Manuscript version: Author’s Accepted Manuscript
The version presented in WRAP is the author’s accepted manuscript and may differ from the published version or Version of Record.

Persistent WRAP URL:
http://wrap.warwick.ac.uk/114605

How to cite:
Please refer to published version for the most recent bibliographic citation information. If a published version is known of, the repository item page linked to above, will contain details on accessing it.

Copyright and reuse:
The Warwick Research Archive Portal (WRAP) makes this work by researchers of the University of Warwick available open access under the following conditions.

© 2019 Elsevier. Licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International http://creativecommons.org/licenses/by-nc-nd/4.0/.

Publisher’s statement:
Please refer to the repository item page, publisher’s statement section, for further information.

For more information, please contact the WRAP Team at: wrap@warwick.ac.uk.
Manuscript version: Author’s Accepted Manuscript
The version presented in WRAP is the author’s accepted manuscript and may differ from the published version or Version of Record.

Persistent WRAP URL:
http://wrap.warwick.ac.uk/114605

How to cite:
Please refer to published version for the most recent bibliographic citation information. If a published version is known of, the repository item page linked to above, will contain details on accessing it.

Copyright and reuse:
The Warwick Research Archive Portal (WRAP) makes this work by researchers of the University of Warwick available open access under the following conditions.

© 2019 Elsevier. Licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International http://creativecommons.org/licenses/by-nc-nd/4.0/.

Publisher’s statement:
Please refer to the repository item page, publisher’s statement section, for further information.

For more information, please contact the WRAP Team at: wrap@warwick.ac.uk

warwick.ac.uk/lib-publications
Dynamic Scheduling of Multi-Product Continuous Biopharmaceutical Facilities: a Hyper-Heuristic Framework

Folarin B. Oyebolu\textsuperscript{a}, Richard Allmendinger\textsuperscript{b}, Suzanne S. Farid\textsuperscript{c}, J"urgen Branke\textsuperscript{a,}\textsuperscript{*}

\textsuperscript{a}Warwick Business School, University of Warwick, Coventry, CV4 7AL, UK
\textsuperscript{b}Alliance Manchester Business School, The University of Manchester, Manchester, M13 9SS, UK
\textsuperscript{c}Department of Biochemical Engineering, University College London, London, WC1E 6BT, UK

Abstract

The biopharmaceutical industry is increasingly interested in moving from batch to semi-continuous manufacturing processes. These continuous bioprocesses are more failure-prone and process failure is more consequential. In addition, the probability of failure is dependent on process run time which generally is determined independent of scheduling considerations. This work presents a discrete-event simulation of continuous bioprocesses in a scheduling environment. Dynamic scheduling policies are investigated to make operational decisions in a multiproduct manufacturing facility and react to process failure events and uncertain demand. First, different scheduling policies are adapted from the stochastic lot sizing literature and a novel look-ahead scheduling policy is proposed. Then, policy parameters (including process run time) are tuned using evolutionary algorithms. Our results demonstrate that the tuned policies perform much better than a policy that estimates policy parameters based on service level considerations and a policy based on a fixed cyclical sequence.

Keywords: Stochastic economic lot scheduling problem, Hyper-heuristics, Biopharmaceutical manufacture, Perfusion, Simulation optimisation, Machine failure

1. Introduction

Biopharmaceutical companies manufacture increasingly sizeable portfolios and face increasing pressure to improve efficiencies by maximising productivity and minimising costs (Farid, 2009; DiMasi et al., 2010) as producing biologics is expensive and time-consuming. Constructing facilities capable of manufacturing multiple products may take half a decade and could cost in the order of $40 to $650 million USD (Farid, 2007). Due to these time and cost pressures, it is essential that the manufacturing schedules are well optimised to make the best use of facility capacity. Manufacturing processes may be placed into one of two groups depending on the mode of operation of the cell culture: fed-batch (batch) or perfusion (continuous). The former is the more established and conventional of the two but of recent there has been increasing interest in the use of the latter. Product material is generally harvested semi-continuously from perfusion processes which contrasts with fed-batch processes where all the material is harvested at the end of the cell culture. As one might imagine, there are process economic trade-offs between these two (Pollock et al., 2013): The continuous processes run for significantly longer than conventional fed-batch processes, may decrease downstream processing (DSP) consumable costs and offer higher daily productivities, which allows for smaller facility footprints. However, they bear higher contamination and equipment failure rates due to the combination of longer cell culture process run times and the increased number and volume of feed additions to the reactor(s) during its operation.

Prior work on scheduling or planning frameworks for bioprocesses has tended to adopt mathematical programming models for fed-batch processes, for example by Lakhdar et al. (2005, 2006, 2007), with less research on perfusion processes (Siganporia et al., 2014). Math-
emathematical programming models under uncertainty have used methods such as chance-constrained programming (Lakhdar et al., 2006) or dealt with uncertainty through scenario analysis (Siganporia et al., 2014). Recently, there have been genetic algorithm (GA) (Holland, 1975) approaches to production planning predominantly for batch processes such as work done in studies by Oyebolu et al. (2017) and Jankauskas et al. (2017, 2019). Dynamic and stochastic simulation models for perfusion culture have focused on capturing the impact of failures and variability on cost of goods rather than on optimal scheduling or planning (Pollock et al., 2013).

This work focuses on investigating optimal scheduling for continuous processes using a dynamic framework. The aim of this work is twofold. The first is to develop a model for perfusion processes that allows manufacturing schedules to be simulated. The second is to adapt and develop dynamic scheduling policies that make operational decisions in a multi-product facility that anticipate and react to changes (such as uncertain demand and process failure events).

To achieve this, a hyper-heuristic is proposed that tunes the parameters of policies tailored to the problem of scheduling multiple perfusion products on a facility. Hyper-heuristics are heuristic search methods that attempt to automate the selection or design of subordinate heuristics to solve hard computational problems (Burke et al., 2013). The distinction between a hyper-heuristic and a meta-heuristic is that the latter searches a solution space (i.e., the search space is comprised of solutions to the problem), however, the former searches within a space of heuristics, rules, or policies. The policy search comprises a simulation optimisation approach which uses an evolutionary algorithm (EA) (Bäck, 1996) as an optimisation algorithm and a custom stochastic bioprocess scheduling model to evaluate performance of candidate policies. The use of scheduling policies allows quick reactions to demand changes and process failure events.

This paper makes several contributions. First, it proposes a new stochastic simulation framework for evaluating operational decisions for a multi-product facility utilising perfusion bioprocesses. It tailors existing policies from the stochastic economic lot scheduling problem (SELSP) literature to the peculiarities of biopharmaceutical manufacturing, including the batch or semi-continuous operation of perfusion processes. In addition, it proposes a novel policy with a custom look-ahead heuristic which enables better performance on the test problem. Finally, process run times are optimised for each product in the portfolio.

This paper is structured as follows. Section 2 provides an overview of related work. Section 3 contains the problem statement and description. A description of the overall hyper-heuristic framework can be found in Section 4 with subsections detailing the bioprocess and discrete-event models, the scheduling policies, and the EAs used for policy parameter tuning. The case study on which the hyper-heuristic is evaluated is laid out in Section 5, which is followed by a report and discussion of the empirical evaluation in Section 6. Finally, the paper concludes with a summary and outlook on future work.

2. Literature Survey

Production planning aims to make best use of production resources in order to satisfy production goals or demand over a planning horizon. As part of this general planning need, there is often the requirement for lot sizing and scheduling, which aims to balance set-up costs and inventory costs. Lot sizing and scheduling as a problem is not new and approaches and solution methods date back many decades. Concurrently, many different variants have emerged which may have various new constraints or applications. This ranges from the simplest variant that considers determining lot sizes for a single item on one machine (Wagner and Whitin, 1958) to more complex problems involving multiple items, parallel machines, and sequence-dependent setup times, among other constraints that reflect the different environments in different industries and instances — see Copil et al. (2017) for a recent classification of the variants.

Historically, most research has been on problems that assume deterministic demand and no randomness or uncertainty in general. However, real-life systems often suffer from uncertainty either in demand, production rates or setup times. This different class of the problem is termed the stochastic lot scheduling problem (SLSP). In their review of the SLSP, Sox et al. (1999) make a distinction between the SELSP and the stochastic capacitated lot sizing problem (SCLSP) to be consistent with their deterministic counterparts — the former assumes continuous time, an infinite horizon, and stationary demand while the latter assumes a finite planning horizon, discrete time periods, and may have non-stationary demand. However, Winands et al. (2011) blur this by defining the SELSP as allowing finite planning horizons but restricting it to stationary demand. In
addition to these surveys, Aloulou et al. (2014) compile an extensive bibliography of publications on the non-deterministic lot-sizing problem and classify them according to the number of products, time-periods, machines, the uncertain parameters, and the modelling approaches. Li and Ierapetritou (2008) review the main methodologies that have been developed to address the problem of uncertainty in production scheduling as well as to identify the main challenges in this area and; Ouelhadj and Petrovic (2009) survey dynamic scheduling in manufacturing systems, covering the limitations of static schedules and approaches in dynamic scheduling.

In general, a production or control policy is required for the SELSP which defines decisions to make in all possible states of the system. These decisions are: whether to continue production of the current product; or switch to another product; or whether to idle the machine. Finite production capacity has to be dynamically allocated between products in order to be responsive to stochastic demands, which adds to the complexity of the problem and means that determining an optimal control policy is non-trivial (Sox et al., 1999). The critical aspects of these policies are the lot-sizing decisions and the sequencing decisions. The lot-sizing decision may either depend on the state of the entire system (a global lot-sizing decision) or just on the stock level of the product currently set up (a local lot-sizing decision). In addition, the production sequence can either be dynamic, fixed with variable cycle length, or fixed with the cycle length fixed as well (Winands et al., 2011).

The SELSP can be formulated as a Semi-Markov Decision Process (SMDP) but this approach does not scale well (Graves, 1980). As a result, simulation optimisation is often used as a solution approach. Simulation optimisation is a powerful technique useful for problems with complex or unknown structure where uncertainty is present (Amaran et al., 2016). Its applications include supply-chain management, inventory replenishment, process design, and bioprocess control (Chu et al., 2015; Jalali and Van Nieuwenhuyse, 2015; Caballero, 2015; Renotte and Vande Wouwer, 2003) with heuristics and meta-heuristics often used as the optimisation algorithm. Recent work in terms of the multi-item SELSP includes the study by Löhndorf and Minner (2013) who formulate the problem as a SMDP and compare different solution approaches including approximate value iteration and global search on simple production policies that had either fixed or dynamic cycles. They find that global control policy search outperforms average value iteration on large problems.

Löhndorf et al. (2014) then extend that work to consider sequence-dependent setup times. Both papers use meta-heuristics to conduct the global search for control policies. Tempelmeier (2013) focuses on discrete time SCLSP models with random demands, fixing production periods and fixing lot sizes under service level constraints. Briskorn et al. (2016) present a fixed cyclic production scheme for multiple products with control strategies to stabilise the cycle length and consider sequence-dependent setup times, backlogging with service level constraints, and limited storage capacity. They use a nested solution approach comprising three levels utilising iterative and neighbourhood search procedures.

An extension to SLSPs looks at production processes which are prone to random machine/equipment failure. In the case of equipment failure, corrective maintenance is done to restore the machine to its ‘normal’ state and any imperfect product items are either re-worked or discarded. Also, preventive maintenance may be carried out in order to mitigate the occurrence of failure events. Nourelatif (2011) determines robust production plans for the SCLSP to ensure that specified service level is met with high probability. The model accounted for random machine breakdowns and random repair times independent of product type and lot size. It did not consider random demand nor preventive maintenance planning. On the other hand, Purohit and Kumar Lad (2016) present a mathematical model incorporating job sequencing, lot sizing, and a schedule for preventative maintenance which is solved with the use of a simulation-based GA approach and outperforms previous conventional approaches.

Though these works are noteworthy, they are not exactly transferable to the problem considered in this paper. This is because the prior work has preventative maintenance scheduling as an additional separate decision variable. However, in this paper, lot-sizing and preventative maintenance are intrinsically linked and therefore inseparable. Equipment failure in this paper is an increasing function of process run-time and therefore lot-size. So a decision on lot-size may be seen as implicitly scheduling preventative maintenance since the process (and time to the next failure) is ‘reset’ at the end of each lot or batch.

The use of hyper-heuristics in generating or designing heuristics for production scheduling is surveyed by Branke et al. (2016). In an indirect GA representation proposed by Kimms (1999), a two-dimensional matrix is used as chromosome, with each entry representing a
rule for selecting the set up state for a machine at the end of a period. Since the entries in the chromosome represent a rule for selecting set up states, the approach can be seen as a selection hyper-heuristic as the search space is on potential rules and not direct solutions to the problem. Similarly, the previously described approach of Löhndorf and Minner (2013) and Löhndorf et al. (2014) can be classed as a hyper-heuristic as there is a global search for control policies. Hyper-heuristics may also incorporate machine-learning techniques such as Artificial Neural Networks (ANNs) (Haykin, 1994) or genetic programming (GP) (Koza, 1992). For example, Burke et al. (2007) have demonstrated automated heuristic generation with GP. For a complex dynamic scheduling problem, Pickardt et al. (2013) proposed a two-stage hyper-heuristic using GP for the generation of work centre-specific dispatching rules. Branke et al. (2015) investigate three different rule representations for optimising rules to compute priority indices for new/arriving jobs in a jobshop environment, namely: linear representation, a feed-forward ANN, and GP with tree-representation.

Literature in the biopharmaceutical industry on planning and production scheduling is scarce but growing (Vieira et al., 2015). However, the models used are either deterministic, do not model perfusion processes, or do not focus on scheduling. For example, Lakhdar et al. (2005) developed a deterministic mixed-integer linear program for the planning and scheduling of a multi-product biopharmaceutical manufacturing facility utilising fed-batch processes and later extended it for use with a multi-facility model where multiple criteria were considered using goal programming (Lakhdar et al., 2007). Siganporia et al. (2014) also develop a mixed-integer linear programming (MILP) model, in this case to optimise an eight-year planning horizon for a mixture of fed-batch and continuous bioprocesses while considering capacity decisions in a few scenarios with different demands and bioreactor titres. They utilised a rolling time horizon to improve computational performance. Recently, the work done by Lakhdar et al. (2005) has been extended to alternative approaches by other authors. First, Vieira et al. (2016) solved a set of example problems based on a Resource Task Network (RTN) continuous-time single-grid formulation focusing on addressing specific operational characteristics of bioprocesses. Jankauskas et al. (2017) then used a continuous-time model optimised by a GA which is underpinned by a dynamic chromosome structure (i.e., vector of decision variables) that is allowed to vary in length. Also, Gatica et al. (2003) and Levis and Papageorgiou (2004) present a mathematical programming approach for the capacity planning problem with a focus on long-term planning and capacity investment decisions under uncertainty of clinical trials rather than scheduling. Marques et al. (2017) present a simulation optimisation approach combining a MILP model and Monte Carlo simulation procedure to integrate process design and planning decisions under clinical trial and demand uncertainty for the pharmaceutical industry. Finally, Pollock et al. (2013) developed a model focused on investigating the economic benefits of continuous perfusion culture and single-use technology for a monoclonal antibody (mAb). As part of this evaluation, stochastic process failure events and their consequences are considered using simulation. This is then extended by Pollock et al. (2017) to include an assessment of various integrated continuous process flowsheets.

3. Problem Domain Description

In this work, characteristics specific to bioprocessing and biopharmaceutical manufacturing are addressed using a simulation optimisation approach. Novel to bioprocessing literature, we optimise the schedule of a facility manufacturing multiple drug products utilising a perfusion cell culture process. This is while incorporating uncertainty in demand and random equipment failure, and simulating any disruptions these stochastic events have on the manufacturing schedule and planning environment as a whole. This necessitates the development of a custom discrete-event simulation model coupled with an optimisation algorithm. For the optimisation algorithm, we use an EA to search for optimal production control policies. As we search the space of possible heuristics or rules as opposed to possible solutions, it constitutes a hyper-heuristic approach.

The problem that is considered by this paper is a variant of the SELSP applied to biopharmaceutical manufacturing. It involves a facility and a set of drug products \(PF\), each associated with a bioprocess which, when operated, manufactures the corresponding product. The state of the facility, \(m\), refers to the product, \(p\), whose bioprocess is currently in operation on the facility or is 0 (zero) if idle. A facility refers to the set of equipment suites available to operate a bioprocess. In the case of two separate, segregated, and independent sets of equipment suites, these would be classed as two separate facilities. The bioprocesses are comprised of
the same multiple stages (unit operations) and are operated in a semi-continuous manner — the bioprocess has to be operated in multiples of a pre-defined batch length (i.e., process duration or run-time) but processed material is made available over the course of the batch and not just at the end. In general, no more than one bioprocess may be in operation in the facility at any time to avoid cross-contamination or product-carryover issues and to ease the validation burden, though there is some exception to this which allows the earlier stages of a subsequent process to be in operation simultaneously with the latter stages of the previous process.

In addition to stationary stochastic demand, the manufacturing process is prone to equipment failure of differing types, risks, and consequences. Demand may be backlogged by meeting unfulfilled demand in one period with production from future periods, but will accrue a backlog penalty. There is also a daily backlog decay on unfulfilled demand — i.e., a certain percentage of backlogged demand is lost every day. The simulation runs over a finite time horizon and discrete-time periods of one day each.

The objective is to maximize the overall profit, calculated as total revenue minus the costs for production, storage, process changeover, wastage, and backlog penalties given a facility with different manufacturing yields and manufacturing costs for the different products.

4. Modelling Framework

The hyper-heuristic proposed in this paper is designed as a custom framework comprising a discrete-event model used to simulate the scheduling environment on the manufacturing facility in which the bioprocesses are operated, policies that dictate scheduling decisions, and optimisation algorithms to tune the scheduling policy parameters. Figure 1 gives an overview of this framework and the interactions between its components. The optimisation algorithm passes policy parameters to the simulation framework which interfaces directly with the bioprocess model and the logic of the scheduling policies. The simulation framework can then return expected profit as the fitness measure for
the policy parameters selected by the optimisation algorithm. This section describes the components of the hyper-heuristic framework and their interactions.

4.1. Bioprocess Model

The bioprocess model utilised by the framework in this paper is defined by the flowsheet and configuration of the manufacturing process, the associated process economics, timings of the unit operations, and the risk and consequences of failure events.

4.1.1. Manufacturing process

The basis of the bioprocess used is a platform mAb manufacturing process utilising a perfusion bioreactor with an alternating tangential flow (ATF) filtration system for cell retention. Perfusion bioreactor systems equipped with an ATF filter have been shown to perform well in economic analyses compared to other cell-retention filter systems and do not suffer consequences as severe in the event of filter failure (Pollock et al., 2013). A flow sheet of this process is shown in Figure 2.

For simplicity, the unit operations of the process can be grouped together into three main steps:

- A seed train which encompasses all cell thawing and expansion operations;
- the upstream processing (USP) which is just the cell culture; and
- the downstream processing (DSP) which accounts for all unit operations from the capture chromatography step (Protein A) to the final finish & polishing steps (UF/DF).

The approach could easily be adapted to different process configurations but this work uses only scenarios with one bioreactor to each DSP train, i.e., a 1:1 USP:DSP train configuration.

4.1.2. Process timing

The seed train takes 14 days and then the production bioreactor can be inoculated. The ramp-up time to reach the desired cell density for harvests is ten days. In this period, no harvests are collected as the process has not yet reached steady-state. From day 11 onwards, daily harvests are collected from the bioreactor and then taken through DSP. Each DSP ‘batch’ then takes two days to be fully processed and material coming out of it then can be put in inventory, sold or otherwise delivered. Therefore, a process that has a cell culture run time of 60 days and ramp-up time of 10 days will produce 50 separate DSP batches. The turnaround on the bioreactor determining the earliest time it can be reused is four days and accounts for cleaning-in-place (CIP) and sterilization-in-place (SIP) operations and other activities in preparation for a new cell-culture operation. Turnaround time is the minimum time between two USP operations of the same product. Changeover between different products takes longer than turnaround.
As our study is looking at various process durations, ATF filter failure events (2%) were within the 60 days. As it is possible for parts of the seed train process to use a different suite, the earliest portion of the seed train of a product can take place concurrently with the latter days of the USP of a previous batch (of the same or a different product). We can then define a threshold, as the minimum time elapsed for a USP operation before a decision on starting a new batch (i.e., a new seed train) can be made. The seed-restart threshold is for starting a new batch of the same product and is the sum of its cell culture run-time and turnaround time minus its seed run time. The changeover threshold is for starting a new batch of a different product and is equal to cell culture run-time plus changeover time minus seed train run time of the subsequent product. Process timing information is contained in Table 1.

4.1.3. Process economics

The economics associated with this process are such that a seed train cost is attributed to every seed operation that is started, and a cell culture setup cost for the setup and prep activities that go into starting up each bioreactor in a batch. The daily cell culture perfusion costs are accrued for every day a bioreactor is in operation, and for every DSP batch commenced, DSP batch cost is accrued. A cost is associated with replacing a fouled ATF filter; this is also captured in the batch setup costs as a new ATF filter is needed for each one. Finally, if the process is idle for more than the setup expiry period, there is a cost of re-establishing sterile and clean holds for all equipment before another batch of the same product is started. This is the same as the changeover costs of setting up the facility when changeover to manufacturing another product occurs.

4.1.4. Process failure

To capture the stochastic failure events and the consequences, previous data adapted from Pollock et al. (2013) was used and this is presented in Table 2. That study used a fixed perfusion duration of 60 days so the probability of ATF culture contamination (6%) and ATF filter failure events (2%) were within the 60 days. As our study is looking at various process durations, this requires some adjusting. We believe that the rate of failure should be low in the early stages of the process and be relatively high towards the end. The process for Pollock et al. choosing a 6% failure rate for the cell culture contamination is based on the assumption that each addition to (or sample from) the bioreactor has a 1 in 1000 chance of introducing contamination to the system; the 60 day batch had approximately sixty such additions leading to the 6% chance of failure. As a result, the chance of failure on any specific day is independent of how far along in the process it currently is. For the equipment failure due to filter fouling, however, Pollock et al. choose a probability of 2% and weight failure to occur at latter stages of the cell culture. We reason that not only additions can cause contamination but equipment that wears or stresses over the course of the process (such as tubing, gaskets, valves, O-rings, filters, seals or connectors). So we deviate from Pollock et al. and use an exponential function of the form shown below to describe the probability of a failure event occurring $P(x)$, on a specific day $x$:

$$P(x) = \frac{\exp(x/a) - 1}{b}$$  \hspace{1cm} (1)

Here, $a$ is benchmarked to 60 and also represents the amount of time it takes (in days) for the probability of failure to increase by a factor of $e$; and $b$ is a scaling constant.

The difference in assumption is illustrated in Figure 3 which shows the cumulative probability of a cell culture contamination event occurring within the duration of a process. For the profile (used by Pollock et al.) where the daily absolute risk of failure is constant, the cumulative probability of failure in the early parts of the process can be significantly larger than for the profile used in this paper that is modelled from an exponential function.

Primarily, just two failure rates were used: 2%, and 10% — the former for the ATF filter failure and the latter for culture contamination. These are defined using Equation (1), as the probability of failure event occurring within the first 60 days unless otherwise specified.

4.2. Discrete-Event Simulation Framework

Based on the bioprocess model discussed in Section 4.1, a discrete-event simulation model was developed in Java™. The model simulates the processing of batches on a facility as a multi-stage process comprising of a seed train, USP, and DSP. In addition, based on the state of the facility (i.e., what it is currently manufactured), inventory levels, and stochastic events, it evaluates the economics of operational decisions and reports key metrics, inventory profiles, and the facility schedule of the given time horizon. Finally, it allows the use and evaluation of dynamic scheduling policies.
Table 2: Process failure events, consequences and the associated risk (adapted from Pollock et al. (2013)).

<table>
<thead>
<tr>
<th>Process Event</th>
<th>$p$(Failure)</th>
<th>Consequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATF culture contamination</td>
<td>10%</td>
<td>Batch loss &amp; discard two pooled perfusate volumes</td>
</tr>
<tr>
<td>ATF filter failure</td>
<td>2%</td>
<td>Replace filter &amp; discard next 24 hours of perfusate</td>
</tr>
</tbody>
</table>

Figure 3: The cumulative probabilities showing how likely the USP operation of a certain duration will fail based on different failure rates. Failure profile with a constant daily failure risk used in Pollock et al. (2013): 1/1000 daily rate. Failure profile with a variable daily failure risk used in this paper: 10% in 60 days.
to make scheduling decisions.

On each day:

- any new activities or operations are started if required whilst any existing ones have their durations advanced by a day;
- any manufacturing takes place and the random variables (such as demand, process failure events, or yield) are realised by sampling their associated probability distributions;
- the process economics related to the model are evaluated;
- scheduling decisions are made if necessary; and
- activities or operations are brought to an end at their target run time (or terminated early due to equipment failure).

Scheduling decisions are made near the end of a batch (at seed-restart or changeover thresholds), in the event of batch failure, or if the facility is idle. The set of decisions available to make is limited to deciding whether to keep the facility idle or not, and if the latter, which product should the new batch produce. Decisions made on one day are implemented and take effect the following day.

In the event a decision is made to start a batch and/or campaign, a seed train operation is created and started up. When that is completed, it triggers the start of a USP operation which proceeds until it reaches its intended end — specified by the process run time — or is terminated because of a process failure event. During the course of USP but past its ramp-up, the daily harvests trigger separate DSP operations which deposit product in the inventory when completed. Over the course of the simulation, all these operations are recorded so that at the end of the simulation, a facility schedule can be generated describing the history of operational decisions taken, the inventory levels for each product, the workload of the facility, and identifying any batches terminated early due to contamination.

Depending on the assumptions made for the purposes of the simulation, demand may be set to be yearly, monthly, or daily. Any available product in inventory is used to satisfy the demand for that period. If the inventory is not sufficient, unfulfilled demand is added to backlog on which a decay function is applied. Any product that has exceeded its shelf-life is deleted from inventory and discarded.

4.3. Scheduling Policies

The scheduling strategies employed in a dynamic simulation environment are based on control policies that initiate new production orders (batches in this case) based on current inventory levels as well as the state of the facility (i.e., the product currently being manufactured), \( m \), in a make-to-stock fashion.

Three base-stock policies (BSPs) are presented here, two of which are adapted from Löhndorf and Minner (2013), and the third is a novel development. In addition, two benchmark policies are described here. The adaptations from the versions presented in previous work are necessary because in our paper, processes can fail and consist of a fixed sequence of multiple batch (and semi-continuous) unit processes. In addition, there is a need to identify an optimum run time for one of the constituent unit processes — the perfusion cell culture in this case. Since the processes here are composed of multiple stages, the decision to make the next batch has to be made before the current batch is over. This means that the amount of product made and delivered in that time lag needs to be estimated and taken into account.

In our case, due to the multi-stage process, the time-period in which a decision can be made (decision epoch) begins at the seed-restart or changeover thresholds. Decisions can also be made once the cell culture is contaminated and fails or when the facility is idle. As a result of the lag between when a decision is required and the end of the current batch, the policies presented here will take into account the expected extra material that will have been produced by the end of the current batch (if the facility is not idle). So, \( \gamma_{pt} \) is set to be the estimated net production (i.e., material produced minus expected demand) in the interval between the decision-making time point and the end of the batch. Decision epochs end when a decision to start a batch has been made.

4.3.1. Simple base-stock policy (BSP1)

This is the first of the policies adapted from Löhndorf and Minner (2013). In this case, there are three policy parameters defined:

1. The reorder point, \( Y^{(1)} \)
2. The order-up-to level, \( Y^{(2)} \)
3. The process run time of batches in the campaign, \( B \)
Algorithm 1 Pseudocode of BSP1.

1: procedure BSP1(current time $t$)
2:     $PF =$ the set of products manufactured
3:     $m_t =$ the product manufactured at time $t$, 0 if idle
4:     $I_{pt} =$ the inventory level of product $p$ at time $t$
5:     $\gamma_{pt} =$ the estimated net output of $p$ between $t$
6:     and end of its batch
7:     $\mu_p =$ the expected demand of product $p$
8:     $Z_t = \{p : I_{pt} \leq Y_p^{(1)}\} \forall p \in PF$
9:     if $m_t > 0$ AND $I_{mt} + \gamma_{mt} < Y_m^{(2)}$ then
10:         Start new seed train of product $m_t$
11:     else if $Z_t^{(1)} \neq \emptyset$ then
12:         Start new seed train of product
13:         $\arg\min_{p \in Z_t} [I_{pt}/\mu_p]$
14:     else if $\forall p : I_{pt} > Y_p^{(1)}$ then
15:         Keep facility idle
16:     end if
17: end procedure

The logic for determining when to start a new seed train is detailed in Algorithm 1. This states that if the facility is idle and there is a product $p$ with inventory below inventory policy parameter $Y^{(1)}$, a new batch — with a cell culture run time of $B$ days — for this product is initiated. If there is more than one product to which this applies, the product selected to be manufactured next is the one with the smallest run-out time — i.e., time until product runs out of stock, $I_{pt}/\mu_p$, where $I_{pt}$ is inventory of product $p$ and $\mu_p$ its expected demand. Once the facility is manufacturing a product, it does not go idle or switch products (changeover) until it has reached inventory policy parameter $Y^{(2)}$.

4.3.2. Can-order base-stock policy (BSP2)

This policy was designed by Löhndorf and Minner (2013) to be an improvement on BSP1 by including two more policy parameters. This allows the control policy to interrupt a campaign (a set of consecutive batches of the same product) if the inventory of another product falls critically low. The policy parameters are identified below:

1. The reorder point, $Y^{(1)}$
2. The order-up-to level, $Y^{(2)}$
3. The can-order point, $Y^{(3)}$
4. The can-order-up-to level, $Y^{(4)}$

Algorithm 2 Pseudocode of BSP2.

1: procedure BSP2(current time $t$)
2:     $PF =$ the set of products manufactured
3:     $m_t =$ the product manufactured at time $t$, 0 if idle
4:     $I_{pt} =$ the inventory level of product $p$ at time $t$
5:     $\gamma_{pt} =$ the estimated net output of $p$ between $t$
6:     and end of its batch
7:     $\mu_p =$ the expected demand of product $p$
8:     $Z_t^{(1)} = \{p : I_{pt} \leq Y_p^{(1)}\} \forall p \in PF$
9:     if $m_t > 0$ AND $I_{mt} + \gamma_{mt} < Y_m^{(2)}$ then
10:         Start new seed train of product $m_t$
11:     else if $Z_t^{(1)} \neq \emptyset$ then
12:         Start new seed train of product
13:         $\arg\min_{p \in Z_t^{(1)}} [I_{pt}/\mu_p]$
14:     else if $\forall p : I_{pt} > Y_p^{(1)}$ then
15:         Keep facility idle
16:     end if
17: end procedure

5. The process run time of batches in the campaign, $B$

The logic for determining when to start a new seed train is detailed in Algorithm 2. If the facility is idle and there is a product $p$ with inventory below $Y^{(3)}$, a new batch for this product is started. The campaign cannot be interrupted until the inventory of that product exceeds $Y^{(4)}$. When this occurs, changeover to another product is allowed if its inventory level is less than $Y^{(1)}$ — again any ties are settled by picking the product with the smallest run-out time. However, if there are no products with inventory below their reorder points, the campaign may continue until it exceeds $Y^{(2)}$.

So the facility does not go idle until all products are above their can-order points. In this manner, this policy works similarly to BSP1 but with the can-order point and the can-order-up-to level it enables interruptions. In general, $Y^{(1)} \leq Y^{(3)} \leq Y^{(4)} \leq Y^{(2)}$ and BSP1 can be considered a special case of BSP2 where the inventory policy parameters are such that $Y^{(1)} = Y^{(3)}$ and $Y^{(4)} = Y^{(2)}$. The disadvantage of this policy is that with more
Algorithm 3 Pseudocode of BSP3.

Procedure \text{BSP3}(\text{current time } t)

1. \text{PF} = \{\text{products manufactured}\}
2. \text{m}_t = \{\text{product manufactured at time } t\}$ if idle
3. \text{I}_t = \{\text{inventory level of product } p \text{ at time } t\}
4. \text{CE} = \{\text{expected net output of } p \text{ between } t\}$ and end of its batch
5. \text{µ}_p = \{\text{expected demand of product } p \text{ during the decision epoch}\}
6. \text{Z}^{(5)} = \{\text{all permutations of products} \}
7. \text{π} = \{\text{estimates the cost of each permutation}\}
8. \text{γ} = \{\text{reorder point during a decision epoch} \}
9. \text{if } \text{π} \neq \{\text{then}\}
10. \text{Select cheapest permutation, } \text{π}^* = \text{arg min}_{\pi \in \text{π}} \text{CE}(\pi)
11. \text{Start new seed train of product } \text{π}^*_i
12. \text{else if } \forall p: \text{I}_t + \text{γ}_p > \text{Y}_p \text{ then}\)
13. \text{Keep facility idle}
14. \text{end if}
15. \text{end procedure}

This policy estimates the costs of each permutation. This estimated cost is the projected sum of inventory costs, backlog penalties, and any changeover costs assuming that the processes will be run without failure and the demand realised is equal to the mean of its probability distribution — other manufacturing costs will be the same regardless of the manufacturing permutation. Cost estimation is done by generating a function describing the piecewise linear estimation of each product’s inventory before, during, and after production of its corresponding batch within the permutation time frame. The inventory cost for each product is the product of the inventory rate and the absolute value of the sum of positive integrals of the function; and the backlog penalty is the product of the backlog penalty cost and the absolute value of the sum of negative integrals of the function. The permutation of batches that has the lowest estimated cost is selected and a batch of the first product in the permutation is started.

4.3.3. Forecasting base-stock policy (BSP3)

This policy is a novel contribution that this paper is proposing and utilizes a ‘look-ahead’ heuristic. The following policy parameters are identified:

1. The reorder point, \text{Y}_p^{(5)}
2. The process run time of batches in the campaign, \text{B}

When the inventory of a product falls below the reorder point during a decision epoch (i.e., during idle time, or after a batch ends or is contaminated), a new batch of that product is started with cell culture run time of \text{B}. If there is more than one product with inventory below \text{Y}_p^{(5)}, the heuristic first generates all possible permutations of manufacturing a single batch of each of the products with inventory less than \text{Y}_p^{(5)}. Next, the heuristic estimates the costs of each permutation. This
of a common cycle time. The method to calculate these values is as follows: Let $\hat{T}$ be the common cycle time and $k$ be a safety factor to mitigate stock-outs.

$$\hat{T} = \max \left\{ \frac{2 \sum_{p \in PF} A_p}{\sum_{p \in PF} \rho_p \mu_p^d (1 - \mu_p^d/(br_p \gamma_d)),} \right\}$$

(2)

$$1 - \frac{\sum_{p \in PF} \alpha_p}{\rho_p} \right\}$$

(3)

Where $\mu_p^d$ is the expected daily demand; $\sigma_p^d$ is the standard deviation of the daily demand; $\alpha_p$ is the process lead time which in this case is the sum of the seed train time, the ramp-up time, and the time of one DSP batch; $br_p$ is the amount harvested from the daily perfusate of the bioreactor; $yd_p$ is the process yield; $\delta_p$ is the lost demand penalty cost; $\theta$ is the daily backlog decay rate; $\rho_p$ is the inventory holding cost; and $A_p$ represents all setup costs associated with the first batch of a campaign prior to its first perfusion harvest, i.e., the sum of the changeover cost, seed cost, cell culture setup cost, and the perfusion costs in the ramp-up period.

Then, the reorder level and order-up-to point are, for each product $p$, set to:

$$Y_p^{(1)} = \max \left\{ \mu_p^d \alpha_p + k_p \sqrt{\sigma_p^2 \hat{T}}, 0 \right\}$$

(4)

$$Y_p^{(2)} = Y_p^{(1)} + \max \left\{ \mu_p^d (1 - \mu_p^d/(br_p \gamma_d)))\hat{T}, 1 \right\}$$

(5)

In this benchmark policy, the process run time for each product is set to 60 days.

A fixed cycle policy. The second benchmark policy is based on a fixed cycle. This policy, denoted as FCP, ensures that the manufacturing facility follows a fixed sequence of batches. The policy parameters for this policy are the sequence and the process run times. This policy is implemented by cycling through each element in the sequence, producing a batch of the product the item corresponds to, and then moving to the next element in the sequence — when it gets to the end of the sequence, it starts from the beginning again. Each element in the sequence can be one of any $p$ in $PF$ which indicates that a single batch of product $p$ is to be manufactured. In addition, any element in the sequence may be zero (0) which indicates that the facility should be made idle. Consecutive identical but non-zero elements in the sequence means that consecutive batches of that product are produced. However, consecutive zeroes in the sequence are pruned down to one instance which means that sequences can be of variable length. A sequence with a leading and trailing zero will have the leading element preserved and the last element deleted. The time the facility spends being idle is determined by the inventory levels of the products. Specifically, idle time is ended when any product’s run-out time falls below 90 days. At that point, the facility will then move on to the next product in the sequence. Also, if a batch ends prematurely due to process failure, the facility will move to the next element in the sequence.

4.4. Optimisation Algorithms

Evolutionary algorithms were used to optimise the policy parameters of the scheduling policies implemented. To be specific, we compare the performance of a genetic algorithm (GA) and a Covariance Matrix Adaptation Evolution Strategy (CMA-ES) for the optimisation of the BSP policy parameters. These were implemented in Java using the ECJ Library (Version 24) (Luke, 1998).

The genomes (chromosomes) used to represent candidate policies were designed to deal with the constraints on policy parameters. For example, a BSP1 policy needs to have its policy parameters such that $Y_p^{(1)} \leq Y_p^{(2)}$ and a BSP2 policy needs its policy parameters so $Y_p^{(1)} \leq Y_p^{(3)} \leq Y_p^{(4)} \leq Y_p^{(2)}$. This is illustrated in Figure 4, where instead of the genome representing the inventory policy parameters directly, the difference between adjacent inventory policy parameters is encoded. The ranges for the genes encoding the policy parameters are detailed in Table 3. The fitness of a candidate solution is determined by evaluating multiple simulations of the solution’s policy parameters using the discrete-event framework.

### Table 3: The minimum and maximum values for the genes in the chromosome.

<table>
<thead>
<tr>
<th>Gene</th>
<th>Type</th>
<th>Min value</th>
<th>Max value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Y_p^{(1)}$, $Y_p^{(2)}$</td>
<td>continuous</td>
<td>0</td>
<td>60</td>
</tr>
<tr>
<td>$B$</td>
<td>integer</td>
<td>14</td>
<td>120</td>
</tr>
</tbody>
</table>
Figure 4: Structure of a BSP2 policy chromosome for a facility manufacturing three drug products where the process run times are simultaneously optimised.

All results reported are an average of 50 runs unless otherwise stated. Below, the parameters of the evolutionary algorithms deployed are detailed.

4.4.1. The Genetic Algorithm

A GA was designed to optimise the policy parameters in each BSP strategy and the GA parameters are as follows: The number of generations was set to be 200, the population size was 30. The elitism implemented was such that the fittest six individuals in the population each generation are carried over to the next generation automatically. The chromosome was determined by the type of BSP and the number of products. So, if the policy did not simultaneously set process duration, the length of the chromosome would be the number of inventory policy parameters times the number of products. In the case where the policy also set the process duration, then additional genes for each product would be added to the chromosome. The first segment of the chromosome representing the inventory policy parameters for each product were real-valued whilst the last few genes representing the process duration were integer (see Figure 4).

The selection process for individuals to be crossed over was a tournament (with replacement) of size two. The probability of crossover being applied was 0.9 and the crossover operator was uniform crossover. Probability of a gene being mutated was the value of the inverse of the chromosome length (i.e., 1/chromosome length). Gaussian mutation was used (with a standard deviation of 6) for the real-valued genes whilst the integer genes used random walk. Random walk mutation performs a random walk starting at the current gene value. At each step in the random walk it adds either +1 or −1 to the current variable value. Then with a probability of 0.9 it iterates another step, else it quits the walk.

FCP is also tuned using a GA with a chromosome encoding a sequence of at most 12 elements and process run times (if these are allowed to be optimised) — all the genes are therefore integers. The GA is identical to the one previously outlined, except that the crossover operator was two-point crossover and to mutate the sequence segment of the chromosome, a gene is removed and randomly re-inserted somewhere else in the sequence.

4.4.2. The Covariance Matrix Adaptation Evolution Strategy

The CMA-ES does not have many user-specified parameters, as a lot of them are calculated based on the chromosome specified. As a result the default settings were used (Hansen and Ostermeier, 2001). The chromosome has the same structure as previously discussed but with the values of the encoded genes normalised to fall between [−1, +1]. These values are then transformed to the actual policy parameters at the point of fitness.
evaluation. The transformation functions are such that to obtain the inventory policy parameter \( Y \) (i.e., \( Y_{(1)} \) or \( Y_{(2)} \)) from the encoded value \( Y' \) the following equation was used:

\[
Y = 30Y' + 30 \quad (6)
\]

To get \( B \) from its encoding \( B' \) the equation below is used, where \( \text{Round}(x) \) rounds \( x \) up:

\[
B = \min \left\{ \text{Round}(53.5B' + 67), 120 \right\} \quad (7)
\]

The CMA-ES was given the same budget as the GA (6, 000 fitness evaluations) and the starting point for the search was set randomly in the decision space.

### 5. Case Study Description

To evaluate our proposed method, we designed a biopharmaceutical industrial case study. The data comprises anticipated market demand and manufacturing facility characteristics. This problem features multiple products to be produced on a single facility with different efficiencies, yields and costs, perishable inventory, and the ability to backlog demand. The processes to manufacture these products are all based on a platform mAb process as described in Section 4.1.

The demand forecast is made up of three different constituents: \( D \) is the annual forecasted demand and \( p_{t} \) is the observed demand of product \( p \) at time \( t \) so that

\[
\Delta_{pt} = \max \left\{ 0, \theta \Delta_{p,t-1} + D_{pt} - S_{pt} \right\} \quad \forall \, p, t \quad (8)
\]

The case study is designed so that each of the three products \( (p1 - p3) \) to be manufactured over the period of seven years (with a year comprising 360 days). The demand forecast shows the expected annual demand which is stationary (i.e., does not change from year to year) but is stochastic (see Table 4). The stochastic demand for each product follows a Normal distribution, \( N(\mu, \sigma^2) \), where \( \mu \) is the annual forecasted demand and \( \sigma \) is 0.025\( \mu \). The demand frequency is set to be daily which means that demand is sampled, is due, and can be delivered every day; sampled demand is truncated and not allowed to be negative. This means the distribution describing the daily demand is \( N(\mu^D, (\sigma^D)^2) \) where \( \mu^D = \frac{\mu}{360} \) and \( \sigma^D = \frac{\sigma}{\sqrt{360}} \). The different constituents of the manufacturing costs are also listed in Table 4 as well as the reactor yields, the sales price for each kilogram of each product, and the periodic penalty cost for each unit of unfulfilled demand. The reactor yield indicates how much product is in each daily harvest from the bioreactor. The product of this and the overall process yield is how much material is deposited in inventory after each DSP batch. For all products, the overall process yield is 69%.

<table>
<thead>
<tr>
<th>Table 4: Process economics parameters in relative monetary units (RMU) unless otherwise stated.</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p1 )</td>
</tr>
<tr>
<td>Seed cost</td>
</tr>
<tr>
<td>Daily cell culture cost</td>
</tr>
<tr>
<td>Cell culture batch setup cost</td>
</tr>
<tr>
<td>ATF replacement cost</td>
</tr>
<tr>
<td>DSP batch cost</td>
</tr>
<tr>
<td>Sales price (RMU/kg)</td>
</tr>
<tr>
<td>Backlog penalty cost (RMU/kg/day)</td>
</tr>
<tr>
<td>Annual demand (kg)</td>
</tr>
<tr>
<td>Reactor yield (kg)</td>
</tr>
</tbody>
</table>

Changeover cost is accrued when switching between two different products or after the setup expiry period lapses between batches of the same product. Changeover and turnaround times are defined as the minimum time between two USP operations for different and same product respectively. Equation (8) describes the daily backlog rate, where for product \( p \): \( \Delta_{pt} \) is the amount of product \( p \) that is late at time \( t \), \( \theta \) is the daily backlog decay rate, \( D_{pt} \) is the observed demand of product \( p \) at time \( t \), and \( S_{pt} \) is the amount of product \( p \) that is sold at time \( t \).

\[
\Delta_{pt} = \max \left\{ 0, \theta \Delta_{p,t-1} + D_{pt} - S_{pt} \right\} \quad \forall \, p, t
\]
Table 5: Case study parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shelf life</td>
<td>720</td>
<td>Days</td>
</tr>
<tr>
<td>Backlog decay</td>
<td>0.5</td>
<td>Per 180 days</td>
</tr>
<tr>
<td>Inventory rate</td>
<td>0.01</td>
<td>RMU/kg/day</td>
</tr>
<tr>
<td>Bioreactor turnaround</td>
<td>4</td>
<td>Days</td>
</tr>
<tr>
<td>Changeover time</td>
<td>10</td>
<td>Days</td>
</tr>
<tr>
<td>Wastage rate</td>
<td>5</td>
<td>RMU/kg</td>
</tr>
<tr>
<td>Setup expiry</td>
<td>30</td>
<td>Days</td>
</tr>
<tr>
<td>Planning period</td>
<td>7</td>
<td>Years</td>
</tr>
<tr>
<td>Number of reactors</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Changeover cost</td>
<td>35</td>
<td>RMU</td>
</tr>
</tbody>
</table>

the former commands a higher sales price per kilogram manufactured but also a larger penalty per kg of unfulfilled demand. On the other hand, comparing $p_3$ to $p_2$, we see that $p_3$ has a similar demand profile as $p_2$ but its yield is a third lower.

This case study will be the basis of simulation optimisation experiments reported in the following section.

6. Results and Discussion

To evaluate the performance of the policies, optimisation runs were carried out in the manner described in Section 4.4. This section presents and discusses the scenarios investigated and their results.

6.1. Evaluating the optimised policies

The policies were optimised twice: once with the process durations fixed to 60 days, and a second time where the optimisation algorithm is free to optimise the process duration. These are differentiated by the suffixes appended to the policy name. Where the process duration is fixed, the suffix is ‘A’ and if the process duration is optimised, the suffix is ‘B’.

The optimisations were run on the case study data previously described. During the EA optimisation, the fitness evaluation uses the average performance of 500 simulations as this was found to give a good estimate without making computation time(s) too long. Finally, each simulation was started with each product having an initial inventory equal to a quarter of the expected yearly demand.

6.1.1. Policies with benchmark run-times

Tables 6 and 7 summarise the performance of all the policy variants averaged over 50 independent runs of the GA. The first of these (Table 6) compares the performance characteristics of the benchmark policies (BSP0 & FCPA) to those tuned base-stock policies (BSP1-3) where the process run times are fixed to 60 days. The performance characteristics for all policies were evaluated using 20,000 simulations on the same random number seeds. The table shows that the tuned base-stock policies outperform the benchmark policies (substantially with regards to BSP0 and less so for FCPA) in terms of the expected profit generated. This is driven primarily by differences in the revenue (and consequently, backlog penalties and customer service level (CSL)) and storage costs. By scrutinizing the seed costs and the cell culture setup costs, it appears that both BSP0 and the optimised base-stock policies start a similar number of batches so the advantage of the optimised base-stock policies is due to the sequence or timing of the batches ordered. It is worth pointing out that BSP0 does outperform the optimised policies in one metric: changeover costs. This means that although the overall timing and sequencing of batches in BSP0 is sub-optimal in terms of the overall objective, it is able to schedule batches of the same product together in longer campaigns thereby reducing changeover costs and potentially making the operation of the facility more straightforward with fewer manufacturing switches between products. In contrast, while FCPA does represent a large improvement on BSP0 with regards to profit its costs are only slightly lower than BSP0 due to its higher changeover, USP, and DSP costs.

Between the optimised policies, BSP1A is the worst, with BSP2A performing a bit better, and BSP3A best. That ranking is the same when looking at revenue but is reversed with regards to the total costs. The only other major difference between them is that BSP2A appears to have on average at least two fewer product changeovers than the other policies. This is however offset by it having larger storage costs than BSP1A and BSP3A. Any differences between two policies in total seed, USP or DSP costs are due to a policy having marginally fewer or more batches started than the other, since they all have the same process run times.

6.1.2. Policies with optimised run-times

In Table 7 the performance characteristics of the best performing policy that had a fixed process run
Table 6: Profit, costs, customer service level (CSL), and other performance characteristics for the three BSPs and FCP with process duration fixed to 60 days as well as the BSP0 solution. Mean ± std. err. are listed of 50 runs (for the tuned policies) each and values reported are in RMU apart from CSL values. Best mean values are highlighted in **bold**; values not statistically significantly different from the best are likewise highlighted.

<table>
<thead>
<tr>
<th></th>
<th>BSP0</th>
<th>FCPA</th>
<th>BSP1A</th>
<th>BSP2A</th>
<th>BSP3A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profit</td>
<td>179015 ± 20.9</td>
<td>186544 ± 14.9</td>
<td>189589 ± 1.6</td>
<td>189651 ± 4.5</td>
<td><strong>189711 ± 1.7</strong></td>
</tr>
<tr>
<td>Revenue</td>
<td>214690 ± 17.9</td>
<td>221708 ± 21.5</td>
<td>222908 ± 6.8</td>
<td>222996 ± 6.0</td>
<td><strong>223075 ± 7.5</strong></td>
</tr>
<tr>
<td>Total costs</td>
<td>35676 ± 5.7</td>
<td>35164 ± 33.4</td>
<td><strong>33319 ± 7.1</strong></td>
<td>33345 ± 6.1</td>
<td>33363 ± 8.0</td>
</tr>
<tr>
<td>Seed</td>
<td><strong>179.0 ± 0.04</strong></td>
<td>185.3 ± 0.12</td>
<td>181.3 ± 0.03</td>
<td>181.7 ± 0.02</td>
<td>181.9 ± 0.03</td>
</tr>
<tr>
<td>USP</td>
<td>8059 ± 0.9</td>
<td>8346 ± 5.4</td>
<td>8155 ± 1.1</td>
<td>8173 ± 1.1</td>
<td>8181 ± 1.5</td>
</tr>
<tr>
<td>Replacement ATF filters</td>
<td><strong>10.3 ± 0.09</strong></td>
<td>10.9 ± 0.02</td>
<td>10.5 ± 0.01</td>
<td>10.6 ± 0.01</td>
<td><strong>10.6 ± 0.01</strong></td>
</tr>
<tr>
<td>Cell culture setup</td>
<td><strong>1064.6 ± 0.22</strong></td>
<td>1100.0 ± 0.68</td>
<td>1076.9 ± 0.15</td>
<td>1079.6 ± 0.15</td>
<td><strong>1080.7 ± 0.21</strong></td>
</tr>
<tr>
<td>DSP</td>
<td>21022 ± 2.5</td>
<td>21736 ± 13.7</td>
<td>21265 ± 2.8</td>
<td>21313 ± 3.1</td>
<td><strong>21342 ± 4.2</strong></td>
</tr>
<tr>
<td>Changeover</td>
<td><strong>478.0 ± 0.24</strong></td>
<td>1091.9 ± 12.17</td>
<td>1132.5 ± 0.44</td>
<td>1053.1 ± 4.00</td>
<td>1150.1 ± 1.11</td>
</tr>
<tr>
<td>Storage</td>
<td>3065 ± 1.0</td>
<td>3114 ± 25.6</td>
<td>2253 ± 5.0</td>
<td>2315 ± 4.7</td>
<td><strong>2210 ± 4.7</strong></td>
</tr>
<tr>
<td>Backlog penalties</td>
<td>2827.3 ± 5.24</td>
<td>642.8 ± 7.82</td>
<td>285.3 ± 1.70</td>
<td>263.8 ± 1.55</td>
<td><strong>251.8 ± 1.74</strong></td>
</tr>
<tr>
<td>Wastage</td>
<td><strong>44.9 ± 0.16</strong></td>
<td>47.8 ± 0.09</td>
<td>45.8 ± 0.02</td>
<td>45.9 ± 0.02</td>
<td>45.9 ± 0.02</td>
</tr>
<tr>
<td>CSL</td>
<td>95.88% ± 0.007</td>
<td>98.96% ± 0.009</td>
<td>99.50% ± 0.003</td>
<td>99.54% ± 0.004</td>
<td><strong>99.58% ± 0.004</strong></td>
</tr>
</tbody>
</table>

Table 7: Profit, costs, customer service level (CSL), and other performance characteristics for the three BSPs and FCP with process durations optimised by the GA compared to the best performing BSP with fixed process duration. Policy names with a suffix of ‘A’ are for optimisations with process duration fixed to 60 days while those with suffix ‘B’ have process duration optimised by the EA. Mean ± std. err. are listed of 50 runs each and values reported are in RMU apart from CSL values. Best mean values are highlighted in **bold**; values not statistically significantly different from the best are likewise highlighted.

<table>
<thead>
<tr>
<th></th>
<th>FCPB</th>
<th>BSP3A</th>
<th>BSP1B</th>
<th>BSP2B</th>
<th>BSP3B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profit</td>
<td>186585 ± 127.1</td>
<td>189711 ± 1.7</td>
<td>189972 ± 6.3</td>
<td><strong>190016 ± 13.9</strong></td>
<td><strong>190125 ± 1.9</strong></td>
</tr>
<tr>
<td>Revenue</td>
<td>221389 ± 73.8</td>
<td>223075 ± 7.5</td>
<td>223101 ± 9.2</td>
<td>223123 ± 9.4</td>
<td><strong>223240 ± 7.3</strong></td>
</tr>
<tr>
<td>Total costs</td>
<td>34804 ± 99.5</td>
<td>33363 ± 8.0</td>
<td>33129 ± 6.8</td>
<td><strong>33107 ± 10.2</strong></td>
<td><strong>33116 ± 7.2</strong></td>
</tr>
<tr>
<td>Seed</td>
<td><strong>172.2 ± 3.72</strong></td>
<td>181.9 ± 0.03</td>
<td>182.2 ± 0.72</td>
<td>178.8 ± 0.86</td>
<td>185.8 ± 0.36</td>
</tr>
<tr>
<td>USP</td>
<td><strong>8128 ± 36.3</strong></td>
<td>8181 ± 1.5</td>
<td>8148 ± 8.3</td>
<td><strong>8110 ± 9.5</strong></td>
<td>8178 ± 5.0</td>
</tr>
<tr>
<td>Replacement ATF filters</td>
<td>13.5 ± 0.36</td>
<td><strong>10.6 ± 0.01</strong></td>
<td>11.1 ± 0.11</td>
<td>11.5 ± 0.09</td>
<td>11.3 ± 0.07</td>
</tr>
<tr>
<td>Cell culture setup</td>
<td><strong>1004.9 ± 21.24</strong></td>
<td>1080.7 ± 0.21</td>
<td>1065.7 ± 4.34</td>
<td>1043.1 ± 4.87</td>
<td>1077.0 ± 2.68</td>
</tr>
<tr>
<td>DSP</td>
<td>21602 ± 44.8</td>
<td>21342 ± 4.2</td>
<td><strong>21248 ± 3.6</strong></td>
<td>21282 ± 5.7</td>
<td>21256 ± 4.0</td>
</tr>
<tr>
<td>Changeover</td>
<td><strong>1119.2 ± 21.22</strong></td>
<td>1150.1 ± 1.11</td>
<td>1211.1 ± 4.13</td>
<td>1150.5 ± 6.02</td>
<td>1249.5 ± 1.63</td>
</tr>
<tr>
<td>Storage</td>
<td>2998 ± 96.7</td>
<td>2210 ± 4.7</td>
<td>2050 ± 8.2</td>
<td>2102 ± 12.6</td>
<td><strong>202 ± 6.1</strong></td>
</tr>
<tr>
<td>Backlog penalties</td>
<td>721.5 ± 21.33</td>
<td>251.8 ± 1.74</td>
<td>243.5 ± 2.26</td>
<td>235.3 ± 3.04</td>
<td><strong>197.8 ± 2.06</strong></td>
</tr>
<tr>
<td>Wastage</td>
<td>62.6 ± 3.41</td>
<td><strong>45.9 ± 0.02</strong></td>
<td>46.9 ± 0.39</td>
<td>48.4 ± 0.36</td>
<td>46.4 ± 0.19</td>
</tr>
<tr>
<td>CSL</td>
<td>98.83% ± 0.035</td>
<td>99.58% ± 0.004</td>
<td>99.60% ± 0.004</td>
<td>99.61% ± 0.004</td>
<td><strong>99.66% ± 0.004</strong></td>
</tr>
</tbody>
</table>

16
time — BSP3A in this case — is compared with the benchmark fixed cycle policy that optimised process run time, and the base stock policies that optimised both the inventory policy parameters and also the process run time. First, one can observe that the FCP policies are always worse than the tuned base stock policies regardless of process run time optimisation. In fact, when comparing FCPA and FCPB, although there is an improvement, it is coupled with an order-of-magnitude increase in variance. This is likely due to the FCPB parameter optimisation running out of its computation budget before fully converging (see Figure A.8 in the Appendix). On the whole, it suggests that utilising BSP1, the best policies have just a second parameter is not particularly useful. This would mean that the policy could be replaced by one with just $Y^{(1)}$ (like BSP3) which would require less effort tuning it — i.e., if it were to be run again but with one policy parameter it should give faster convergence but similar results. This assumption has been confirmed experimentally.

The reason that the optimised $Y^{(1)}$ values are higher than that of the benchmark policy is most likely due to the fact that the optimised policies implicitly account for the fact that the product changeovers can only hap-

---

Table 8: Policy parameters for $p1$, $p2$, and $p3$ in each of the base-stock policies. The best solutions (i.e., the best solution out of all EA runs for each tuned policy) are reported.

<table>
<thead>
<tr>
<th></th>
<th>$y^{(1)}$</th>
<th>$y^{(2)}$</th>
<th>$y^{(3)}$</th>
<th>$y^{(4)}$</th>
<th>$y^{(5)}$</th>
<th>$B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BSP0</td>
<td>6.2</td>
<td>52.5</td>
<td>32.7</td>
<td>12.8</td>
<td>42</td>
<td>60</td>
</tr>
<tr>
<td>BSP1A</td>
<td>16.4</td>
<td>16.7</td>
<td>15.2</td>
<td>16.2</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>BSP1B</td>
<td>18.7</td>
<td>19.9</td>
<td>17.1</td>
<td>47</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BSP2A</td>
<td>10.5</td>
<td>27.5</td>
<td>15.2</td>
<td>60</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BSP2B</td>
<td>12.8</td>
<td>20.3</td>
<td>15.0</td>
<td>43</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BSP3A</td>
<td>16.3</td>
<td>43</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BSP3B</td>
<td>16.3</td>
<td>43</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$y^{(1)}$</th>
<th>$y^{(2)}$</th>
<th>$y^{(3)}$</th>
<th>$y^{(4)}$</th>
<th>$y^{(5)}$</th>
<th>$B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BSP0</td>
<td>11.1</td>
<td>93.6</td>
<td>32.7</td>
<td>12.8</td>
<td>42</td>
<td>60</td>
</tr>
<tr>
<td>BSP1A</td>
<td>28.8</td>
<td>28.9</td>
<td>15.2</td>
<td>60</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BSP1B</td>
<td>25.1</td>
<td>25.5</td>
<td>17.1</td>
<td>58</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BSP2A</td>
<td>23.5</td>
<td>38.4</td>
<td>25.3</td>
<td>60</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BSP2B</td>
<td>22.4</td>
<td>31.5</td>
<td>23.4</td>
<td>59</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BSP3A</td>
<td>23.8</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BSP3B</td>
<td>23.8</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$y^{(1)}$</th>
<th>$y^{(2)}$</th>
<th>$y^{(3)}$</th>
<th>$y^{(4)}$</th>
<th>$y^{(5)}$</th>
<th>$B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p3$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BSP0</td>
<td>10.7</td>
<td>77.5</td>
<td>32.7</td>
<td>12.8</td>
<td>42</td>
<td>60</td>
</tr>
<tr>
<td>BSP1A</td>
<td>23.8</td>
<td>23.9</td>
<td>15.2</td>
<td>60</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BSP1B</td>
<td>21.0</td>
<td>21.6</td>
<td>17.1</td>
<td>65</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BSP2A</td>
<td>19.1</td>
<td>39.9</td>
<td>21.3</td>
<td>60</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BSP2B</td>
<td>6.6</td>
<td>36.4</td>
<td>18.8</td>
<td>74</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BSP3A</td>
<td>28.8</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BSP3B</td>
<td>19.2</td>
<td>79</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 9: The optimal sequence and process run times for the fixed cycle policies overall from all GA runs.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>$B_{p1}$</th>
<th>$B_{p2}$</th>
<th>$B_{p3}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCPA</td>
<td>1-3-2-3-0-3-0-2</td>
<td>60</td>
<td>60</td>
</tr>
<tr>
<td>FCPB</td>
<td>2-0-1-3-0-2-3-0-1-0-2-3</td>
<td>42</td>
<td>48</td>
</tr>
</tbody>
</table>
pen after the end of a batch of a predetermined run time. This means it can maintain higher levels of safety stock and reduce the likelihood of stock-outs. And with the narrower ranges of the inventory policy parameters, more frequent product changeovers help avoid product inventories from falling critically low. The benchmark base-stock policy is based on heuristics that do not model batch or semi-batch production; this highlights how previous approaches to lot-sizing approaches don’t easily apply to biopharmaceutical manufacturing contexts. The best FCP solutions also highlight that campaigns of multiple batches are seemingly sub-optimal at this problem load. Though these policies can schedule consecutive batches of the same product, the best solutions do not implement that strategy.

Observed solutions for process run times are based on striking a balance between the process yields and demand forecasts of each product as well as the increasing risk of process failure with longer process duration. It is intuitive, based on the specifics of the case study, that p1 would have a shorter process run time than the other products because it has much lower demand. By the same token, it makes sense that p3 has a longer process run time than p2 because it has similar demand but the yield of its manufacturing process is a third lower.

There are no major differences in the optimised process run times between the different base-stock policies apart from p2 in BSP3B and p3 in BSP1B where the run times are significantly lower than that of the other tuned BSPs. Also, all the optimised durations deviate from the ‘benchmark’ process run time of 60 days especially with p1 and p3. It is interesting to note that the optimised policies tend to have process run times that cannot produce all of the expected demand for a year in a single batch even if it is possible — process durations of 53 and 88 would suffice for p1 and p2 respectively. As previously mentioned, the factors determining these decisions are the need to mitigate process failure and the ability to changeover to other critical products. It is not clear which role each of those factors play in each case but it is fair to say that mitigating process failure is more influential for p2 than it is for p1.

6.1.4. Production schedule(s) for the facility

Although the Gantt charts in Figure 5 are just for one scenario and simulation run, many of the points previously discussed are illustrated here. The Gantt charts shown are for BSP0 and the best performing solutions for FCPB, BSP3A, and BSP3B. BSP0 schedules batches of the same product together so that campaigns have multiple batches and minimises changeover costs as a result. As a result of these sustained campaigns, the inventory will be built up and incur higher storage costs. The BSP0 schedule makes it clear how this policy can accrue very high backlog costs. For example, in this instance, p2 is not produced until well into the second year meaning all of the first year’s demand would have been backlogged and subsequently penalised. These trends are all reflected in Table 6.

Moving from BSP0 to BSP3A and then to BSP3B, the number of multi-batch campaigns decreases with BSP3B not scheduling any two batches of the same product together in this particular scenario — this also applies to FCPB as the sequence does not include any consecutive batches of the same product. With increasing number of product changeover, there is more production time lost. This is due to the fact that changeover between batches of different products requires more setup time than the turnaround between two batches of the same product. This is why the BSP0 chart appears to be less utilised. The difference between the policies with fixed process run times and those where the process run times are optimised is also clear from these schedules. This is most visible when comparing the Gantt charts of BSP3A and BSP3B. The former has batches that are more uniform in size unless a failure occurs whereas the latter looks more irregular.

What these charts also illustrate is a disadvantage of FCPB (and the FCP policies in general): although it is able to dictate a schedule ahead-of-time, it is not able to find a ‘good’ decision to make when the environment changes. This is true especially in the event of process failure where it will move on to the next batch in the sequence regardless of any critically low inventory levels. As shown in Table 7, this can lead to higher backlog penalties and lower revenue. In this instance, this occurs in the second batch of p3 in the third year of the FCPB schedule. On the other hand, while the BSPs cannot fix a schedule in advance or determine an absolute decision to make when process failure occurs, they can react to changes by deciding at each point whether to continue with that product, go idle or switch to another product.

6.1.5. EA performance and statistical analysis

Statistical testing on the performance of the policies was carried out, the results of which are presented in Table 10. It contains a matrix of p-values from Mann-Whitney U tests comparing the profits of the final solutions from each policy. It indicates that in terms of
Table 10: Matrix of the observed significance level of Mann-Whitney tests comparing the mean profit of the final solutions from each policy. The p-values are for the test that policy$_1$ > policy$_2$.

<table>
<thead>
<tr>
<th>policy$_1$</th>
<th>BSP1A</th>
<th>BSP1B</th>
<th>BSP2A</th>
<th>BSP2B</th>
<th>BSP3A</th>
<th>BSP3B</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSP1A</td>
<td>&lt;0.001</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>BSP1B</td>
<td>1</td>
<td>&lt;0.001</td>
<td>1</td>
<td>&lt;0.001</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>BSP2A</td>
<td>&lt;0.001</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>BSP2B</td>
<td>&lt;0.001</td>
<td>&lt;0.001</td>
<td>&lt;0.001</td>
<td>&lt;0.001</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>BSP3A</td>
<td>&lt;0.001</td>
<td>1</td>
<td>&lt;0.001</td>
<td>1</td>
<td>&lt;0.001</td>
<td>1</td>
</tr>
<tr>
<td>BSP3B</td>
<td>&lt;0.001</td>
<td>&lt;0.001</td>
<td>&lt;0.001</td>
<td>&lt;0.001</td>
<td>&lt;0.001</td>
<td>&lt;0.001</td>
</tr>
</tbody>
</table>

Figure 6 shows the convergence of the optimisations of the base stock policies. In addition to the GA, optimisation with CMA-ES is also tested. Across all the policies both optimisation algorithms deliver essentially the same quality of final solutions. However, the more state-of-the-art CMA-ES converges faster than the GA. This gets more pronounced the more decision variables the problem has — e.g., BSP2 which has four inventory policy parameters per product — and with the policies that have to optimise process durations in addition to the inventory policy parameters. In addition, these EAs were compared with random search and it is shown that the EAs perform significantly better on most of the policies. The exceptions to this are the BSP3 policies where there is no statistical difference between the performance of random search and the EAs. This suggests that BSP3, by virtue of its in-built forecasting heuristic, is able to make good scheduling decisions during simulation. It also helps that the policy has very few decision variables to optimise (six at most and as few
as three when process run times are fixed) so the search space is smaller.

To test the benefit of the forecasting heuristic, the BSP3 policies were compared with BSP1 policies where \( Y^{(1)} = Y^{(2)} \). This constraint is to ensure that both policies work in the same way except that the BSP3 policies break ties with the forecasting heuristic and the BSP1 policies break ties by comparing run-out times. So for both the BSP3 and the constrained BSP1, 2,000 policies were randomly generated and evaluated. The mean performance (± standard error) of the BSP3 policies was 167933 ± 850 RMU and the performance of the constrained BSP1 policies was 166709 ± 855 RMU. A two-tailed Mann-Whitney U test was performed on the policies and indicated that the increase in performance of the BSP3 policies was statistically significant (observed significance level of <0.001).

Overall, these analyses have shown that it is beneficial to tune policy parameters for the scheduling policies instead of relying on estimated policy parameters or a fixed schedule or sequence of batches. Furthermore, optimising process run times offers additional advantages as it allows the policy to schedule batches so that product changeover can occur when inventory falls to critical levels and also select run times with efficient productivities and an acceptable risk of process failure. Thirdly, the choice of optimisation algorithm, in this scenario, does not have a significant effect on the quality of the final solution. However, as the number of decision variables increases, a more efficient algorithm such as CMA-ES can contribute to a faster convergence (with regards to fitness evaluations).

### 6.2. Evaluating the sensitivity and robustness of optimised policy solutions

The case study that the policies have been optimised for and evaluated on has a demand forecast which is stochastic but follows a known distribution. In reality, it is not easy to accurately predict or forecast demand — which is why the demand is defined as a probability distribution with a mean based on targets and market research (and variance based on estimated margin of error in predictions). However, as actual demand is being realised it can be difficult to determine whether the observed demand actually is sampled from the same

---

**Figure 6:** Convergences of the all the BSP optimisations runs with the genetic algorithm (GA), CMA-ES, and with random search (RS). Errorbars indicate standard error of the mean.
Table 11: Robustness of the optimised policies to overestimates (-10%) and underestimates (+10%) of demand. Mean profit ± std. err. are listed of the best solutions from the 50 GA runs from each of the tuned policies — values reported are in RMU. Best mean values are highlighted in bold; values not statistically significantly different from the best are likewise highlighted.

<table>
<thead>
<tr>
<th>Demand case</th>
<th>BSP0</th>
<th>FCP1B</th>
<th>BSP1B</th>
<th>BSP2B</th>
<th>BSP3B</th>
</tr>
</thead>
<tbody>
<tr>
<td>−10%</td>
<td>163109 ± 20</td>
<td>167100 ± 158</td>
<td>170650 ± 7</td>
<td>170679 ± 14</td>
<td>170719 ± 8</td>
</tr>
<tr>
<td>Standard</td>
<td>178940 ± 21</td>
<td>186585 ± 127</td>
<td>189972 ± 6</td>
<td>190015 ± 14</td>
<td>190125 ± 2</td>
</tr>
<tr>
<td>+10%</td>
<td>191997 ± 22</td>
<td>200088 ± 535</td>
<td>200938 ± 176</td>
<td>201904 ± 217</td>
<td>200471 ± 82</td>
</tr>
</tbody>
</table>

Figure 7: Box plots of the performance of the tuned policies relative to the benchmark heuristic BSPO when demand is overestimated (-10%) and underestimated (+10%). Whiskers of the box plots represent 95% of the data range; outliers are not shown.

distribution as the one predicted. So the observed demand could potentially be from a distribution with a different mean and/or variance. This section aims to determine the behaviour of the previously obtained solutions on slightly different problem instances to the one they were trained on.

With those considerations, some of the policies were tested at different demand scales to determine their robustness. Specifically, the best solutions from each of the 50 optimisation runs of FCPB, BSP1B, BSP2B, and BSP3B as well as the previously estimated BSP0 solution were evaluated over 20,000 simulations on demand load cases ±10% of the standard demand case used to train the policies. As before, the demand due in each year is stochastic and described by a Normal distribution, \(N(\mu, (0.025\mu)^2)\). Apart from the change in demand load, all the other model and problem parameters were unchanged from those in the case study used in Section 6.1.

Results of these tests are shown in Table 11. The −10% case is where demand has been overestimated — i.e., the standard demand case used to train the policies is an overestimate of the actual demand. Similarly, the +10% case is where demand has been underestimated. Figure 7 shows the descriptive statistics of the performance, relative to BSP0, of 50 solutions from each policy. A summary of statistical tests of these experiments is in the Appendix (Table A.12). The data at the standard demand case is just a replication of the information from Tables 6 & 7 showing that the solutions of BSP3B are better than those of BSP2B, followed by BSP1B and then FCPB.

Looking at cases where the demand is overestimated (−10%), the rankings stay the same (compared with the standard demand case), with policies with a higher ranking being statistically significantly better than the
policies ranking below it. In addition, apart from FCPB, the variance in performance for all policies is very small and is within narrow ranges. These rankings change in the case demand is underestimated (+10%). Here, BSP2B has better mean performances, followed by BSP1B, then BSP3B and FCP1B. At this demand, no policy is statistically significantly better than all of the others. BSP3B has a much tighter distribution over its 50 best solutions compared to the other policies. 95% of BSP3B solutions fall within a range of 2%, while the performance of each of the other policies spans almost double that range at least. For example, some FCPB solutions improve on BSP0 by only 1% but some ‘good’ solutions show as much as 6% improvement.

BSP3B is less robust in terms of the policy parameters to a higher demand load even as its expected performance is less variable. However, when we run the robustness test with BSP3B where its \( Y^{(3)} \) values are the \( Y^{(5)} \) values of BSP1B and process run time values are substituted for that of BSP1B, BSP3B outperforms BSP1B which shows that our forecasting heuristic is still rather robust and effective.

Relative to the benchmark policy, which has its parameters heuristically chosen, the performances of the tuned policies (apart from FCPB) generally decline as one moves away from the standard demand case. For the BSPs, a partial explanation for this trend is that at less loaded problems, sub-optimal decisions incur less harsh penalties. For example, the main driver for the difference between BSP0 and the tuned policies at the standard demand case is backlog penalties. So with lower observed demand, clearly this becomes less important and the difference decreases. At higher demand load, scheduling in campaigns becomes more important as it saves on changeover time and allows more production time in the facility. As a result, policies that schedule more batches of the same product together (like BSP0 does) can perform better. This explains the poor performance of BSP3B as it ‘links’ very few batches of the same product together into campaigns compared to the other policies. For FCPB, the trend is increasing average relative performance with increasing demand. An explanation for this is that the FCPB solutions we have are composed of a significant proportion of policies which have long process run times. These longer process run times help increase performance at the higher demand. A more general explanation applicable to FCPB and the BSPs is that tuning a policy exploits the specific structure and characteristics of the problem instance it is trained on to generate better solutions and performance. This also means that it loses its applicability to moderately different or more general problems — this is referred to as overfitting.

Finally, we carried out further sensitivity analysis by doing separate GA optimisation runs for BSP2B and BSP3B at both +10% and −10% demand cases — i.e., we specifically tune policy parameters to these demand loads. Here, we find that BSP3B retains its advantage over BSP2B with similar margins to the standard demand case. In addition, compared to these newly optimised policies, the solutions trained on the standard case but evaluated on the +10% demand case perform relatively badly. They perform between 3% and 4% lower than either BSP2B or BSP3B policies optimised for the higher demand load. At the −10% demand case, the difference is much smaller and the policies trained on the standard case have performance similar to the policies optimised for the lower demand load.

Overall what can be observed from these tests is that BSP3B policies perform best when demand is lower or equal to the demand instance they were trained on. Although other policies outperform BSP3B when demand is higher, its forecasting heuristic remains robust. In addition, at higher demand loads, the policies trained on the standard demand case perform poorly. This implies that the demand forecasts need to be done carefully and there should be a mechanism to adapt policy parameters to changing demand.

7. Conclusions

This paper has considered the stochastic economic lot scheduling problem (SELSP) in the context of a biopharmaceutical manufacturing scenario consisting of multiple products on a single facility utilising semi-continuous perfusion processes that are prone to various types of process failure that have significant operational consequences. To deal with the challenges that this problem poses, a simulation optimisation approach was proposed and developed.

First, a custom discrete-event simulation framework modelling continuous bioprocesses in a scheduling environment was developed. Then a few dynamic scheduling policies were adapted from the literature to fit the problem being investigated. In addition to this, a novel policy with a forecasting heuristic was proposed. These policies were then tuned on a synthetic case study using an evolutionary algorithm (EA) in what amounts to a hyper-heuristic search. Evaluation of these policies and further comparison with a heuristically determined
benchmark policy as well as a benchmark policy based on a fixed sequence demonstrated the benefit of tuning policy parameters and utilising policies that use the state of the scheduling environment to make decisions. Further tuning of process run times led to improved performance as this enables better lot-sizing decisions which may allow hedging against process failure by utilising a shorter run time. Finally, robustness and sensitivity analysis showed that the optimised policies are more robust to demand overestimation than demand underestimation with some policies suffering higher uncertainty and variance at unexpectedly higher demand loads.

There are a few possible ways to extend the work presented in this paper. The obvious one is to apply this method to different case studies — e.g., clinical manufacturing; or instances where the process yields are also uncertain and stochastic; or problems that introduce sequence-dependent changeover times by including both fed-batch and perfusion cell cultures. It may be worth developing a method so that policy parameters are adaptable to changing demand, or combining the better robustness of BSP2 with the forecasting heuristic of BSP3. In addition, problems with more products and multiple facilities may require a more diverse representation of potential rules (Branke et al., 2015) and careful consideration of strategies to deal with expensive evaluations and noise as the problem size increases (Jin and Branke, 2005; Branke, 2018). Finally, the run times of batches are fixed ahead of each simulation so policies can only react to changes at the end of a batch. It would be interesting to investigate policies that allow flexible process run times to enable more fine-grained and immediate responses to changes in the simulation environment.

Acknowledgements

The first author would like to acknowledge funding through the EPSRC Centre for Doctoral Training in Emerging Macromolecular Therapies, EP/L015218/1. The third and fourth authors gratefully acknowledge funding through the Future Targeted Healthcare Manufacturing Hub EP/P006485/1. UCL Biochemical Engineering hosts the Future Targeted Healthcare Manufacturing Hub in collaboration with Warwick Business School and other UK universities and with funding from the UK Engineering & Physical Sciences Research Council (EPSRC) and a consortium of industrial users and sector organisations.

Abbreviations and Notation

**Abbreviations**

**ATF** alternating tangential flow  
**CMA-ES** Covariance Matrix Adaptation Evolution Strategy  
**CSL** customer service level  
**DSP** downstream processing  
**EA** evolutionary algorithm  
**GA** genetic algorithm  
**mAb** monoclonal antibody  
**MILP** mixed-integer linear programming  
**RMU** relative monetary units  
**SCLSP** stochastic capacitated lot sizing problem  
**SELSP** stochastic economic lot scheduling problem  
**SLSP** stochastic lot scheduling problem  
**USP** upstream processing

**Parameters**

The indices \( p \) and \( t \) denote individual products and discrete time points respectively. The subset characterising the facility being considered is \( PF \), the set of products produced by the facility.
\(\alpha_p\) lead time for production of first DSP batch of product \(p\), days
\(\delta_p\) unit cost charged as penalty for each kilogram of unfulfilled demand of product \(p\), RMU per kilogram per day
\(\xi_p \) shelf-life of product \(p\), days
\(\mu_p^D\) mean daily demand of product \(p\), kilograms
\(\rho_p\) unit cost for each stored kilogram of product \(p\), RMU per kilogram
\(\sigma_p^D\) standard deviation of daily demand of product \(p\), kilograms
\(\theta\) daily backlog decay rate
\(\upsilon_p\) unit sales price for each kilogram of product \(p\), RMU per kilogram
\(\psi_p\) changeover cost for starting a campaign of product \(p\), RMU
\(A_p\) sum of all costs incurred up till the end of ramp-up for first batch in campaign of product \(p\), RMU
\(br_p\) bioreactor yield per daily perfusion harvest, kilograms
\(yd_p\) overall process yield, %

**State variables**

\(\Delta_{pt}\) amount of product \(p\) which is late at time \(t\), kilograms
\(D_{pt}\) observed demand of product \(p\) at time \(t\), kilograms
\(I_{pt}\) the amount of product \(p\) stored at time \(t\), kilograms
\(m_t\) the product being manufactured on the facility at time \(t\), 0 if idle
\(S_{pt}\) the amount of product \(p\) sold at time \(t\), kilograms
\(Z_t^{(i)}\) the set of products \(p\), such that \(I_{pt} \leq Y_p^{(i)}\)

**Decision variables (policy parameters)**

\(B\) run time of perfusion cell culture batch, days
\(Y_p^{(1)}\) reorder point, kg
\(Y_p^{(2)}\) order-up-to level, kg
\(Y_p^{(3)}\) can-order point, kg
\(Y_p^{(4)}\) can-order-up-to level, kg
\(Y_p^{(5)}\) reorder point, kg (BSP3)

### Appendix A. Statistical Tests and FCP Convergence

This appendix details the statistical testing for the demand robustness study as well as the progress and convergence of the fixed cycle policy optimisation run(s). The former is shown in Table A.12 and the latter in Figure A.8.

![Figure A.8: Convergences of the genetic algorithm optimisation of both the fixed cycle policies, FCPA and FCPB.](image)

Table A.12: Matrices of observed significance level of Mann-Whitney tests comparing the mean profit of the best solutions of each policy when demand is overestimated and underestimated. The \(p\)-values are for the test that \(\langle row \rangle > \langle column \rangle\).

<table>
<thead>
<tr>
<th>Demand overestimate (−10%)</th>
<th>FCP1B</th>
<th>BSP1B</th>
<th>BSP2B</th>
<th>BSP3B</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCP1B</td>
<td>1.00E+00</td>
<td>9.99E+00</td>
<td>1.00E+00</td>
<td></td>
</tr>
<tr>
<td>BSP1B</td>
<td>3.43E-18</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BSP2B</td>
<td>3.43E-18</td>
<td>6.74E-04</td>
<td></td>
<td>9.68E-01</td>
</tr>
<tr>
<td>BSP3B</td>
<td>3.43E-18</td>
<td>1.56E-09</td>
<td>3.21E-02</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Demand underestimate (+10%)</th>
<th>FCP1B</th>
<th>BSP1B</th>
<th>BSP2B</th>
<th>BSP3B</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCP1B</td>
<td></td>
<td>7.28E-01</td>
<td>9.21E-01</td>
<td>9.44E-01</td>
</tr>
<tr>
<td>BSP1B</td>
<td>2.72E-01</td>
<td></td>
<td></td>
<td>1.00E+00</td>
</tr>
<tr>
<td>BSP2B</td>
<td>7.88E-02</td>
<td>1.19E-04</td>
<td></td>
<td>1.78E-07</td>
</tr>
<tr>
<td>BSP3B</td>
<td>5.56E-02</td>
<td>8.50E-01</td>
<td>1.00E+00</td>
<td></td>
</tr>
</tbody>
</table>


Branke, J., Hildebrandt, T., Scholz-Reiter, B., 2015. Hyper-heuristic


