 billion over the next decade (Precedence Research 2025). This increase in demand has encouraged new compounding-focused business models. For instance, Apotheek 15 in the Netherlands consolidates the compounding operations of multiple university hospitals to ensure efficient and timely patient access to high-quality compounded medications.

A key operational challenge for compounding pharmacies is ensuring timely access to medications(Masselink et al. 2012, Dobson et al. 2015, Richardson 2019). Compounding involves a series of complex processes that often require advanced training for pharmacists and specialized raw materials, facilities, and equipment. To ensure exact dosages and prevent cross-contamination between batches of medications that differ in form, ingredients, and dosages, time-consuming cleaning procedures are necessary. Moreover, regulations generally mandate a valid prescription prior to compounding (i.e., compounded medications are generally make-to-order), and the combination of short medication shelf-life and wide variety of medications mean that prescription requests must wait in a queue before being compounded. As delays in medication availability are correlated with poorer medical outcomes (340B Health 2023), improving timely access to compounded medications is fundamental to improving health outcomes for patients requiring these specialized medications.

Motivated by discussions with the executive management team of Roseway Labs, a United Kingdom (UK)-based compounding pharmacy, we develop and evaluate a novel dynamic production control policy that aims to minimize delays for patients. Two key aspects to production dynamics in compounding are (1) batch production and (2) sequence-dependent production times for batches. The grouping of identical prescriptions requests into production batches is standard practice in compounding pharmacies as a critical component of quality control, as production is often governed by master formula sheets that

ensure uniformity and adherence to compounding standards (Osteen 2018, p.304). Batching also improves operational efficiency (Dobson et al. 2015) by reducing the frequency of time-consuming set-up and cleaning procedures between different medications. These set-up/switchover times in turn depend on the production sequence. For example, consider a set of orders for Minoxidil 2.0%, Minoxidil 5.0%, Minoxidil 6.0% which can be compounded in sequence by successively increasing the concentration of Minoxidil (the active ingredient). Therefore, compounding the orders in this sequence mitigates the risk of over-dosage and enables shorter switchover times between batches, saving significant amounts of lab and technician resources. However, if not completed in increasing sequence, time consuming cleaning procedures are required to mitigate risks and ensure exact dosage. Beyond medication ingredients, switchover and production times are also affected by the medication form (e.g. capsule, tablet, liquid, or foam), which determines the equipment needed (e.g. capsule fillers, tablet press and oven, mixers, etc.) and packaging. Dynamically exploiting more efficient batch production sequences that minimize lengthy cleaning/equipment switchovers in a stochastic demand environment is integral to reducing production delays for patients, however, doing so optimally is quite challenging.

The production of batches of different types of medications in compounding pharmacies falls within the class of multi-product production management problems in make-to-order settings studied by operations management researchers. In the presence of setup/switchover times, even when sequence independent, the optimal production control for this class is recognized as extremely challenging and often intractable (Lan and Olsen 2006). Therefore, research in this area has focused on developing implementable heuristics with strong performance, such as those proposed by Olsen (1999), Kim and Van Oyen (2000), and Duenyas and Van Oyen (1996). However, existing heuristics do not allow for fully-dynamic sequencing while accounting for the combination of batch production *and* sequence-dependent switchover times, both of which are essential in the compounding pharmacy context. To address this gap, we propose a novel heuristic to sequence medication production that accounts for these complexities.

We begin by modeling the dynamics of batch production with sequence-dependent switchover times using a representative fluid model. From this model, we derive a theoretical lower bound on the time-average cost of requests waiting in the system using techniques similar to Dobson (1992), which provides a benchmark for this setting. We then propose

a heuristic that combines the fluid model dynamics with an optimal control technique known as the State-Dependent Riccati Equation (SDRE) approach. The SDRE approach is a collective term for a family of nonlinear controllers that are widely used in practice for a range of challenging optimal control problems, such as in aeronautics/astronautics, autopilot/guidance systems, and robotics/unmanned vehicles (see reviews by Çimen 2010 and Nekoo 2019). Despite its flexibility and empirically validated success in practice, the SDRE approach has not been widely applied by the operations management community, nor to production control settings.

We propose a modest but novel modification of an infinite-horizon steady-state SDRE controller, also known as the *frozen Riccati equation* controller, as the basis of a dynamic index policy. Originally proposed by Pearson (1962), it approximates a non-linear dynamic system through a rolling linear and static system by applying an infinite horizon Linear Quadratic Regulator (LQR) controller at each decision epoch. Given the solution for the value function of the LQR is known and can be easily computed, we propose it be used as the basis of a forward value function approximation that serves directly as a dynamic index policy. The advantage of this approach is its simplicity in that it only depends on the current state of the system with linearized dynamics, which results in an easily implementable online approach for solving real-time control problems and extends the *frozen Riccati equation* approach to discrete decisions while maintaining methodological simplicity.

We then use the modified SDRE architecture to approximate the original production control problem. Specifically, we use theory to choose the cost function parameterization that proxies the cost of the long-run average time medication requests spend in the system. We then show that this proxy yields a surprisingly simple, state-independent and closed form expression for the value function. This enables the heuristic to easily approximate the complex tradeoffs between online sequencing decisions for any given state of the system and significantly reduces computational complexity, further facilitating real-world implementation.

To evaluate the performance of the proposed SDRE-Index heuristic, we conduct a series of numerical analyses comparing its performance to the lower bound and to the policy of producing a batch of the medication with the oldest request, which we note is the *de-facto* policy in use at a partnering pharmacy. We begin in a fluid (i.e., deterministic) setting,

comparing the performance of the proposed heuristic to a theoretical lower bound on the time-average cost of requests waiting in the system (i.e., the cost rate) across 30 randomly chosen sets of representative parameters. We find that for large systems, in this case more than 124 unique medications, the proposed heuristic is on average within 11% of optimal, while the oldest request policy is on average only within 70% of optimal. These results demonstrate the strong theoretical performance of the proposed heuristic relative to the optimal solution within the fluid setting for larger (more representative) settings.

To evaluate the potential improvement from implementing the SDRE-Index, we extend the analysis to a calibrated stochastic setting, using arrival rates estimated from data provided by the partnering compounding pharmacy, which we assume are known and constant. We focus on a subset of 1,192 "common" medications that account for 83% of all order requests. We find that the proposed SDRE-Index policy reduces the average cost rate by 49.3% compared to the oldest request policy. However, the oldest request policy offers substantially stronger worst-case performance, i.e., the maximum time requests may spend in the system, which offers some insight into why it is used in practice. Therefore, we further propose a hybrid policy that combines the SDRE-Index policy with an age limit, which achieves a 41% reduction in the average cost rate without sacrificing worst-case performance compared to the oldest request policy.

Finally, we perform a data-driven simulation for the same 1,192 medications that introduces the need to estimate dynamic arrival rates from past arrivals. We find that a lightly modified version of the the hybrid-SDRE policy (with an age limit) that corrects for unknown arrival rates by using the ages of requests in the system (which are observable) can reduce the average cost of time requests spend in the system by 32.5%. These numerical results indicate that the proposed heuristic can provide substantial benefits by reducing production delays and improving timely patient access to compounded medications. Moreover, the heuristic is scalable and implementable using free and open source software, requiring only basic information on average arrival rates, changeover/setup times, and production times. This makes the approach not only theoretically sound but also highly practical for real-world application. For compounding pharmacies, the proposed heuristic offers a novel and implementable production sequencing decision support tool.

Methodologically, we contribute to the operations management literature by introducing the SDRE approach to production control problems. More importantly, we propose a

novel use of the SDRE technique as a value function approximation, which in turn provides a simple yet innovative method for extending the SDRE approach to settings with discrete or constrained decision spaces. Thus, the proposed SDRE-Index represents a scalable approximate dynamic programming (ADP) technique (Powell 2016) that is useful in solving complex optimal control problems. Relative to classical ADP techniques, the proposed index eliminates the need to learn the value function approximation or the optimal policy, simplifying implementation. Ultimately, the basis of the heuristic is generalizable to other challenging production and queueing control problems. In other words, by changing the fluid model dynamics and objective function appropriately, the same technique can be reapplied in other control settings where optimal policies are intractable.

The remainder of this paper is structured as follows. §2 provides an overview of the related literature and the contributions of this paper. §3 presents a fluid model which approximates the production dynamics in our setting, the costs of waiting, and the a lower bound on the average cost of waiting per unit time. §4 develops our proposed heuristic. §5 presents a series of numerical experiments to evaluate the theoretical performance and estimate the potential improvement from switching to the SDRE-Index heuristic. We conclude with a discussion in §6.

2. Literature Review

There has been limited examination of production control policies in the context of hospital-based compounding pharmacies, where a common focus has been chemotherapy medications. Within this context researchers have focused on evaluating the trade-off between proactive pre-production of planned treatments to reduce patient delays in-clinic/on-unit, and the chance that the medication is wasted if treatment plans change, e.g., because upon patient arrival/assessment physicians revise dosages, or patients are rescheduled/no-show and medications subsequently expire (Masselink et al. 2012). Both Dobson et al. (2015) and Richardson (2019) apply integer programming approaches to balance on-unit delays and wastage, where the former focuses on creating a cyclical batch production schedule for an individual medication, and the latter focus on constructing production schedules for planned treatments across multiple medications and when to pre-produce. Similarly, we also focus on reducing patient medication delays. However, instead

of focusing on delays occurring in-clinic/on-unit that can be reduced via proactive pre-production but create a risk of wastage, we focus on production policies to minimize production delays in the pharmacy. Specifically, we consider delays as the time between when a prescription request is made and when the medication is ready. Moreover, in contrast to planning for known future treatment needs, we consider a stochastic and dynamic setting for a portfolio of medications when future demand is uncertain.

The dynamic production control problem for compounding pharmacies lies within the literature on control of polling queues and particularly the applied work on production scheduling for multi-product make-to-order systems with switchover times and batch production/service. To this literature, we contribute a novel and fully dynamic heuristic that addresses both sequence-dependent setup times and batch production, the approach for which is generalizable and builds upon the research on the SDRE approach within optimal control. Specifically, we introduce a modest but novel modification of a simple SDRE controller, adapting it for use as a value function approximation technique. This represents a novel application of the SDRE approach as a generalizable ADP technique that significantly simplifies the process of developing the approximation/policy.

2.1. Polling Queues and Production Scheduling.

There is extensive literature on polling queues (see Takagi (1997), Boon et al. (2011), Borst and Boxma (2018) for comprehensive surveys). When there are no switchover times for moving between queues, the $c\mu$ is either optimal or asymptotically optimal, and recently Sun and Zhu (2025) extended this to a robust optimal control setting. However, in the presence of switchover times, even those that are sequence-independent, finding optimal policies is notoriously difficult. For instance, Reiman and Wein (1998) derive the optimal dynamic scheduling problem for the case of two queues in heavy traffic, and Hu et al. (2022) examines systems with fixed visitation sequences (polling tables), showing that a binomial-exhaustive policy is asymptotically optimal under large changeover times. For polling models with batch service, the most closely related work is van der Wal and Yechiali (2003), which considers optimal (*semi-*)dynamic control of polling queues with batch service in the presence of sequence-independent switch-in and switch-out times. They restrict their search to fair (*Hamiltonian*) policies where each queue is only visited once within a cycle (where cycles are determined dynamically). For the case that is most closely related

to our work (that with locally gated service), they show that constructing such a fair policy in a stochastic setting is NP-hard.

As a result of the analytical intractability of optimal control policies in the presence of set-up times, the operations community has focused on the development of implementable dynamic heuristics. For example, Duenyas and Van Oyen (1996) and Olsen (1999) develop heuristics for stochastic settings with fully dynamic control where switchover times are sequence independent, with the former focusing on average waiting times while the latter also considers tail-performance. Lan and Olsen (2006) and Niño-Mora (2009) further incorporate setup costs (in addition to setup times) with Lan and Olsen (2006) using a fluid model to develop bounds and heuristics while Niño-Mora (2009) study the problem within the context of a multi-armed restless bandit framework to derive a dynamic index policy.

Incorporating sequence-dependent switchover times adds another layer of complexity, with the literature concentrating on static sequences or cyclic policies. Bertsimas and Xu (1993) develop a convex optimization model that provides a lower-bound for fixed (static) visit sequences, then propose an integer programming formulation to construct static heuristic policies that are close to optimal. For a make-to-stock system, Lan (2000) similarly derive a convex optimization problem to derive a lower bound under static policies. In the case of the economic lot scheduling problem (when inventory can be produced ahead of time and held), Dobson (1992) studies cyclic policies using a Lagrangian relaxation to provide a lower-bound problem and a related heuristic. Adelman and Barz (2014) study the same problem by formulating the problem as a semi-discrete Markov decision process while also using an ADP approach. However, they focus on deriving lower bounds rather than dynamic heuristics. Their use of the ADP approach is also different in that they relax the constraints of the associated infinite-dimensional linear program of the dynamic programming problem. We show that in our problem, where inventory cannot be held (as in Bertsimas and Xu (1993)) with sequence-dependent switchover times and batch production, that a static cyclic policy can be used to derive a lower bound within a fluid setting. However, our work seeks a fully dynamic heuristic that responds to the state of the system.

2.2. The SDRE approach

The heuristic production control policy that we propose to handle both sequence dependent set-up times and batch service is based on the SDRE literature. Specifically, we build on the

discrete-time infinite-horizon steady-state controller, sometimes referred to as the *frozen Riccati equation* approach (Pearson 1962, Çimen 2010). Extensive research has gone on to further develop the SDRE approach by exploring ways to optimize and generalize this approach. E.g., Dutka et al. (2005) relax the linear-static approximation by projecting the system's trajectory into the future, an approach referred to as model predictive control. This technique improves performance of the *frozen Riccati equation* approach and enables constraints on controls or states, e.g., (Bemporad et al. 2002, Chang and Bentsman 2013), albeit at the cost of increased complexity.

Our proposed approach sidesteps the need for model predictive control to address the discrete (constrained) production sequencing decisions that preserves the simplicity of the frozen Riccati equation approach. We achieve this through a simple but novel insight: the solution of the frozen SDRE can be used as a value function approximation directly. The basis for this proposal is derived directly from analysis of the LQR problem, in which the solution to the Riccati equation enables one to evaluate the value function for any given state (we note this is a known result that we exploit as opposed to derive), and has also been indirectly noted within the context of the SDRE in Alla et al. (2023). Thus, enumerating the value function under a set of alternative discrete decisions serves as the basis of a dynamic index policy. This contributes a simple way to extend the frozen SDRE approach to systems with constrained control (decision) spaces.

The proposed SDRE-Index heuristic can be framed within the broader ADP literature, for which an excellent summary is provided by Powell (2016). Specifically, the heuristic represents a novel and generalizable value function approximation technique. Within value function approximation techniques, the heuristic also significantly simplifies the process of developing the approximation/policy. The three classical steps of designing policies based on value function approximations are outlined in Powell (2016) as "`(1) designing a value function approximation, (2) updating the value function approximation for a fixed policy, and (3) learning value function approximations while simultaneously searching for a good policy.`" As we further detail in 4.3 when constructing the heuristic, and elaborate on in Remark 2, the choice of the SDRE architecture with theory-driven cost parametrization obviates the need for learning the value function, thus the approximation can be used directly as an index, which in turn removes the need to search for a good policy. Given the broad flexibility

of the SDRE approach to model complex systems, further exploration of the SDRE as a value function approximation technique may offer promising research opportunities for theory driven heuristic development in cases where optimal policies are intractable.

3. The Fluid Model

In this section, we first introduce a fluid model to capture the production dynamics of a single server that produces batches of medications to order by assuming that the demand of each medication arrives at a constant rate. We then define the cost of waiting for medication requests and derive a lower bound as a benchmark for the heuristic policy developed in the following section.

We note that while the fluid assumption ignores the stochasticity it captures the key features of production dynamics and facilitates the derivation of SDRE heuristic in Section 4. In sections 5.2 and 5.3 we present numerical experiments in stochastic settings and introduce slight modifications to handle both stochasticity and uncertainty of arrival parameters respectively.

3.1. Fluid Production Dynamics

We consider a production scheduling problem with n ($n \geq 2$) medications in a deterministic (i.e., fluid) setting. The demand rate of medication i (for $i = 1, \ldots, n$) is denoted by λ_i. For notational convenience, we combine the switchover time from medication i to j, and batch production times for medication j (with mean μ_j^{-1}), into a single parameter denoted as S_{ij}. That is, S_{ij} captures the time required to clean equipment, set up or switchover from medication i to j, and produce a batch of medication j. We note that as switchover times are a function of the differences in medication characteristics (ingredients, dosages, and form), if batches of the same medication are produced consecutively this minimizes switchover time, i.e., $S_{j,j} = \min_i S_{i,j}$.

We formulate the problem within a degenerate (non-stochastic) semi-Markov decision process framework. Let the superscript $t = 1, 2, \ldots$ index the decision epochs, where at each decision epoch the system manager can take action $a^t \in \{1, ..., n\}$ representing the medication to produce next. At each time t, the decision maker observes the queue length of each medication i denoted as q_i^t, representing the number of unfilled orders of medication i. The state of the system at each decision epoch can be described by $(q_1^t, \cdots, q_n^t, a^{t-1})$, where a^{t-1} captures the index of the medication most recently produced.[1]

[1] As we focus on minimizing the long-run average cost, the initial state a^0 does not affect the optimal policy when the policy induces a single recurrent Markov chain.

Given the current state of the system and an action a^t, the time until the next decision epoch $(t+1)$, is S_{a^{t-1},a^t}. During this time, demand of medication i arrives at a rate of λ_i, and a batch of size $q^t_{a^t}$ is produced for medication a^t. The state transitions are as follows:

$$\begin{aligned}&\left(q_1^{t+1},...,q_{a^t}^{t+1},...,q_n^{t+1},a^t\right) \mid \left(q_1^t,...,q_{a^t}^t,...,q_n^t,a^{t-1},a^t\right) \\ &= \left(q_1^t+\lambda_1 S_{a^{t-1},a^t},...,\lambda_{a^t} S_{a^{t-1},a^t},...,q_n^t+\lambda_n S_{a^{t-1},a^t},a^t\right).\end{aligned} \tag{1}$$

Intuitively, this captures that a batch of the medication (a^t) being produced fulfills all existing orders (for that medication) in the system when the batch is started, and the new demand arrives to each queue during the production time. Figure 1 provides an illustration of the queue dynamics over time for the case of 2 medications where production rotates between the two medications and we depict the queue length processes between decision epochs.

Figure 1 Fluid Queue Length Processes Illustration - 2 Medications

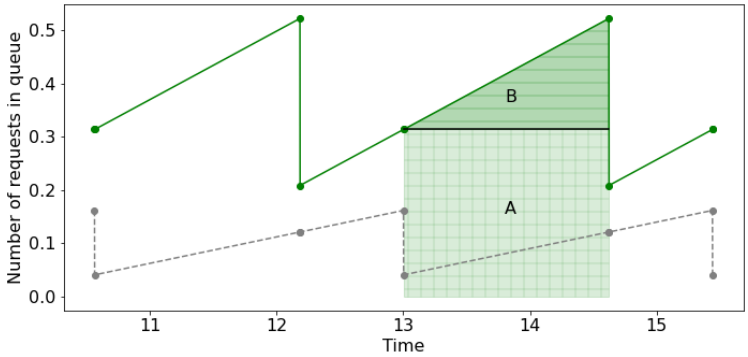


Figure 1 illustrates the queue length processes over time for two medications in a fluid setting. The the shaded regions illustrate the total waiting time for all requests in a single queue between 2 decision epochs.

The cost associated with the time spent in the system for each request accrues at rate w_i per unit time. Therefore, the total cost incurred between decision epochs t and $t+1$ when taking action a^t given $(q_1^t,\cdots,q_n^t,a^{t-1})$ can be expressed as

$$c(q_1^t,\cdots,q_n^t,a^{t-1},a^t)=\sum_{i=1}^{n} w_i\left(q_i^t S_{a^{t-1},a^t}+\frac{\lambda_i}{2}(S_{a^{t-1},a^t})^2\right), \tag{2}$$

The first term inside the parenthesis of (2) captures the waiting time of requests already in the system at decision epoch t, and the second term captures the waiting time of new arrivals between time t and $t+S_{a^{t-1},a^t}$. To make this clear, Figure 1 illustrates these costs

for a single queue in a single decision period – the shaded area labeled A (with both horizontal and vertical hatching) represents the first term, and the shaded area labeled B (with horizontal hatching) represents the second term. Thus, the cost incurred (across all queues) within each decision epoch can also be interpreted as the weighted sum of the integrals of each queue length process, with weights w_i for each queue.

A feasible policy $\pi : (q_1^t, \cdots, q_n^t, a^{t-1}) \to a^t$ is defined as a function that maps the state of the system to the scheduling action. Let Π denote the set of all feasible policies. Our objective is to find the optimal dynamic scheduling policy π that minimizes the long-run (time-average) cost of the time requests spend in the system, given some initial medication produced a^0. Specifically, we aim to minimize:

$$V^\pi = \lim_{t\to\infty} \frac{\sum_{s=1}^{t} c(q_1^s, \cdots, q_n^s, a^{s-1}, a^s)}{\sum_{s=0}^{t} S_{a^{s-1},a^s}}. \tag{3}$$

The numerator of equation (3) is the total holding cost incurred in the first t production cycles, while the denominator is the total production time of the first t production cycles. Thus, the objective value V^π computes the long-run average cost rate under a given policy π.

We denote the optimal value as $V^* = \inf_{\pi\in\Pi} V^\pi$. Note that V^* is always finite because a simple cyclical policy that produces the medications in sequence, i.e., in the order of its index, gives a finite value in equation (3). Given the limited results on optimal dynamic control of polling queues, it is perhaps unsurprising that the problem of minimizing (3) subject to (1) does not appear analytically tractable. Therefore, before developing a heuristic policy for this setting, we first seek a lower bound on this problem to facilitate evaluation and comparison of policies relative to optimal.

3.2. Lower bound

This section presents the derivation of a lower bound on (3) subject to (1), that will serve as a benchmark for evaluating the performance of heuristic policies relative to optimal.

We begin by restricting our attention to cyclical policies, which we prove to be optimal in Lemma 1. A cyclical policy can be represented by a m-dimensional vector $r = (r_1, \cdots, r_m)$, denoting a sequence of production, where $r_j \in \{1, \ldots, n\}$ indexes the medication produced on the j-th production run in a cycle and m is the total number of production runs per cycle. In addition, we define $r_{m+1} = r_1$ so that the policy is cyclical.

To compute the long-run average cost of a given cyclical policy, we first let T_c denote the total time of running a production cycle, which can be computed as follows:

$$T_c = \sum_{j=1}^{m} S_{r_j, r_{j+1}} \tag{4}$$

Let L_j (for $j = 1, \ldots, m$) denote the set of positions in the sequence starting at position $j+1$ and continuing up to, and including, the position where medication r_j is made again. Also, let J_i denote the set of positions in the sequence where medication i is produced. Thus, the long-run average cost of the cyclical policy r is

$$h(r) = \frac{1}{T_c} \sum_{i=1}^{n} \frac{w_i}{2\lambda_i} \sum_{j \in J_i} (D_j + 2\lambda_j S_{r_j, r_{j+1}}) D_j \tag{5}$$

where

$$D_j = \lambda_j \sum_{k \in L_j} S_{r_k, r_{k+1}}, j = 1, \ldots, m; \tag{6}$$

In equation (6), D_j represents the inventory level at the beginning of next time when medication r_j is served. Thus, the objective function (5) computes the total holding cost over one whole cycle, which is then divided by the total cycle time, yielding the average inventory cost rate in one cycle. Next, Lemma 1 states that there always exists a cyclical policy that is optimal. Thus, we can restrict our attention to the cyclical policies to derive lower bounds.

LEMMA 1. *There exists a cyclical policy r^* such that $h(r^*)$ achieves the same cost as the optimal long-run average cost defined in equation (3).*

It follows from Lemma 1 that the MDP problem (3) is equivalent to the following optimization problem:

$$\inf_r h(r) \quad \text{s.t.} \quad (4), (6).$$

In what follows in this subsection, we want to find the a lower bound of the long-run average cost under optimal cyclical polices of this optimization problem.

To this end, we consider a similar approach in Dobson (1992) and derive a lower bound of the long-run average cost rate if the frequencies of production for each medication in one cycle is given. Let $f = (f_1, \cdots, f_n)$ denote the frequency of production (the number of runs per cycle) of each medication, where $f_i \in \mathbb{N}_+$ with $m = \sum_{i=1}^{n} f_i$ being the total number of

production runs per cycle. The production sequence vector r needs to be consistent with the frequency vector f, i.e., for all i,

$$\sum_{j=1}^{m} \mathbb{I}_{\{r_j=i\}} = f_i,$$

where I_A is the indicator function such that $I_A = 1$ if A is true and $I_A = 0$ otherwise. We first derive a lower bound of the value function for a given frequency vector f. Then, we optimize the lower bound function of the frequency f and derive the lower bound of the orignal problem.

We only focus on frequency vectors such that $f_i \geq 1$ for all medications, so that all orders will be served after a certain period of time. Otherwise, the value of equation (3) goes to infinity, dominated by a simple cyclical policy that serves the medications in the order of $(1, \cdots, n)$.

To derive a lower bound, we relax the constraint eq. (6) by summing up all positions related to the same medication and replacing it with the following constraint:

$$\sum_{k \in J_i} D_k = \lambda_i T_c, \ \ i = 1, \ldots, n. \tag{7}$$

If we fix the value of T_c and ignore the term $2\lambda_j S_{r_j, r_{j+1}}$, optimizing the function $h(r)$ subjected to the relaxed constraint (7) is equivalent to optimizing the value of D_k. The optimal solution is $\bar{D}_k = \lambda_i T_c / f_i$ by the first-order condition, which we note has a simple intuitive meaning that production runs for each medication should be evenly spaced within the sequence. Thus, by substituting the values of $\bar{D}_k$ in the objective function, we derive a lower bound function $\bar{h}(f, T_c)$ of the objective function $h(r)$ as follows:

$$\bar{h}(f, T_c) = T_c \sum_{i=1}^{n} \frac{w_i \lambda_i}{2 f_i} \tag{8}$$

Since the function $\bar{h}$ increases in the total cycle time T_c, one of its lower bound is given by $\bar{h}(f, g(f))$, where $g(f)$ denotes the minimal time to produce all medications for the given frequency vector f in one cycle. Then given that $S_{j,j} = \min_i S_{i,j}$ (because making two batches of the same medication has the minimal set-up time), for any frequency vector f,

$$g(f) \geq \sum_{i=1}^{n} S_{i,i} f_i.$$

Thus, the following theorem provides a lower bound of the long-run average cost by replacing $g(f)$ with its lower bound, we have that

$$h(r) \geq \bar{h}(f, g(f)) \geq \tilde{h}(f) = \left(\sum_{i=1}^{n} S_{i,i} f_i\right)\left(\sum_{i=1}^{n} \frac{w_i \lambda_i}{2 f_i}\right).$$

Therefore by minimizing $\tilde{h}(f)$ over the vector f of production frequencies for each medication we have the following lower bound on V^*.

THEOREM 1. *For any cyclical production sequence with production frequencies given by the vector f:*

$$V^* \geq \min_f \tilde{h}(f). \tag{9}$$

Theorem 1 shows that a lower bound of the long-run average cost can be computed by minimizing $\tilde{h}(f)$ over the space of production frequencies for each medication. However, $\tilde{h}(f)$, as the product of an affine and a convex functions of f is not guaranteed to be jointly convex in the production frequencies, meaning numerical optimization (e.g., via gradient descent) may not find a global minimum over f. A simple solution for this follows from observing $\tilde{h}(f) = \tilde{h}(\alpha f)$ for $\alpha > 0$, i.e., it is scale invariant. Adding a normalization constraint $(\sum_{i=1}^{n} S_{i,i} f_i) = C$ for some positive constant C, then enforces a fixed scale for the relative frequencies, and more importantly simplifies the objective function to $C\left(\sum_{i=1}^{n} \frac{w_i \lambda_i}{2 f_i}\right)$ yielding a jointly convex form in the production frequencies (as the sum of convex functions). Thus, $\min_f \tilde{h}(f)$ subject to $(\sum_{i=1}^{n} S_{i,i} f_i) = C$ yields an easily computable lower bound on V^*.

4. SDRE-Index Heuristic Policy

The core idea of our the proposed heuristic is a simple yet novel adaptation of the SDRE approach, repurposing it as a theory-driven value function approximation technique. This framework enables us to approximate the original problem of minimizing (3) subject to (1) within a tractable modified SDRE architecture. To facilitate this, we first rewrite the production control problem introduced in §3.1 with linear algebraic notation, and then begin construction of the heuristic from the Linear Quadratic Regulator (LQR) problem, which the SDRE approach generalizes to systems with non-linear dynamics.

Let $\lambda \in \mathbb{R}^n$ denote a vector of arrival rates and let $S \in \mathbb{R}^{n \times n}$ denote the total switchover and production times when switching from medication i to j, i.e. S_{ij} is the sum of the

switchover time from medication i to j and the production time for a batch of medication j. Further we define $S^2 \in \mathbb{R}^{n\times n}$ such that $S^2_{ij} = (S_{ij})^2$. The state of the system at the decision epoch times indexed by the subscript $t \in \{1, 2, 3, ...\}$ is defined as $x_t \in \mathbb{R}^{2n}$ where the first n terms capture the length of each request queue and the second n terms capture the server location (expressed a vector i_m with a one in the m^{th} position if the server is at queue m and zeros in all others). Given a current state x_t and an action indicating which queue to serve next, denoted $a_t \in \mathbb{R}^n$ with a one in the m^{th} position if the server chooses to serve the queue indexed by m and zeros in all others, then the state at the next decision epoch x_{t+1} can be expressed as:

$$x_{t+1} = A\, x_t + B(x_t)\, a_t, \tag{10}$$

$$\text{where } A = \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix},\ B(x_t) = \begin{bmatrix} \lambda \otimes x_t^T \tilde{V} S - \text{diag}(x_t^T \tilde{Q}) \\ I \end{bmatrix},$$

$$\text{and } \tilde{Q} = \begin{bmatrix} I \\ 0 \end{bmatrix},\ \tilde{V} = \begin{bmatrix} 0 \\ I \end{bmatrix}.\ ^{2}$$

Letting $w \in \mathbb{R}^n_{++}$ denote the cost of waiting one unit of time for each type of medication. The cost of taking the action a_t given x_t can be written as $c(x_t)a_t$ where,

$$c(x_t) = \left[x_t^T \tilde{V} S_\mu\right] \left(x_t^T \tilde{Q} w\right) + \left[x_t^T \tilde{V} S^2\right] \left(\frac{\lambda^T w}{2}\right). \tag{11}$$

We note here that $c(x_t)$ is a row vector, where parallel to (2) the first term captures the sum of the waiting cost for all tasks already in the queue during the time between decision epochs, and the second captures the waiting costs for tasks that will arrive between decision epochs. We seek a policy $\Pi(x_t)$ that minimizes the time average of this measure, i.e., a policy which minimizes the time average cost rate of waiting measured as time-average weighted sum of the integrals for each queue length process. Formally, we seek to minimize the long-run average cost rate, i.e.,

$$\lim_{t\to\infty} \frac{\sum_{s=0}^{t} \left(c(x_s)a_s\right)}{\sum_{s=0}^{t} \left(x_s^T \tilde{V}\, S\, a_s\right)}. \tag{12}$$

[2]We note that we use $\otimes$ to denote the outer product strictly for clarity, and that $\lambda \otimes x_t^T \tilde{V} S = \lambda x_t^T \tilde{V} S$ as λ is a column vector and $x_t^T \tilde{V} S$ is a row vector.

4.1. Linear Quadratic Regulator

Here we present a brief overview of the LQR problem to aid in development of intuition, elucidate the key ideas we exploit, and to keep the paper self-contained. We take advantage of existing results without proof, providing references for the interested reader. The infinite horizon discrete-time LQR problem is shown in (13), whose name follows from the formulation where the dynamics are linear in state space x_t and controls a_t while the objective is quadratic, meaning Q and R are positive semi-definite.

$$\min_{a_1,a_2,\dots} \sum_{t=1}^{\infty} x_t^T Q\, x_t + a_t^T R\, a_t \tag{13}$$
$$x_{t+1} = A\, x_t + B\, a_t$$

When reformulated as a dynamic programming problem the Bellman equation becomes,

$$V(x_t) = x_t^T Q\, x_t + \min_{a_t}(a_t^T R\, a_t + V(A\, x_t + B\, a_t)). \tag{14}$$

Analysis of the LQR problem shows that the solution to (14) can be expressed as

$$V(x) = x^T P\, x, \tag{15}$$

where P is the solution to,

$$P = Q + A^T \left(P - PB(R + B^T PB)^{-1} B^T P\right) A. \tag{16}$$

Equation (16) is known as the discrete-time algebraic Riccati equation (DARE) (see Boyd 2008, Tedrake (2024)[Chapter 8.]), for which efficient off-the-shelf solvers are available. The solution, P, is used to compute the optimal control a_t via the *gain matrix* K as shown in (17).

$$a_t = Kx_t, \text{ where } K = -(R + B^T PB)^{-1} B^T PA. \tag{17}$$

We highlight here that a core insight from the LQR problem that we exploit is that P also yields the ability to evaluate the value function (15) for any given state x.

We also note here that a necessary condition for (16) to have a solution is that (A, B) is stabilizable (Tedrake 2024)[Chapter 8.]. After having introduced the proposed heuristic, we will remark on what this means within the context of the fluid model for our setting, and for which we will ultimately demonstrate existence and uniqueness by construction.

4.2. The State-Dependent Riccati Equation (SDRE) Approach

The SDRE approach is a collective term for a range of controllers that generalize the LQR problem to the case with state dependent parameters that capture non-linear dynamics as shown in (18).

$$\min_{a_1,a_2,\ldots} \sum_{t=1}^{\infty} x_t^T \mathbf{Q}(\mathbf{x_t})\, x_t + a_t^T \mathbf{R}(\mathbf{x_t})\, a_t \tag{18}$$
$$x_{t+1} = \mathbf{A}(\mathbf{x_t})\, x_t + \mathbf{B}(\mathbf{x_t})\, a_t.$$

While there are many SDRE approach based controllers, which, we note, are generally known to be sub-optimal (Çimen 2010), we will focus on a simple one, namely a rolling infinite-horizon steady-state controller originally proposed in Pearson (1962) and referred to as the *frozen Riccati equation* controller. We note that infinite-horizon steady-state formulations for SDRE controllers are used in various applications due to their advantage of simplicity (Çimen 2010, Mathavaraj et al. 2021). The *frozen Riccati equation*, given by (19) effectively applies a rolling linear and static approximation of a non-linear dynamic system, where at each decision epoch the infinite-horizon steady-state $P(x_t)$ is computed using the state-dependent parameters. In effect, it solves a rolling LQR Ricatti equation, where the dynamics (parameters) are updated at each step and offers a simple benchmark to evaluate extensions, e.g., as in Dutka et al. (2005) who relax the static approximation by projecting the system trajectory forward in time.

$$P(x_t) = Q(x_t) + A(x_t)^T \left(P(x_t) - P(x_t)B(x_t)(R(x_t) + B(x_t)^T P(x_t)B(x_t))^{-1} B(x_t)^T P(x_t)\right) A(x_t). \tag{19}$$

Then, following the LQR controller (17), the *frozen Riccati equation* controller a_t is computed via a state-dependent *gain matrix* $K(x_t)$, specifically,

$$a_t = K(x_t)x_t, \tag{20}$$
$$\text{where } K(x_t) = -(R(x_t) + B(x_t)^T P(x_t)B(x_t))^{-1} B(x_t)^T P(x_t)A(x_t).$$

This control is then re-computed and applied at each decision epoch as x_t evolves, and generalizes the LQR to the setting with state-dependent parameters.

4.3. SDRE Dynamic Index Heuristic

The key insight behind the heuristic proposed in this paper follows from connecting (i) the evidence from the literature that the *frozen Riccati equation* controller (20) works reasonably well in a wide variety of applications, with (ii) the result that the value function in the the LQR problem can be expressed as $x_t^T P x_t$ (i.e., (15)). This connection suggests that the solution of the *frozen Riccati equation* at each decision epoch, which then inserted into (15) should function as an approximation of the value function when the parameters are state dependent. In this way, we can interpret the proposed heuristic through the lens of approximate dynamic programming (ADP), and specifically as a value function approximation (Powell 2016).

By using the solution of the *frozen Riccati equation* as a value function approximation, accommodating the discrete decision space of which queue to serve next, as well as the policy, becomes effectively trivial. We propose that given a state x_t simply enumerating the value function approximation under each possible decision a_t to generate a dynamic index, and a policy of choosing the lowest index. Specifically, for each a_t in the decision space, letting $x_{t+1} = A\,x_t + B(x_t)\,a_t$, updating $B(x_{t+1})$ accordingly, then solving (19) for $P(x_{t+1})$, the dynamic index for each a_t can be computed as

$$x_{t+1}^T P(x_{t+1})\,x_{t+1}, \text{ where } x_{t+1} = A\,x_t + B(x_t)\,a_t. \tag{21}$$

The above heuristic index directly extends the SDRE architecture for control of non-linear dynamic systems to discrete decisions (e.g. production sequencing). Moreover, it does so without adding significant complexity, e.g, without the added complexity of model predictive control based approaches.

Remark 1. For the solution of the *frozen Riccati equation* (19) to be used as an approximation of the value function for the set of potential states x_{t+1} implicitly assumes that the dynamics remain static (do not change from time $t+1$) and in subsequent decision epochs $(t+2, ...)$ that the unconstrained optimal LQR control is applied. Thus, the conditions under which such a $P(x_{t+1})$ exists are the same as for the LQR given the corresponding state-dependent dynamics and cost function, i.e., $(A, B(x_t))$ must be stabilizable for all x_t. Intuitively, this means that requests must arrive to the system slower than they can depart when the batch of a given medication is produced. More specifically, following from (10) and denoting $\tilde{B}(x_t) = \lambda \otimes x_t^T \tilde{V} S - \text{diag}(x_t^T \tilde{Q})$, this means that the diagonal terms of $\tilde{B}(x_t)$

must be negative (so the number of requests waiting can be driven towards zero when a queue is served).

We next derive a quadratic approximation for the average cost of waiting (12) that drives the value function approximation, and subsequent sequencing decisions, to minimize the cost of time that requests spend in the system. Specifically, following from the interpretation of (12) as the time average weighted sum of the integrals of each queue length process, we choose $Q(x_t)$ and $R(x_t)$ to capture this sum of integrals.

In a fluid model, the area under the queue length curve for all requests currently in the system is quadratic in the queue lengths and can be interpreted as the cumulative age of all requests in the system. For instance, if there are x requests of type i in the system in the system at time t denoted as (x_t^i), then the oldest request of type i has been in the system for (x_t^i/λ_i) units of time,[3] and the cumulative time all requests of type i have spent in the system is $x_t^i \frac{x_t^i/\lambda_i}{2}$. Thus, the cumulative weighted age for all requests in the system at a given decision epoch t can be expressed without state-dependent coefficients as

$$x_t^T Q x_t, \text{ where } Q = \begin{bmatrix} \text{diag}(w/2\lambda) & 0 \\ 0 & 0 \end{bmatrix}. \tag{22}$$

With the above intuition, letting $R(x_t) = 0$ implies the objective (22) should drive the SDRE index policy to minimize the future cumulative area under the queue length curves for all requests in the system and offers a theory-driven quadratic approximation for (12).

Remark 2. We remark that the choice of the matrices Q and R to reflect the long run average cost per unit time (12) can be thought of as part of the process of "designing a value function approximation" as discussed in (Powell 2016). In the ADP literature this generally then requires learning the value function approximation through a chosen statistical learning method. However, in the proposed heuristic the SDRE architecture provides the functional form, and the parameterization of Q and R are driven by theory and problem structure (capturing waiting times) as opposed to learned. A significant advantage of this relative to traditional ADP approaches is that given a problem specific choice of Q and R, there is no need to learn the value function or subsequently search for a policy, making this approach significantly simpler and easier to implement. That said, it is important to note that the choice of Q and R is critical to performance as they determine the associated costs to be minimized.

[3] Where x_t^i/λ_i is the time it would take for the fluid arrivals to sum to x_t^i.

One potential computational challenge in implementing the proposed heuristic is that the index policy as shown in (21) requires solving *n frozen Riccati equations* (one for each a_t). This is computationally cumbersome for a large portfolio of medications. However, following from $R = 0$ and the specific structure of our problem, in which costs are only derived from a subset of the state variables (queue lengths for each medication type), Proposition 1 shows that (19) has a unique closed form solution.

PROPOSITION 1. *Given A and $B(x_t)$ as defined in (10) and Q as defined in (22), with $R = 0$, the unique solution to the state-dependent Riccati equation (19) is $P(x_t) = Q$.*

Proposition (1) shows that despite the state dependence of $B(x_t)$, $P(x_t)$ is not state dependent in this setting. Intuitively, we note that a necessary condition this result is reliant on is a the exclusive focus on the cost of the state of the system captured by Q in evaluating the impact on future costs, where $R = 0$ means one does not need to consider the costs associated with a specific control decision at time $t + s$ in the future. This is particularly useful computationally, given that $P(x_t) = Q$, there is no need to solve the state-dependent Ricatti equation, which drastically reduces the computational complexity. This enables the heuristic to not only to be easily implementable, but scalable to large systems as the indices for each a_t in (21) reduce to,

$$x_{t+1}^T Q\, x_{t+1}, \text{ where } x_{t+1} = A\, x_t + B(x_t) a_t. \tag{23}$$

For the index of the medication most recently produced (and only this index), we make a small modification. Specifically, we add a penalty term equal to the maximum index across all medications to discourage consecutive batches. This modification is motivated by an intermediate result within the derivation of the lower bound, which indicates batches of a given medication should be roughly evenly spaced over time (i.e., not produced consecutively).

The proposed SDRE-index policy contributes a novel and theory-driven implementable heuristic to the operations research / operations management literature on multi-product production systems for a system with sequence-dependent set-ups and batch processing. Moreover, it makes a modest yet novel contribution to the literature on the SDRE approach, not only extending its use to a new area of application but providing a simple and direct modification allowing for discrete controls. Perhaps more importantly, the

ideas behind this heuristic connect the SDRE approach to the approximate dynamic programming literature as a novel value function approximation technique that is flexible and generalizable to a wide variety of problems that can be modelled with piecewise linear dynamics and quadratic objectives.

5. Performance

This section presents a series of numerical experiments designed to evaluate the performance of the proposed heuristic, denoted the *SDRE-Index* policy. Performance is primarily assessed relative to a practice-based heuristic in use at the partnering pharmacy, denoted as the *oldest* policy, where the request that has been waiting the longest determines the next medication produced.[4] The first experiment is designed to assess theoretical performance and compares both heuristics within a fluid setting relative to the lower bound on the optimal cost developed in §3.2. The second and third experiments assess the potential benefit of switching from the current policy to the proposed heuristic in stochastic settings. Specifically, in the second experiment we assume arrivals occur following Poisson processes with known and constant rates, and in the third we consider a data-driven simulation using historical orders that requires estimating dynamic arrival rates from the past orders.[5]

5.1. Performance in a Fluid Setting

To evaluate the theoretical performance of the heuristic policies relative to the lower bound on the best possible performance developed in §3.2, we begin the numerical experiments within a fluid/deterministic setting.

We vary the number of distinct medications across $n \in \{4, 12, 24, 36, 72, 124\}$, each arriving to their own queue, where larger system sizes have greater scope (produce a wider array of medications) and are more representative of reality. We note that in an effort to provide a general assessment of the theoretical performance of both heuristics, for each n we simulate 30 replications with randomly drawn parameters for λ and S that are representative of a compounding pharmacy setting. This approach ensures the generality of

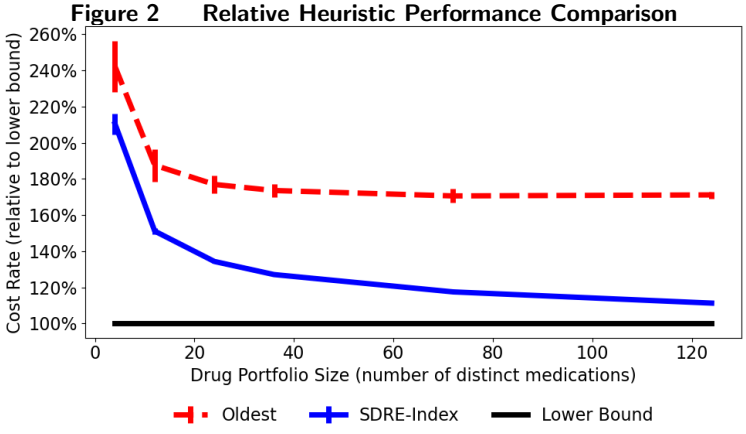
[4] While the oldest policy is naive in that it does not take advantage of any of the problem structure, it is extremely easy to implement in reality and does not require any data. Moreover, we note here that the simulation results demonstrate that this policy delivers strong performance in a stochastic setting with respect to the maximum waiting times for requests, offering insight into why it is used in practice. However, in the appendices for the numerical experiments we provide supplementary results comparing the proposed heuristic to an alternative heuristic (proposed in Appendix C) that exploits problem structure and demonstrate our proposed heuristic continues to offer substantial improvement over less naive policies.

[5] The data and code necessary to replicate all numerical experiments is publicly available at https://doi.org/xx.xxxx/xxxx.xxxx.xxxx. *(to be posted.)*

our findings, preventing them from being implicitly influenced by relationships between parameters specific to the data from the partnering pharmacy. The mechanical details along with supporting and supplemental numerical results are presented in Appendix D.

Figure 2 summarizes the performance of each policy relative to the lower bound on the optimal cost, where we assume the cost of waiting w is uniformly one across medications. The y-axis is the average cost per unit time as measured by (12) relative to the lower bound, which is uniformly 100%, and the x-axis is system size n, where each point represents the average performance across 30 replications (with 95% confidence intervals as error bars). For systems with only a few medications, the average cost rate for both the *oldest* policy and the *SDRE-Index* policy are more than twice the lower bound, however as the number of unique medications (n) grows, the performance of both policies improves. The average performance of the *oldest* policy stabilizes at approximately 70% greater than the lower bound. The SDRE-index policy (23) exhibits considerably stronger performance, particularly for larger systems, e.g., for the system with 124 unique medications the performance is on average only 11.27% greater than the lower bound. This experiment thus demonstrates proof of concept for the proposed heuristic with strong theoretical performance in large scale settings, and which we also note indirectly demonstrates that the lower bound is relatively tight (i.e., close to optimal) for large systems.[6]

Figure 2 Relative Heuristic Performance Comparison



The fact that the SDRE-Index policy performs better is, in and of itself, not surprising. Compared to the *oldest* policy, the *SDRE-Index* policy requires more information, e.g., arrival rates, switchover, and processing times, which enable it to exploit shorter switchover

[6]The relatively poor performance of the heuristics for small scale systems may indicate that the lower bound is substantially lower than the optimal (i.e. is not tight) for small systems.

times when choosing between medications to improve performance. However, the important insights from these results are that (1) there is substantial room for improvement in average cost of waiting when following the *oldest* policy, and (2) the *SDRE-Index* yields greater relative improvement over the *oldest* policy in larger systems that are more representative of real-world compounding pharmacies.

To assess the potential improvement from switching from the *oldest* policy to the *SDRE-Index* in progressively more realistic settings, we conduct 2 further calibrated simulation experiments using the data provided by the partnering pharmacy. The first reintroduces stochastic request arrivals (which we had assumed away up until now) and the second adds the need to estimate dynamic arrival rates from previous requests.

5.2. Stochastic Simulation – Poisson arrivals with constant and known rates

This experiment reintroduces stochasticity in arrivals, which are assumed to follow Poisson processes with known and fixed rates $\lambda \in \mathbb{R}^n$. We calibrate the arrival rates for a subset of 1,192 compounded medications using approximately 4 years of data which is detailed in Appendix B. Each medication is identified by the combination of ingredients, their respective dosages, and the medication form, e.g., tablets vs capsules, and this information is used to compute the switchover/processing times S which are based on expert practitioner opinion. The mechanical details along with supporting/supplemental results are provided in Appendix E.

Adapting the fluid-based heuristic to a stochastic setting requires a small numerical refinement to the *SDRE-Index* heuristic. In the stochastic simulation, queues can be empty, in which case Proposition (1) would fail to hold (specifically the upper sub-matrix of $B(x_t)$ would not be invertible). Intuitively, this is also related to the stabilazability of the system and follows from the infinite-horizon (steady-state) approximation based on instantaneous dynamics, which implies queue lengths will go to infinity for queues that are empty. To see this, consider that if a given queue is empty (there are no waiting requests for that medication), a batch from that queue would be of size zero (because production is make-to-order). Thus, there is no way for future arrivals occurring at rate λ to depart the system, meaning the queue length will tend to infinity over an infinite horizon given the instantaneous dynamics. Thus, we introduce a minor numerical refinement that ignores parameters for empty queues (which would not be processed next anyway). This is achieved by eliminating the parameters within (23) related to queues that are empty, unless that

medication was the most recently produced in which case we ignore anticipated future arrivals.

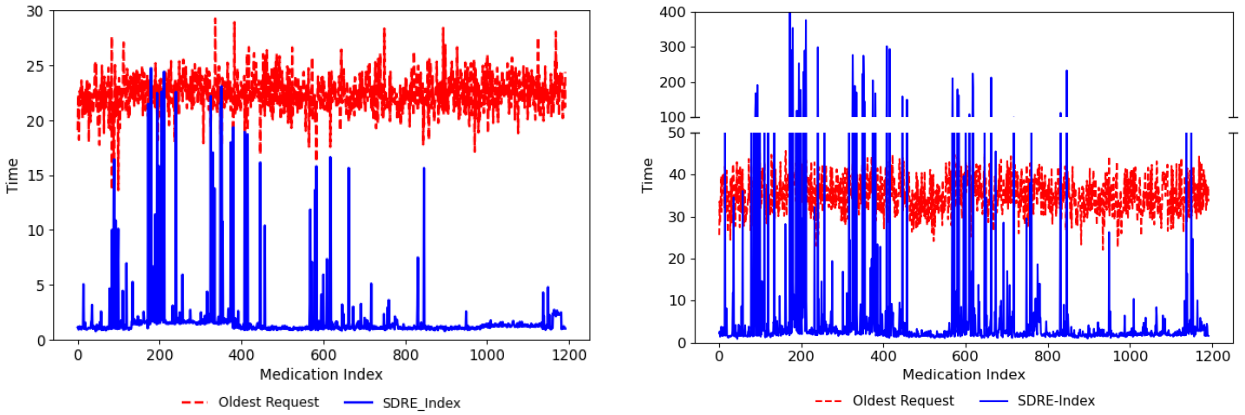
With the refinements for a stochastic setting discussed, we next compare the policies. We begin with a comparison of the performance as measured by the costs of waiting metric (12). As in the fluid setting when the cost per request per unit time is normalized to be 1 across all medications, the *SDRE-Index* policy greatly reduces the average cost rate (as measured by (12)) compared to the *oldest* policy. Specifically, the *oldest* policy results in an average cost rate of 37.1 compared to 18.8 for the *SDRE-Index*, a reduction of almost 50% in the average cost of waiting.[7]

We now turn our focus from the average cost of waiting across the entire system to a comparison of the time medication requests spend in the system in order to provide a better sense of what this means for reductions in production delays for patients. Figure 3a depicts the average time spent in system across the 1,192 unique medications (each with its own queue) for the *SDRE-Index* and the *oldest* policies. The *SDRE-Index* policy leads to a reduction in the average time in the system for almost all medications (98.8%), and reduces the average time in system by almost half, from 19.0 hours to 9.7 hours. Illustrating substantial benefits in terms of reduced production delays.

However, considering the worst case performance (i.e., maximum time spent in the system within each medication) as depicted in Figure 3b, the *SDRE-Index* policy is in some cases significantly worse. Specifically the maximum time in the system under the *SDRE-Index* policy is 403 hours, which is almost 9 times longer than the 45.7 hours under the *oldest* policy. This is not surprising if one considers the *oldest* policy as analogous to a minimax policy. By always choosing the oldest request, i.e., the queue with the request that has the maximum waiting time, this policy in effect attempts to minimize the maximum time any request can spend in the system, but this comes at cost of increasing average costs/delays. This result offers a key insight into why the *oldest* policy is used in practice: in addition to being extremely simple to implement, it ensures roughly uniform delays across medications with strong worst case performance in a stochastic setting.

[7] We note the costs per unit time are significantly lower than smaller systems in the fluid model as shown in Appendix D. The reason is that when many medications are requested infrequently (i.e. arrival rates are very low for individual medications), most queues are empty most of the time in the stochastic model, but in the fluid model no queues are ever empty, thus costs are incurred at a lower rate in the stochastic simulation.

Figure 3 SDRE vs. Oldest



(a) Average (b) Worst Case

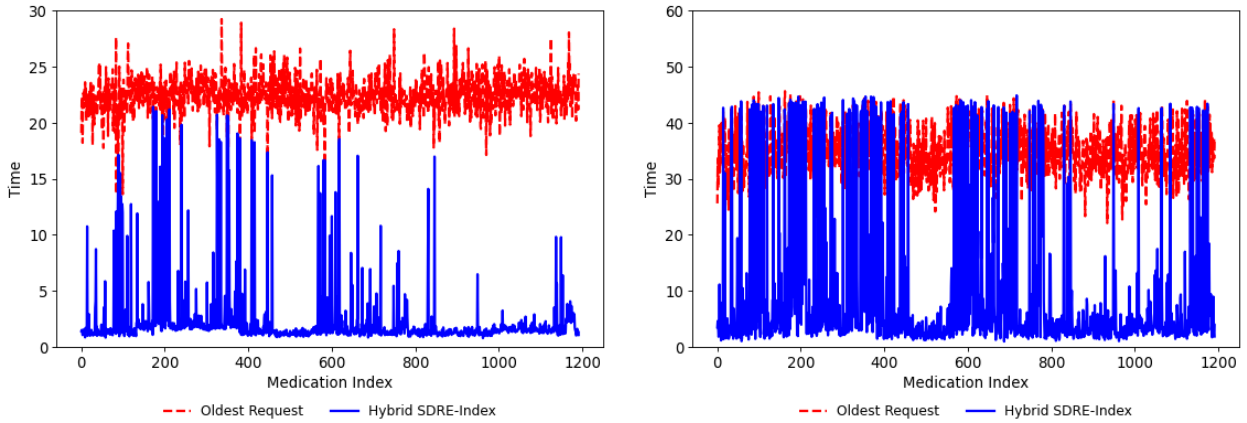
In order to address the worst case performance when following the *SDRE-Index* policy, we test a simple *Hybrid SDRE-Index* with an age limit. Specifically, given an age limit, the *Hybrid SDRE-Index* policy serves any queues with requests above the age limit, and if all are under the age limit, it follows the *SDRE-Index* policy. The idea being that age-limit imposes an indirect trade off between average performance and worst case in order to ensure a given worst case performance. By setting the age limit to slightly better than the worst case of the *oldest* policy (in this case 40 hours), the *Hybrid SDRE-Index* policy is able to substantially improve on average delays without sacrificing worst case delays. The average cost per unit time for the *Hybrid SDRE-Index* policy is 21.9, a relative reduction of 41% compared to the *oldest* policy (37.1), and average time spent in system (depicted for each medication in 4a) is substantially reduced from 19.0 to 11.2 without meaningful sacrifices in worst case delays as shown in 4b.

Ultimately, this simulation demonstrates that in a calibrated stochastic setting with more than 1000 distinct medications, the *SDRE-Index* policy based on a fluid model, continues to show strong ability to improve average performance over the *oldest* policy. Additionally, to ensure comparable worst case performance, a simple *Hybrid-SDRE-Index* policy imposing an age limit is sufficient, and sacrifices relatively little in terms of average performance.

5.3. Data-Driven Simulation

In this final numerical experiment we compare the performance of the *hybrid-SDRE-Index* proposed in the previous experiment and the *oldest* policy in a data driven simulation where arrival rates are unknown, dynamic, and must be estimated from past arrival data.

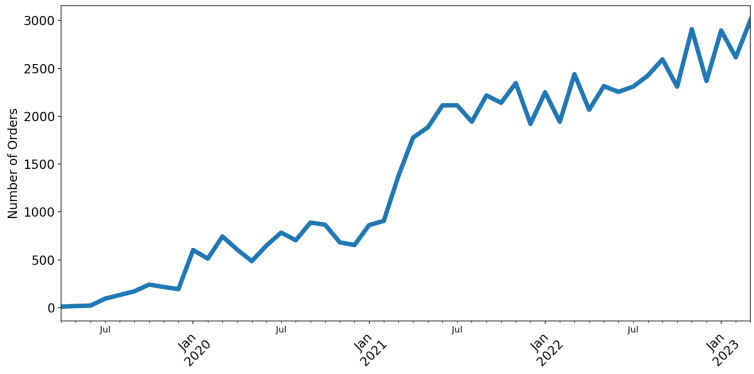
Figure 4 Hybrid SDRE vs. Oldest



(a) Average (b) Worst Case

For the same 1192 medications used in the previous simulation, we use the associated 67,563 distinct order requests spanning from April-2019 to March-2023 as the arrival process. The aggregated orders per month is shown in Figure 5. We note that while the partnering pharmacy expanded capacity as demand increased, we model a single server processing all requests to focus on a comparison of the policies when the *SDRE-index* relies on estimated arrival rates within a dynamic setting.

Figure 5 Total Orders over Time



For each medication, we estimate the arrival rate based on the past week (7 days) worth of arrivals, where medications not ordered within the last week are assumed to have an arrival rate of 1 per year. We update the estimated arrival rates every 8 hours and use these in computing the indices (23), where the estimated arrival rates λ appear in $B(x_t)$. While these could also be used in Q, we introduce a modification in this setting to minimize the effect of unknown true arrival rates by using information that is readily available

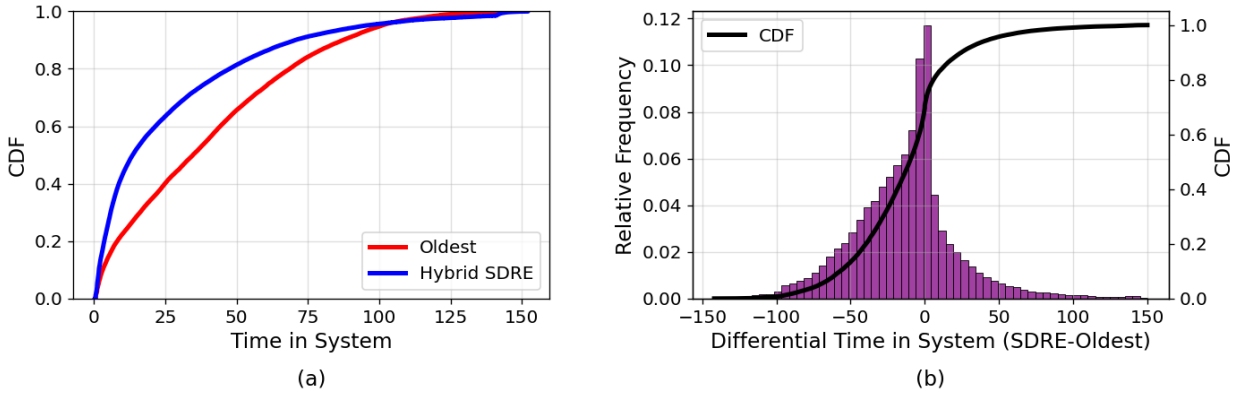
(specifically the request ages from the *oldest* policy and the queue lengths). Specifically, x^TQx captures the cumulative weighted age of all requests in the system for a fluid system but is only an approximation in a stochastic setting.

To make this exact in a stochastic setting, we set the diagonal terms of the upper left sub-matrix of Q as the sum of cumulative ages divided by the square of the queue length for each medication. We denote this as Q^{age} such that $x^TQ^{age}x$ captures the cumulative weighted age of all requests in the system in the stochastic system. This modification grounds the cost function in the actual, observable state of the system (the age of waiting requests) rather than relying solely on noisy arrival rate estimates. To maintain comparable worst case performance, we continue to use the hybrid policy with an age limit developed in the previous experiment, where we again set the age limit based on the performance of the *oldest* policy to maintain comparable worst case delays. The remaining mechanical details and supporting results are provided in appendix F.

Switching from the *oldest* request policy to the (age corrected) *hybrid-SDRE-index* policy reduces the average cost of waiting by 32.5% (from 79.5 to 53.7) and reduces the average time requests spend in the system from 40.1 hours to 27.0 hours. Sub-figure (a) of Figure 6 plots the CDFs of time in the system for each policy – demonstrating a higher probability of spending less time in the system for up to approximately 100 hours, without significantly sacrificing worst case performance as ensured by the age-limit hybridization. Sub-figure (b) of Figure 6 provides a more direct comparison of the potential impact on medication delays for patients. It depicts the distribution of the pairwise differences in the time spent in the system for each request, where a negative differential indicates requests that spend less time in the system under the *SDRE-Index* policy compared to the *oldest* policy. While not every order is expedited, approximately 70% of medication requests are filled faster under the *hybrid SDRE-index* policy.

The results of this experiment demonstrate that even in a stochastic setting with dynamic arrival rates that must be estimated from data, the (age corrected) *hybrid-SDRE-index* policy, originally developed from a fluid model, yields substantial reductions in production delays for patients compared to the *oldest* policy. Moreover they illustrate proof of concept on real data and show promise of a scalable approach to sequencing production in compounding pharmacies that is implementable in practice.

Figure 6 **Policy Comparison – Oldest vs. hybrid-SDRE-Index**



6. Discussion

This paper introduces a novel and easily implementable heuristic for production sequencing in compounding pharmacies. Our numerical results demonstrate the strong theoretical performance for large (fluid) systems, where the proposed heuristic is within 11% of optimal on average. In a more realistic and data-driven setting when arrival rates are unknown and dynamic, a hybrid policy with an age limit shows that this approach can improve patient access to compounded medications by significantly reducing average production delays by approximately 32.5%, without sacrificing worst-case performance.

Within the compounding pharmacy context, one interesting direction for future research is extending the model to a multi-server environment, which better reflects the operational scale of many compounding pharmacies operating multiple production stations in parallel. In practice, different stations may specialize in producing particular types of medications or dosage forms (e.g., creams, capsules, or liquids). This introduces new challenges, including balancing workloads across stations while accounting for setup times and managing cross-contamination risks. In a larger-scale setting, the multi-server framework could even capture the problem of dispatching orders across different facilities. Extending the SDRE-Index heuristic to manage multi-server operations could yield valuable insights into efficiently allocating production tasks across multiple resources.

Methodologically, our works makes two key contributions: it extends the application of the SDRE literature to production control, and more generally, it establishes proof-of-concept for how the SDRE architecture can be modified for use as an ADP technique. By designing a theory-driven cost function based on the problem's underlying structure, our method eliminates the need to learn the value function and the optimal policy, offering a more direct and tractable approach to the classical value function approximation approach.

Following from this, a second, broader avenue for future research lies in generalizing the approach of applying modified SDRE controllers as an ADP method, and specifically a value function approximation technique, to other challenging production control and queueing problems where optimal policies are intractable or impractical. This could be achieved by altering the state space, fluid model, and cost function to be representative of alternatives settings to further test, refine, and generalize the approach. For instance, the emerging field of continuous pharmaceutical manufacturing, which requires novel control models to ensure production quality as noted by Martagan et al. (2024), presents opportunities for extending and testing this approach.

References

340B Health. 2023. Restrictions on 340B contract pharmacy increase drug company profits but lead to lost savings, patient harm, and substantial burden for safety-net hosptials. URL https://www.340bhealth.org/files/Contract_Pharmacy_Survey_Report_March_2023.pdf.

Adelman, D., C. Barz. 2014. A unifying approximate dynamic programming model for the economic lot scheduling problem. *Mathematics of Operations Research* **39**(2) 374–402.

Alla, A., D. Kalise, V. Simoncini. 2023. State-dependent riccati equation feedback stabilization for nonlinear pdes. *Advances in Computational Mathematics* **49**(1) 9.

Bemporad, A., M. Morari, V. Dua, E. N. Pistikopoulos. 2002. The explicit linear quadratic regulator for constrained systems. *Automatica* **38**(1) 3–20.

Bertsimas, D. J., H. Xu. 1993. Optimization of polling systems and dynamic vehicle routing problems on networks .

Boon, M., O. J. Boxma, E. Winands. 2011. On open problems in polling systems. *Queueing Systems* **68** 365–374.

Borst, S., O. J. Boxma. 2018. Polling: past, present, and perspective. *TOP* **26** 335–369.

Boyd, S. 2008. EE363: Lecture Slides. Lecture 1. https://stanford.edu/class/ee363/lectures/dlqr.pdf.

Chang, I., J. Bentsman. 2013. Constrained discrete-time state-dependent riccati equation technique: A model predictive control approach. *52nd IEEE Conference on Decision and Control*. IEEE, 5125–5130.

Çimen, T. 2010. Systematic and effective design of nonlinear feedback controllers via the state-dependent riccati equation (sdre) method. *Annual Reviews in control* **34**(1) 32–51.

Dobson, G. 1992. The cyclic lot scheduling problem with sequence-dependent setups. *Operations Research* **40**(4) 736–749.

Dobson, G., D. Tilson, V. Tilson. 2015. Optimizing the timing and number of batches for compounded sterile products in an in-hospital pharmacy. *Decision Support Systems* **76** 53–62.

Duenyas, I., M. Van Oyen. 1996. Heuristic scheduling of parallel heterogeneous queues with set-ups. *Management Science* **42**(6) 814–829.

Dutka, A. S., A. W. Ordys, M. J. Grimble. 2005. Optimized discrete-time state dependent riccati equation regulator. *Proceedings of the 2005, American Control Conference, 2005.*. IEEE, 2293–2298.

Hu, Y., J. Dong, O. Perry. 2022. Asymptotic optimality of the binomial-exhaustive policy for polling systems with large switchover times. *The Annals of Applied Probability* **32**(6) 4803 – 4848.

Kim, E., M. Van Oyen. 2000. Finite-capacity multi-class production scheduling with set-up times. *IIE transactions* **32**(9) 807–818.

Lan, W.-M., T. L. Olsen. 2006. Multiproduct systems with both setup times and costs: Fluid bounds and schedules. *Operations Research* **54**(3) 505–522.

Lan, W. 2000. Dynamic scheduling of multi-product systems: Bounds and heuristics. Ph.D. thesis, PhD dissertation, University of Michigan.

Martagan, T., M. Baaijens, C. Dirckx, J. Holman, R. Meyer, O. Repping, B. van Ravenstein. 2024. Msd: Continuous pharmaceutical manufacturing data for the 2024 msom data-driven research challenge. *Manufacturing & Service Operations Management* **26**(5) 1587–1604.

Masselink, I. H., T. L. van der Mijden, N. Litvak, P. T. Vanberkel. 2012. Preparation of chemotherapy drugs: Planning policy for reduced waiting times. *Omega* **40**(2) 181–187.

Mathavaraj, S., R. Padhi, S. Mathavaraj, R. Padhi. 2021. Infinite-time lqr and sdre for satellite formation flying. *Satellite Formation Flying: High Precision Guidance using Optimal and Adaptive Control Techniques* 45–65.

Minghetti, P., D. Pantano, C. G. M. Gennari, A. Casiraghi. 2014. Regulatory framework of pharmaceutical compounding and actual developments of legislation in europe. *Health Policy* **117**(3) 328–333.

Nekoo, S. R. 2019. Tutorial and review on the state-dependent riccati equation. *Journal of Applied Nonlinear Dynamics* **8**(2) 109–166.

Niño-Mora, J. 2009. A marginal productivity index rule for scheduling multiclass queues with setups. *Network Control and Optimization: Second Euro-NF Workshop, NET-COOP 2008 Paris, France, September 8-10, 2008. Revised Selected Papers 2*. Springer, 78–86.

Olsen, T. L. 1999. A practical scheduling method for multiclass production systems with setups. *Management Science* **45**(1) 116–130.

Osteen, R. B. 2018. Batch compounding. *Compounding Sterile Preparations* .

Pearson, J. 1962. Approximation methods in optimal control i. sub-optimal control. *International Journal of Electronics* **13**(5) 453–469.

Powell, W. B. 2016. Perspectives of approximate dynamic programming. *Annals of Operations Research* **241** 319–356.

Precedence Research. 2025. Compounding pharmacy market size, share and trends 2025 to 2034. URL https://www.precedenceresearch.com/compounding-pharmacy-market.

Reiman, M. I., L. M. Wein. 1998. Dynamic scheduling of a two-class queue with setups. *Operations Research* **46**(4) 532–547.

Richardson, D. 2019. Operations research frameworks for improving make-ahead drug policies at outpatient chemotherapy infusion centers. Ph.D. thesis, University of Michigan.

Sun, X., X. Zhu. 2025. Dynamic control of a make-to-order system under model uncertainty. *Management Science* .

Takagi, H. 1997. Queueing analysis of polling models: Progress in 1990– 1994. *Frontiers in Queueing: Models and Applications in Science and Engineering*. CRC Press, New York.

Tedrake, R. 2024. *Underactuated Robotics: Algorithms for Walking, Running, Swimming, Flying, and Manipulation*. MIT. URL https://underactuated.csail.mit.edu. Last modified 2025-02-14.

van der Wal, J., U. Yechiali. 2003. Dynamic visit-order rules for batch-service polling. *Probability in the Engineering and Informational Sciences* **17**(3) 351–367. doi:10.1017/S0269964803173044.

Appendix

A. Proofs

Proof of Lemma 1.

Let π^* be an optimal policy. Note that all medications must be visited at least once. Otherwise, the cost in equation (3) under policy π^* is not finite, contradicting to its optimality. For any initial state $(q_1^s, \cdots, q_n^s)$, we can restart the calculation at the first time when all medications are all produced for at least once under the optimal policy π^*. Since equation (3) calculates the average cost, restarting the calculation does not change the optimal value. Thus, it suffices to consider a countable set of states in which q_i^s can be written as $q_i^s = \lambda_i \sum_{j,k=1}^{n} \alpha_{j,k} S_{j,k}$ for some integers $\alpha_{j,k}$. That is, we can ignore the initial queue lengths of the medications and consider the case when the queue length of medication i is the total new arriving demand between two consecutive service of this medication.

Now, for an initial state $(q_1^s, \cdots, q_n^s)$, if any state is visited more than once, we can construct a cyclic policy that achieves the same cost. To be specific, suppose $(\tilde{q}_1^s, \cdots, \tilde{q}_n^s)$ will be visited more than once. Then, the dynamics of the system repeats itself every time it visits the state $(\tilde{q}_1^s, \cdots, \tilde{q}_n^s)$. We can construct a route r^* using the production sequence between the consecutive visits of the state $(\tilde{q}_1^s, \cdots, \tilde{q}_n^s)$. Then the cyclical policy achieves the same cost of equation (3) under π^*.

We will end the proof by showing that under the optimal policy π^*, there exists at least one state $(\tilde{q}_1^s, \cdots, \tilde{q}_n^s)$ that will be visited for more than once by contradiction. Suppose this is not true. Then at least one of the queue length will be unbounded. Otherwise, there are only finite number of states that will be visited. Next, we will argue that for any large number K, there exists a time t_K that at least the queue length of one queue exceeds K for all $t \geq t_K$. Since there is no state that will be visited for more than once, the queue length must exceeds K after a certain time. This is because there are only finite number of states that all queue lengths are below K. However, this implies that the cost is unbounded, contracting to the optimality of the policy. This completes the proof. Q.E.D.

Proof of Proposition 1. Given the choice of Q and R, the discrete time algebraic Riccati equation with state dependent parameters (19) reduces to,

$$P(x_t) = Q + A^T \left(P(x_t) - P(x_t)B(x_t)(B(x_t)^T P(x_t) B(x_t))^{-1} B(x_t)^T P(x_t) \right) A. \tag{24}$$

The proof follows from substituting Q for $P(x_t)$ in (24). Specifically,

$$Q = Q + A^T \left(Q - QB(x_t)(B(x_t)^T QB(x_t))^{-1} B(x_t)^T Q\right) A. \tag{25}$$

For this to be true, we need to show that $A^T \left(Q - QB(x_t)(B(x_t)^T QB(x_t))^{-1} B(x_t)^T Q\right) A = 0$ or equivalently,

$$Q = QB(x_t)(B(x_t)^T QB(x_t))^{-1} B(x_t)^T Q. \tag{26}$$

Substituting in, $Q = \begin{bmatrix} \tilde{Q} & 0 \\ 0 & 0 \end{bmatrix}$, $B(x_t) = \begin{bmatrix} \tilde{B}(x_t) \\ I \end{bmatrix}$, and then distributing the inverse, the right hand side of (26) reduces to,

$$\begin{bmatrix} \tilde{Q}\tilde{B}(x_t)\tilde{B}(x_t)^{-1}\tilde{Q}^{-1}(\tilde{B}(x_t)^T)^{-1}\tilde{B}(x_t)^T\tilde{Q} & 0 \\ 0 & 0 \end{bmatrix}. \tag{27}$$

Therefore, when $\tilde{B}(x_t)$ and $\tilde{Q}$ are invertible, we have that $B(x_t)B(x_t)^{-1} = I = (B(x_t)^T)^{-1}B(x_t)^T$ and $Q^{-1}Q = I$, which when substituted into the above yield the result (where we note uniqueness follows from the uniqueness of the inverse of a square matrix). Given that $\tilde{Q}$ is a diagonal matrix with non-zero terms, it is invertible. We next show that $\tilde{B}(x_t)$ is either invertible or a small perturbation yields an invertible $\tilde{B}(x_t)$, and thus can be assumed invertible without loss of generality.

To prove $\tilde{B}(x_t)$ is invertible we use the Sherman-Morrison formula which computes the inverse of a rank 1 update. The Sherman-Morrison formula shows that given an invertible matrix A and two column vectors u, and v (of appropriate dimension), then $A + uv^T$ is invertible iff $1 + v^T Au \neq 0$. Writing the terms of of $\tilde{B}(x_t)$ following this structure, we have that $\tilde{B}(x_t) = [-\operatorname{diag}(x_t^T\tilde{Q})] + \lambda \otimes x_t^T\tilde{V}S$. Here $[-\operatorname{diag}(x_t^T\tilde{Q})]$ is a diagonal matrix with non-zero terms is invertible and can be substituted in for A[8], λ can be substituted in for u and $x_t^T\tilde{V}S$ can be substituted in for v^T.

Therefore as long as $x_t^T\tilde{V}S[-\operatorname{diag}(x_t^T\tilde{Q})]\lambda \neq -1$, then $\tilde{B}(x_t)$ is invertible. Thus, while it is technically possible that given the problem parameters there exist some set of states (x_t) such that $x_t^T\tilde{V}S[-\operatorname{diag}(x_t^T\tilde{Q})]\lambda = -1$, then perturbing one of the problem parameters by $\epsilon > 0$ will break the equality, and therefore without loss of generality $\tilde{B}(x_t)$ can be assumed to be invertible. Q.E.D.

[8]We note the non-zero terms along the diagonal is always true in the fluid system where queues are never empty and that this can easily be generalized to a stochastic system by removing data from queues that are empty (and therefore are not under consideration for the next queue to process).

B. Data Description

We collaborate with Roseway Labs, a compounding pharmacy in the United Kingdom. Compounding pharmacies produce customized medications for individual patients from raw ingredients. These medications are not commercially available, such as those that treat rare or complex health conditions, or those that meet the needs of patients with allergies or sensitivities where pharmacists can substitute different features of the medication, such as the capsule coating to avoid certain films or sugars.

We collect order level data over the course of four years from April 2019 to March 2023, where data was anonymized and no patient or provider identifiable data was included. During this time, the pharmacy processed approximately 100,000 orders. Of these, 20,000 orders include only off-the-shelf prescriptions that do not require laboratory compounding and are fulfilled by a separate team of pharmacists. The number of orders varies significantly over time. It ranges from no orders during a business day to more than 250 orders. Orders can arrive during operating hours as well as outside of them. On average, there are 60.5 orders per day.

Each order is identified by an order number and includes a time stamp down to the second, item description, quantity, form of the product, dispatch method, and the raw ingredients. The same combination of ingredients can take multiple forms, including capsules, tablets, creams, gels, and foams. Raw ingredients may also exist in multiple dosages. For example, Paroxetine is used in dosages from 10mg to 80mg in 10mg increments. We process the order data and create unique ingredient-dosage identifiers that can capture the hierarchy of the compounding process, i.e., a set of drugs can be compounded in sequence. For example, consider a set of orders for Minoxidil 2.0%, Minoxidil 5.0%, Minoxidil 6.0% which can be compounded in sequence by successively increasing the concentration of Minoxidil (the active ingredient). Therefore, compounding the orders in this sequence mitigates the risk of over-dosage and enables shorter switchover times between batches, saving significant amounts of lab and technician resources. However, if not completed in increasing sequence, time consuming cleaning procedures are required to mitigate risks and ensure exact dosage. However, all three orders need to be in the queue when the compounding begins in order to take advantage of the time savings. The pharmacy does *not* compound drugs for inventory.[9] Note that our proposed policy does not directly target sequences

[9]In the U.S., the FDA does not allow compounding pharmacies that fulfill patient prescriptions (503A) to compound drugs from inventory.

that process orders in increasing dosage; however, we do take advantage of such structures through our waiting cost calculations. The ingredient-dosage identifiers allow us to identify medication pairs where switching costs are significantly reduced.

The order data contains 727 unique ingredient combinations used to formulate 4,682 dosages that are then made into 5,378 unique medications when accounting for different product forms. When looking at ingredient combinations, the top 10 combinations make up 78% of orders. Conversely, there are a significant number of medications that are rarely ordered. We define a rare order as a medication that is requested less than once a year on average. We exclude this subset of medications, which account for 13,615 of the 81,178 orders (16.8%) in our data. This yields a sample of 1,192 distinct medications accounting for 67,563 orders. This processed dataset is used by the stochastic and data-driven simulation and is publicly available at `https://doi.org/xx.xx.xx` *(to be posted)*.

The respective switchover and processing times are given in Table B.1 and eq. (28) below.

$$S_{ij} = \begin{cases} 3\,Minutes \text{ if} & Ingredients_i = Ingredients_j \,\& \\ & Form_i = Form_j \,\& \\ & Dosage_i \geq Dosage_j \\ 15\,Minutes \text{ if} & Ingredients_i = Ingredients_j \,\& \\ & Form_i \neq Form_j \\ 20\,Minutes & \text{otherwise.} \end{cases} \tag{28}$$

C. Benchmark Policies for Numerical Experiments

Due to its us in practice our main benchmark policy is the *oldest* policy. In the fluid model the oldest request policy is a simple index where the age of the oldest request in each queue is given by the queue length divided by the arrival rate q_i/λ_i. In the stochastic setting the policy is implemented by taking the maximum of the difference between the current simulation time and the arrival epochs for all requests in the system.

In appendices D,E, and F we include supplemental results for a less naive heuristic we denote as the *X-Index* (Interchange-Index) as a secondary heuristic benchmark policy

Product Form	**Time (Minutes)**
Capsules	45
Cream	15
Topi CLICK	15
Gel	15
Lozenges	30
Pessary	30
Tonic	15
Tablets	90
Oral Drops	15
Suspension	15
Foam	15
Serum	15

Table B.1 Production Times (without switchover)

to ensure the estimated magnitude of the potential improvement from implementing the SDRE-index heuristic is robust. Given the most recently produced medication i, the *X-Index* policy prioritizes producing medication j with the highest index as defined by,

$$\frac{w_j q_j}{S_{ij}} - w_j \lambda_j / 2. \tag{29}$$

To motivate the X-Index heuristic, consider the case when the changeover time is sequence independent. Thus, we denote the total processing as S_i for product i. We compare the total cost two sequences: serving i before j and serving j before k. If we serve class i before class j, then the total holding cost of products i and j in next two period is

$$\begin{aligned} &(w_i(q_i + S_i\lambda_i)/2 + w_j[q_i + S_i\lambda_j/2])S_i \\ &+ (w_j(q_j + (S_i + S_j)\lambda_j)/2 + w_i(S_i\lambda_i + S_j\lambda_i/2)S_j. \end{aligned} \tag{30}$$

If we switch the order, the total holding cost of products j and k incurred is

$$\begin{aligned} &(w_j(q_j + S_j\lambda_j)/2 + w_i[q_i + S_j\lambda_i/2])S_j \\ &+ (w_i(q_i + (S_i + S_j)\lambda_i)/2 + w_j(S_j\lambda_j + S_i\lambda_j/2)S_i. \end{aligned} \tag{31}$$

Note that to compare the two produce sequence, the holding cost of other products are the same, so it is optimal to serve class i before class j if the following holds:

$$\frac{w_i q_i}{S_i} - w_i \lambda_i / 2 \geq \frac{w_j q_j}{S_j} - w_j \lambda_j / 2. \tag{32}$$

An optimal policy should pick the class with the greatest index $w_i q_i / S_i - w_i \lambda_i / 2$. We then account for sequence-dependence when implementing this policy by simply substituting $S_{i,j}$ for S_i yielding index given by (29) and then picking the product with the greatest index.

D. Fluid Model Simulation

This appendix provides the mechanical details and supporting/supplemental results for the simulation within a fluid setting presented in Section 5.1.

D.1. Mechanical Details

To ensure comparability of policies and facilitate replicability of results, we select 30 random sets of parameter randomization and initialization seeds (chosen from discrete uniform random variables ranging from 1 to 65,536) that are used to randomize and initialize the parameters for each replication. Specifically, these ensure that within each replication for a given system size $n \in \{4, 12, 24, 36, 72, 124\}$, each policy is compared on the same combination of parameters with the same initial system conditions and ensures replicability of the exact estimates for the simulated performance measures.

Given a parameter randomization seed, we draw the n service times (excluding set-up time) from a uniform distribution with minimum 0.25 (representing a quarter of a hour) and a maximum of 1.5 (representing 1.5 hours), i.e., ($\mu_i \sim U[.25, 1.5]$) where the unit of time represents 1 hour. This is to roughly reflect the scale of variable production times at the partnering compounding pharmacy based on discussions with the leadership team. Changeover times S_{ij} are drawn from a discrete distribution taking values of .05, .25, and .33 (corresponding to 3,15 and 20 minutes respectively) with probabilities .1, .3, and .6 for $i \neq j$ and 0.05 for $i = j$, to capture that the vast majority of options when selecting the next medication will require longer switchover times, and that repeating the same medication always has minimum switchover time. Arrival rates are drawn $\lambda_i, i \in 1, ..., n$ from an exponential distribution with scale parameter 0.05 $\lambda_i \sim exp[scale = .05]$ to capture that the vast majority of compounded medications are requested rarely with a small number of more common medications.

For each set of randomized queueing parameters and system size, we set $W = 1$ uniformly for all queues, i.e., we assume all medications have the same cost of delay, and then solve for the lower bound (1) using a gradient descent algorithm to serve as a benchmark that the heuristics can be compared relative to.

Given an initialization seed, we then draw random initial queue lengths following a discrete uniform distribution taking values between 1 and n for each queue, and randomly allocate the server to one of the queues with uniform probability. Then, following a given heuristic policy for choosing the next queue, the simulation run-in / warm-up period is set to one month (31 days), after which point the linear objective function (12) is computed after each decision, generating a sequence of this performance measure over time. The simulation runs for up to an additional year, but terminates early if the sequence converges. We assess the sequence as converging if the range of the performance measure sequence over the previous $n * 100$ decision epochs is less than one percent of the final value in the sequence, and each queue has been visited at least 30 times. We then re-initialize using the same initialization seed (to ensure the system always starts in the same state across policy comparisons) and repeat this process for the remaining decision policies.

Using the set of random parameter and initialization seeds, we then repeat 30 replications of the process above, for each computing the lower bound and the linear objective function (12) for each policy.

D.2. Results

Performance results for each individual replication under different system sizes (n) are shown in the subfigures of Figure D.1. The y-axis of each subfigure captures the cost rate (cost per unit time), and the x-axis is the replication number. Average absolute performance across replications is depicted in the left subfigure of D.2, compute 95% confidence intervals for the average depicted as error bars. Average performance relative to the lower bound shown in the right subfigure.

We remark on the performance of the *X-index* policy as depicted in D.2 as this policy is not included in the main paper to maintain focus. Within a fluid setting the *X-index* policy also significantly outperforms the *oldest* policy. As with the *SDRE-index* this is not supersizing as the *X-index* policy exploits problem structure while the oldest request policy is naive. For a system with 124 unique medications the *X-index* policy is on average within 35% of the lower bound on the optimal cost rate, where we note the *SDRE-index* policy

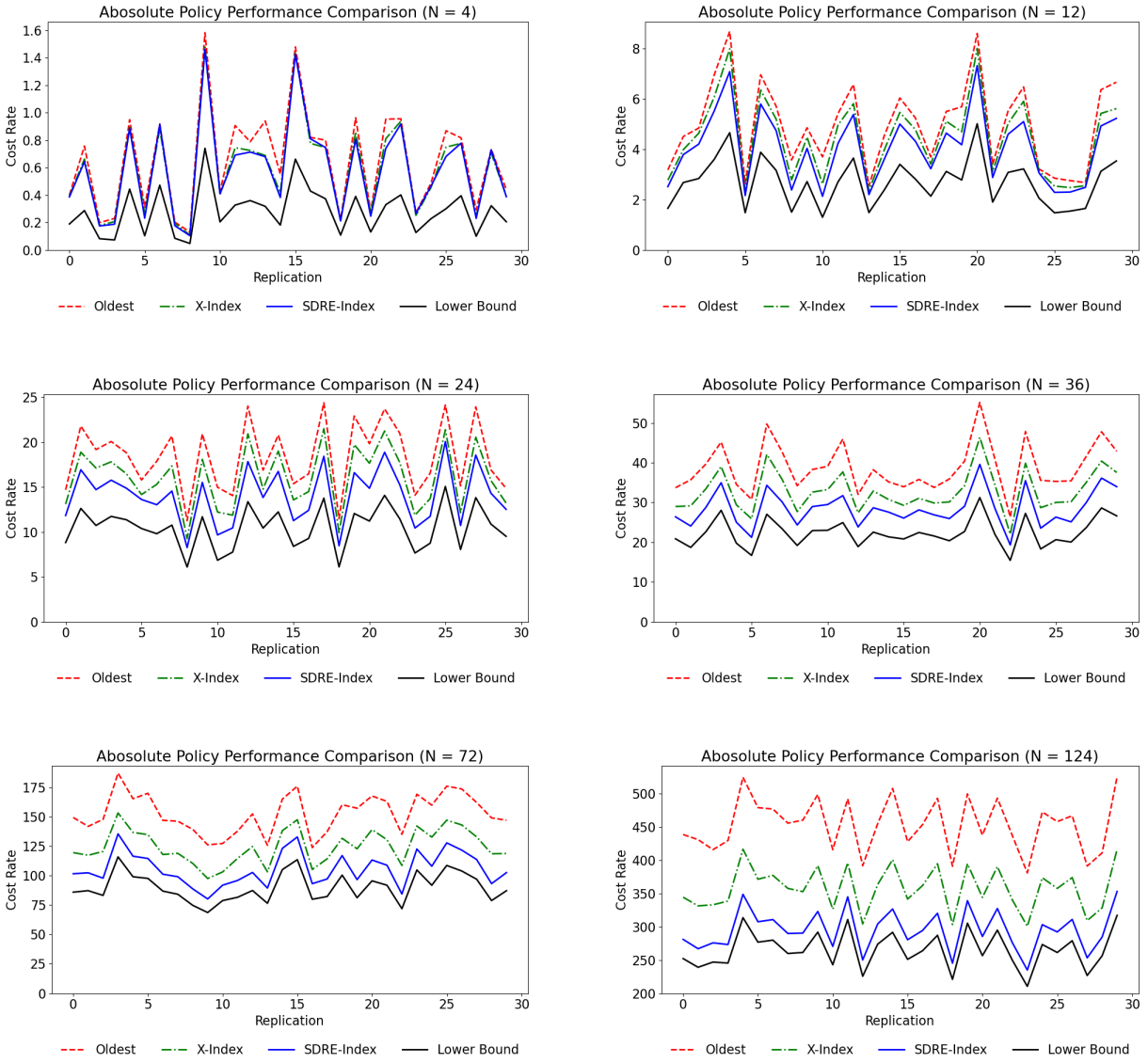


Figure D.1 Performance Comparisons Across Replications

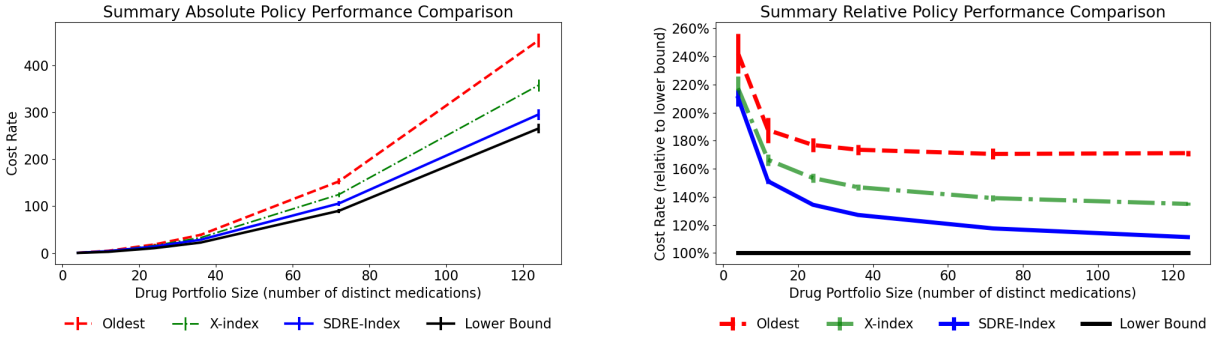


Figure D.2 Summarized Performance Comparison

is on average within 11%. This suggests that the *SDRE-index* can still provide substantial benefits over a less naive policy.

In the subsequent numerical experiments, we estimate the potential benefit of switching to the *SDRE-index* from the *oldest* policy, as the *oldest* is used in practice, but we also provide comparisons to the *X-index* in the supplement results to demonstrate that the *SDRE-index* continues to outperform both alternative heuristics.

E. Stochastic Simulation

For this simulation we use the average arrival rates for 1,192 of the most common medications that account for 83% of all medication requests. Guided by expert opinion at the partnering compounding pharmacy, we set the service times depending on the medication type as shown in Table B.1, and the switchover times following (28).

E.1. Mechanical Details

We being by specify an initialization seed, which randomly allocates the server to a given queue when the simulation is initialized. For each policy, we then simulate an identical 7 days worth of arrivals before the server begins processing requests. We use the following 31 days as warm-up period before collecting the performance measures (costs of waiting metric (12) and the time in system for all requests). The simulation terminates after the equivalent of 10 additional years (on top of the initial 7 days and the 31 day warm-up period), at which point the waiting time data is processed.

E.2. Results

Below we present the key simulation outputs for each policy as a combination of summary metrics and figures depicting average and maximum waiting times across each medication for the (1) *oldest* policy, (2) *SDRE-index* policy, (3) the *Hybrid SDRE-index* policy, (4) the *X-index* policy and (5) the *Hybrid X-index* policy. We note that for the *Hybrid X-index* policy, we implement this in the same way as the *Hybrid SDRE-index* policy, namely we set an age limit (based on the performance of the *oldest* policy), and if a request is above the age limit, that medication is produced, and otherwise (all requests are below the age limit) the respective policy is followed.

The key performance measures for each policy are given in Table E.2. The key evidence that demonstrates the robustness of our estimates for the potential improvement of switching to the *SDRE-index* based policies is that the *X-index* policy, which outperformed the

Policy	Average Cost Rate	Average Time in System	Maximum Time in System
Oldest request	37.1	19.0	45.7
SDRE-index	18.8	9.7	403
Hybrid SDRE-index	21.9	11.2	44.9
X-index	68.3	34.9	18601
Hybrid X-index	36.9	18.9	54.7

Table E.2 Summary Performance Measures – Stochastic Simulation

oldest policy in a fluid setting (and its hybrid version with an age limit, in this case 40 hours) does not outperform the *oldest* policy. This provides evidence that even though the *oldest* policy is naive, it is a reasonable benchmark policy to estimate to potential improvement from switching to the $SDRE-index$ based policies. We note for the interested reader that the reason the *X-index* policy performs poorly in the stochastic setting appears to be due to the requests for medications with longer production times. As the *X-index* has the sum of switchover and production times in the denominator, this has a nonlinear effect in lowering the index, resulting in very long delays between when requests for these medications arrive and when these medications are produced. This also illustrates that policies that work well in fluid systems with synthetic data are not guaranteed to work well in stochastic settings.

The subfigures in Figure E.3 depict the average and maximum time in system by medication index for each policy for reference. Note that for the *SDRE-index* and *X-index* policies we use broken y-axis to illustrate the difference in scale of maximum delays for these policies.

F. Data Driven Simulation

This appendix provides the mechanical details and supporting/supplementary results for the data-driven simulation which allows for dynamic and unknown arrival rates that must be estimated using data on past arrivals.

F.1. Mechanical Details

We utilize the pharmacy's actual order flow data, collected over approximately four years, to simulate order sequencing and dispatching. The data includes 67,563 requests spanning 1,192 distinct medications received between April of 2019 and March of 2023. Order information includes the order number, patient ID, prescriber ID, confirmation date, dispatch

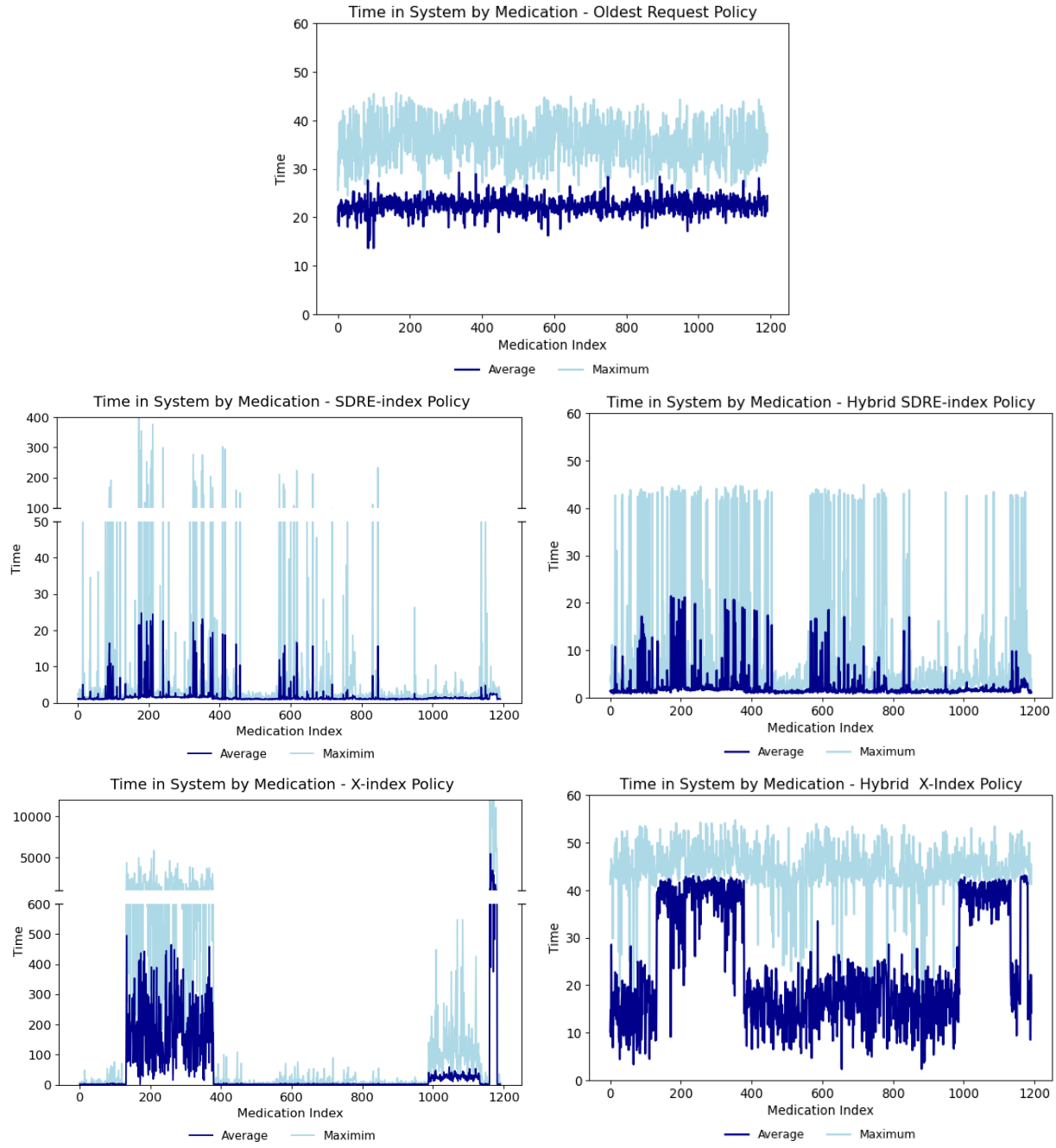


Figure E.3 Performance by Policy

date, dispatch method, ingredients with their respective dosages, and the product form. Product forms include creams, gels, capsules, tablets, pessaries, among 15 possible forms. We process this data to formulate the unique ingredient-dose-product form combinations, and set the changeover and processing times to create the S matrix. Appendix B provides additional details. The estimated arrival rates are computed throughout the simulation as rolling average arrival rates over the past 7 days and are updated every 8 hours. For

Policy	Average Cost Rate	Average Time in System	Maximum Time in System
Oldest	79.5	40.1	149.8
Hybrid SDRE-index	53.7	27.0	152.1
Hybrid X-index	62.0	31.3	164.7

Table F.3 Summary Performance Measures – Data-Driven Simulation

medications that are not ordered (i.e. have estimated arrival rates of zero over the past 7 days) we set their arrival rate to 1 per year.

We simulate order sequencing following the *Oldest request* and the *Hybrid SDRE-index* with the age correction (as described in§5.3). The age limit for the hybrid policy is set at 140 hours and is based on the maximum time in system when following the *Oldest request* policy. We again use a 1 month warm-up periods before we collect performance data and run the simulation until all requests in the order flow have been produced.

F.2. Results

The key performance measures for each policy are given in Table F.3. We note that we forego investigating the base *SDRE-index* and *X-index* policies knowing their shortcomings as it relates to the maximum time spent in the system. We again find the *Hybrid-SDRE-index* based policy offers substantial improvement over the *Oldest* policy. Interestingly the *Hybrid-X-index* policy using estimated arrival rates performs well on the historical order flow – however, given the results with constant and known arrival rates suggests this may be due to the sample path as opposed to indicative of generalizable performance. Determining the drivers/boundaries of performance for the *X-index* heuristic is beyond the scope of this study which is focused on the *SDRE-index* policy, thus we leave further investigation of the *X-index* in this setting for future research.

The subfigures in Figure F.4 depict the average and maximum time in system by medication index for each policy for reference. The subfigures in Figure F.5 parallel those in §5.3 but instead compare the *Hybrid SDRE-index* policy to the *Hybrid X-index* policy.

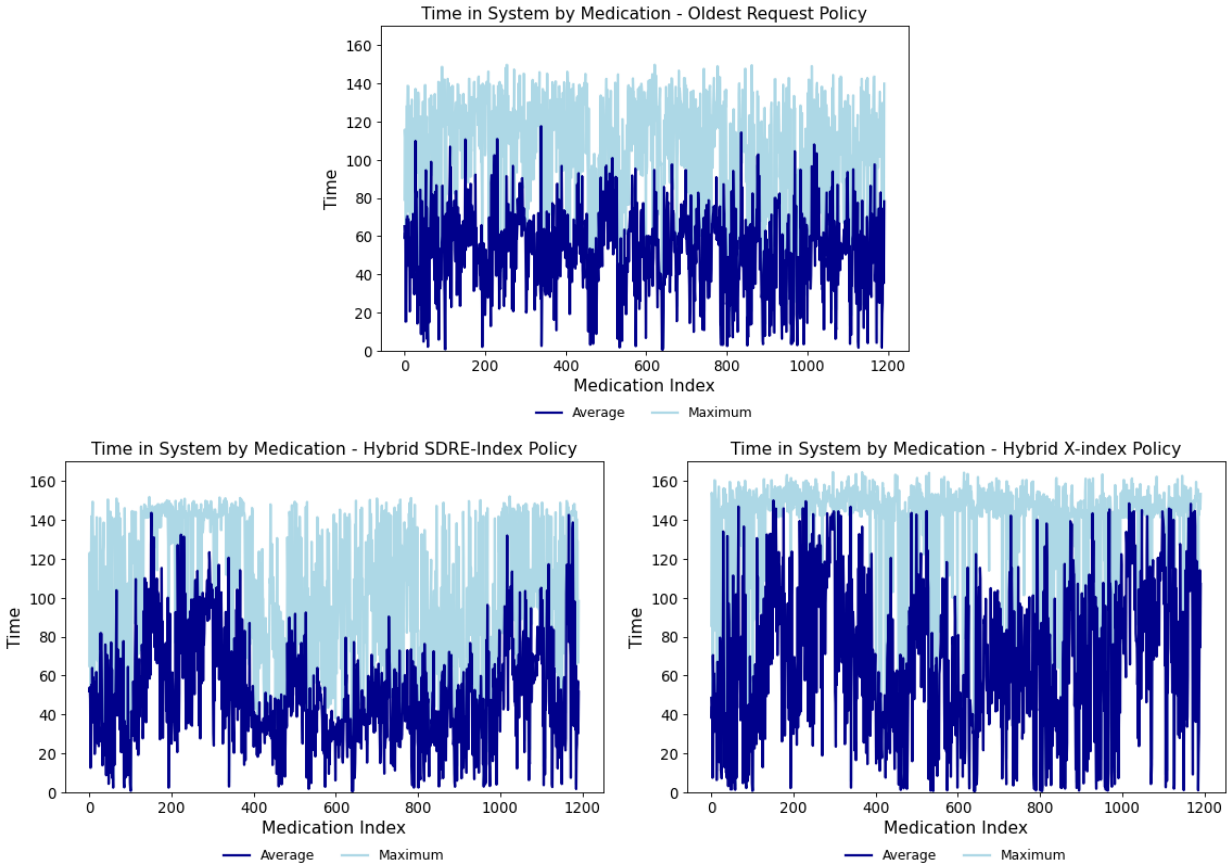


Figure F.4 Performance by Policy

Figure F.5 Policy Comparison – X-Index vs. hybrid-SDRE-Index