Optimising the design of buffer preparation in bioprocessing facilities

Sean Tully, M.A. (*Cantab.*), M.Eng.

July 16, 2017

A thesis submitted to University College Dublin in part fulfilment of the requirements of the degree of Master of Science in Business Analytics

Michael Smurfit Graduate School of Business,

University College Dublin

August, 2017

Supervisor: Prof. M. O'Neill

Head of School: Professor Ciarán Ó hÓgartaigh

Dedication

To . . .

Contents

List of Figures

List of Tables

List of Algorithms

Preface

> The cold smell of potato mould, the squelch and slap
> Of soggy peat, the curt cuts of an edge
> Through living roots awaken in my head.
> But I've no spade to follow men like them.
> Between my finger and my thumb
> The squat pen rests.
> I'll dig with it.
>
> — Seamus Heaney, *Digging*

This thesis was motivated by ...

We begin in Chapter 1 with a short overview of ...

In Chapter 2 we ...

Chapter 3 introduces ...

In Chapter 4, the ...

We explore in Chapter 5 ...

Chapter 6 examines ...

Finally, in Chapter 7, we ... and indicate some possible avenues for further research following on from the ideas introduced in this thesis.

University College Dublin Sean Tully
July 16, 2017

Acknowledgements

I would like to thank ...

Abstract

It is intended that the abstract be used, while work is ongoing, as a brief summary of the state of progress of the dissertation. The abstract shall be completed once the main body of the dissertation is complete. On completion, it will contain a high-level overview of the work done, in simple, plain language.

Chapter 1

Introduction

> Well I don't think we're *for* anything. We're just products of evolution. You can say, "Gee, your life must be pretty bleak if you don't think there's a purpose." But I'm anticipating having a good lunch.
>
> — James Watson, *in conversation with Richard Dawkins*

1.1 Background

This chapter gives a brief outline of bioprocessing and explains the background to the research.

1.1.1 Bioprocessing

Before the advent of biotechnology, most therapeutics (medicines) were what are now termed *small molecule* drugs. The pharmaceutical industry was concerned with the synthesis of these products via predominantly chemical processes, such as reaction, distillation and crystallisation. Small molecule drugs typically consist of tens or hundreds of atoms, such as paracetamol, which has a molar mass of approximately 151 g/mol, or aspirin (acetylsalicylic acid), which has a molar mass of approximately 180 g/mol.

With the advent of recombinant D.N.A technology, biochemists gained the ability to re-program the D.N.A. of simple biological microorganisms such as *Escherichia coli* and, eventually, mammalian cells, such as those of the Chinese Hamster (*Cricetulus griseus*). The genetic structure of these cells could be modified to produce complex molecules, which had previously proved difficult or impossible to synthesise by any other means. The biopharmaceutical industry is concerned with the synthesis of therapeutics via such biological pathways. These products are known by various names, such as *biopharmaceuticals*, *protein therapeutics* or, colloquially, as *biotech drugs*.

The first protein therapeutic to be synthesised on a large scale using biological pathways was insulin, which is a hormone used to regulate metabolism and is administered to individuals suffering from diabetes. Human insulin has a molar mass of approximately 5808 g/mol. It was not possible to commercialy synthesise insulin chemically and it was initially produced by extracting the hormone from the pancreases of mammals such as cows or pigs. In 1978, scientists working at the American company Genentech (now a subsidiary of the Swiss pharmaceutical company F. Hoffmann-La Roche AG) successfully modified cells of *E. coli* to produce insulin and this synthetic insulin was first brought to market in 1982.

The industry that has grown up around the production of biopharmaceuticals is known as the *biopharmaceutical industry*, or, colloquially, as the *biotech* or *biopharma* industry. A report by strategy consultants McKinsey & Company (Otto *et al.*, 2014) estimates that the biopharmaceutical market had global revenues of $163 billion per annum and was worth 20% of the overall pharmaceutical market. Otto *et al.* (2014) also note that large-scale biopharmaceutical manufactring facilities typically cost in the region of "$200 million to $500 million or more" to build.

1.1.2 Bioprocess Engineering

Schaschke (2014) defines bioprocess engineering as "A specialist branch of (chemical) engineering that involves the design and operation of processes used

for the production of biological products such as foods, pharmaceuticals, and biopolymers." The design of a large-scale biopharmaceutical facility typically takes about two years and requires a multidisciplinary team of engineers and scientists.

1.1.3 Upstream and Downstream

A complete facility typically starts with frozen vials of cells (the *working cell bank*) and finishes with the final formulated product in either bulk form or filled into its final packaging *e.g.* syringes. Facilities are nominally divided into *upstream* and *downstream* sections. The upstream section is predominantly concerned with the expansion of cells from a small vial into progressively larger tanks of *media*. The final stage of this growth occurs in the *production bioreactor*, wherein the conditions can be altered to encourage the cells to produce the target protein.

At the interface between upstream and downstream lies the *harvest* section. In the harvest section, some initial separation is performed to begin to isolate the target protein from the contents of the batch (which at this point include cells, cell waste, growth media, antibiotics and myriad other contaminants). At the end of the harvest section, all traces of the host cells should be removed and the batch is said to be *cell-free*. Different interpretations exist in the industry as to where the upstream-downstream split occurs, but it usually is defined as being at some point in the harvest section.

Downstream processing is concerned with taking the cell-free but otherwise contaminated batch and purifying it through a series of orthogonal processes. Such processes can include filtration, ultrafiltration/diafiltration (*UF/DF*), chromatogrpahy, reaction, virus inactivation and formulation. At the end of downstream processing, a batch should consist of formulated bulk product, ready to be filled into its final packaging for delivery. The final fill/finish steps often occur in a separate, sterile facility.

1.1.4 Buffers and Media

Both upstream and downstream sections require large volumes of aqueous solutions to be prepared and stored. Solutions used upstream are typically referred to as *media* and those used downstream are typically referred to as *buffers*. Strictly speaking, media refers to the solutions of nutrients into which cells are expanded, but the term is usually used to encompass all other upstream solutions, such as acids and bases antifoam used in the bioreactors. Strictly speaking, a chemist would define a buffer as a solution which maintains its pH over a wide range of concentrations. Most solutions used downstream do indeed meet this criteria, but the term *buffers* is generally used as a catch-all for all solutions used in the downstream section. A typical process to produce a *monoclonal antibody* (a common family of protein therapeutics) can use tens or hundreds of litres of buffers and media per litre volume in the production bioreactor. Typical large-scale production bioreactor volumes for such processes are in the range 10,000–30,000 litres. Each batch may use in the region of 20–40 different buffers and media.

1.1.5 Buffers and Media Preparation

One of the reasons for the catch-all definitions of *buffers* and *media* in the section above is to do with segregation. The upstream and downstream sections of the plant are segregated to prevent cross-contamination. As a result, there are typically two main areas where solutions are prepared. Buffers are prepared in an area called *buffer preparation*, for use downstream and media are prepared in an area called *media preparation*, for use upstream. There may also be a seperate area for preparing sterile buffers for the final formulation, again to reduce the possibility of contamination and ensure sterility is maintained. In both media and buffer preparation, one vessel is used to prepare the solution and then it is typically sterile filtered into either a hold vessel or the destination vessel.

1.1.6 Design of Buffer Preparation Areas

For a given product, a production process is defined at laboratory scale. The definition of key parameters at laboratory scale can allow process engineers to generate a production-scale mass balance. This mass balance provides, amongst other things, lists of all buffers and media required to make a batch. Two complex optimisation problems now emerge; how do we design the buffer and media preparation areas? For media, the problem is relatively easy to solve with some trial and error, since there are typically only about 10 media used per batch and they often differ vastly in scale; the initial bioreactor may be 20 litres in volume and the production bioreactor may be 20,000 litres in volume. Because of this, the sizing of preparation vessels usually proceeds by picking a vessel capable of preparing the largest medium, defining a minimum fill volume and seeing what else can be prepared in it, then defining another vessel, and so on until sufficient vessels are defined. Media hold vessels, if required, may be similarly defined. The design of media preparation is usually relatively insensitive to schedule.

The problem of designing a buffer preparation area is more difficult to solve. There may be 20 or more different buffer compositions. Often each buffer is used multiple times in the same operation or multiple times across multiple operations. In operations such as chromatography, somewhere in the region of 5–10 buffers may be needed in rapid succession. They tend to be of similar volumes, so an efficient solution will look to maximise the number of buffers prepared in a given preparation vessel. Since they may be needed in rapid succession, this then involves a requirement for multiple hold vessels so some buffers can be made ahead of time. Where buffers are required for multiple steps in an operation or across multiple operations, is it best to perform many preparations, or few? Additionally, is it best to hold the buffer in many seperate hold vessels, allowing them to be individually freed up more quickly, or is it better to consolidate and minimised the number of vessels? Defining success in the design of buffer preparation is also difficult. There exist trade-offs between efficiency and flexibility, capital and operating costs, and many other factors such as installed area, installed volume, operability, layout/adjacency,

piepwork complexity and cleanability.

Due to the high salt concentrations in some buffers, they can prove corrosive to the commonly used grades of stianless steel, such as 316L. Such buffers may have to be prepared or held in vessels made from expensive alloys such as AL-6XN®, which is about 3.5 times more expensive than 316L, or a steel from the Hastelloy® family, which can be eight or more times more expensive than 316L. Ideally, the use of these alloys should be minimised.

Another factor is the use of disposable technology. Buffer preparation, at scales of up to 3,000 litres, can be carried out in disposable sterile bags, rather than vessels. These have a higher consumable cost but are faster to turn around between preparations and reduce the utilisation of cleaning equipment. Similarly, buffers can be held in disposable bags at volumes of up to 5,000 litre. Development of disposables technology is currently rapid and the available sizes and product ranges are increasing each year, so much so that the state of the art has often moved on between the finish of the detailed design of a facility and the start of the first saleable production batch.

1.1.7 Problem Definition

The task of designing a buffer preparation area is complex. Current workflows are largely based on trial-and-error methods using process engineering scheduling software. Typically, a conservatively large array of vessels is chosen and the schedule is run. If there is an individual vessel for each task, the schedule will resolve easily, but the capital and space requirements will be onerous. Via trial-and-error, individual vessels may be removed or resized and the schedule re-run to see if it can be resolved. After some iteration, it becomes difficult or impossible to remove or reduce the vessels any further and resolve the schedule. At this point, iteration stops. In the early feasibility or concept stages of a project, this process is cumbersome, the end points are poorly defined and any development of the underlying process which varies the volumes required may necessitate starting the optimisation again from scratch. An additional factor is that a working solution may exist for a given configuration, but the

scheduling software is unable to resolve the problem, giving a false negative. The scheduling tools used for the process tend to be deterministic, rather than stochastic (although some senitivity analysis is usually built in as an add-on); this makes it difficult to have confidence that a working schedule can handle the real-world batch-to-batch variability inherent in a process that has living cells as its engine.

A more streamlined methodology for solving this optimisation problem is required and this dissertation is concerned with developing such a methodology and a software tool to implement it. The aim is to start with a reduced or basic case, including a number of simplifying assumptions, and develop a working tool to schedule the operations in buffer preparation and to vary the size and number of vessels and the preparation strategy to optimise the process with respect to some metric. One a working framework has been developed, additional constraints can be added and simplifications can be removed to provide a better approximation of real-world conditions.

Success will be defined both in terms of the ability for the software tool to provide an optimum solution and the speed at which it can be implemented relative to other methods or benchmarks.

The business imperatives are twofold. For an engineering consultancy, the ability to rapidly, accurately and repeatably solve such problems gives a competitive edge, which can be used to win more business and to deliver designs more cheaply. For a biopharmaceutical client, optimising this problem results in cost savings and having a well defined methodology for doing so gives confidence that an in-progress design is indeed optimal and robust.

Chapter 2

Literature Review

> It was long before I got at the maxim, that in reading an old mathematician you will not read his riddle unless you plough with his heifer; you must see with his light, if you want to know how much he saw.
>
> — Augustus de Morgan, *letter to W. R. Hamilton*

2.1 Introduction

In this chapter, recent papers related to the design of bioprocess facilities are reviewed. This is followed by the review of papers related to solving similar problems in related industries. Finally, papers relating to more general methodologies for solving scheduling and optimisation problems are reviewed.

2.1.1 Bioprocess Facility Design

Current bioprocess facility design leverages software packages for creating mass balances and for schedule simulation. The mass balance tool *SuperPro Designer*® and the scheduling tool *SchedulePro*® from Intelligen, Inc. (*Scotch Plains, New Jersey, U.S.A.*) are examples of software packages that are widely used, particularly by American firms. The *INOSIM* family of software from

INOSIM Software GmbH *Dortmund, Germany* is used particularly by German and Swiss firms.

Petrides *et al.* (2014) outline a workflow for the design of a "typical" monoclonal antibody facility using *SuperPro Designer*® and *SchedulePro*® and compares the use of these packages with other methodologies. While this paper does touch on buffer preparation, it does not elaborate on a strategy for optimising the area, stating:

> In most real processes, buffer scheduling is considerably more complex and challenging because of the larger number of buffers required for a typical process (more than twenty), the shared use of pipe segments and transfer panels, as well as constraints imposed by the limited availability of labor.

Petrides *et al.* (2014) cite an earlier paper by Toumi *et al.* (2010) which also looks at design of a facility for monoclonal antibody production. Toumi *et al.* (2010) also mention the difficulty of optimising the sizing and selection of buffer equipment, but do not outline a methodology for doing so.

Dietz *et al.* (2008) outlines a genetic algorithm approach to optimising the design of protein production facilities, which is capable of dealing with imprecise demands. It is primarily looking at the specification of the main process equipment and mass balance and it does not touch on buffer preparation.

No mentions could be found of methodologies for optimising buffer preparation in bioprocess facility design.

2.1.2 Facility Design Optimisation in the Process Industry

Casting the net wider to the pharmaceutical, chemical and other process industries yields several articles that deal with similar families of problems.

Of particular relevance to buffer preparation are efforts to simulate tank farm design (Al-Otaibi *et al.*, 2004; Stewart and Trierwiler, 2005; Sharda and Vazquez,

2009; Terrazas-Moreno *et al.*, 2012). Tank farms commonly exist in large pharmaceutical, chemical and oil & gas facilities and consist of arrays of tens of tanks that are usually dedicated to particular chemicals or products, but may me multi-use. These tanks are used to support the main process being carried out in the facility, similar to the role of buffer hold vessels. There are a great deal of papers that deal with optimising throughput or productivity of existing tank farms or existing batch processes, but very little articles relating to the optimisation of the design of such processes.

Al-Otaibi *et al.* (2004) describes efforts to optimise the design of a tank farm for the oil & gas industry using both linear programming and Monte Carlo simulation methods. Their brief magazine article does not provide any detail on how the simulations were carried out.

Stewart and Trierwiler (2005) cite the work of Al-Otaibi *et al.* (2004) and outline a method for optimising tank farm design using Monte Carlo simulation. They mention the use of a software tool called *GRTMPS* from *Haverley Systems, Inc., Houston, Texas, U.S.A*, supported by *Excel* spreadsheet software and *Access* database software (*Microsoft Inc, Redmond, Washington, U.S.A*). Again, this is a short magazine article. It indicates that the scheduling produced useful results but does not give any technical detail as to how the simulation was carried out.

Sharda and Vazquez (2009) outline the use of the discrete-event simulation tool *Arena*® from *Rockwell Automation, Inc., Milwaukee, Wisconsin, U.S.A.* to optimise the utilisation of existing tank farm facilities and cite the work of Sharda and Vazquez (2009)

Terrazas-Moreno *et al.* (2012), citing Stewart and Trierwiler (2005) and Sharda and Vazquez (2009) provide a far more detailed description of efforts to optimise tank farm operations using mixed integer linear programming *MILP*. Their work looks at optimising both schedule and tank selection, but is not strictly designed with the design of the facility itself, but rather the optimal operation of a designed facility.

The work of Terrazas-Moreno *et al.* (2012) provides some useful information on techniques that could be applied to solve the problem of buffer preparation

area design, namely MILP and process scheduling. Branching out further into these fields yields more relevant literature.

Dedieu *et al.* (2003) suggests a hybrid apprach using genetic algorithms and discrete-event simulation to address the problem of multiobjective batch plant design.

In Cavin *et al.* (2004, 2005), tabu search is discussed as a methodology to optimise the design of multi-purpose batch plants.

2.1.3 Simulation and Optimisation

Casting the net wider still to look at techniques for solving generalised scheduling and optimisation problems yields far more material, but further research is required to decide which techniques, if any, are relevant to the problem at hand.

There are two aspects to optimising the design. The first aspect is scheduling, as the ability to make a tank do more than one task will depend on the times at which the tasks must (or may) occur. The second aspect is selecting the optimal sizes and numbers of vessels, which can be seen as a combinatorial optimisation problem.

In his seminal 1957 paper, George Dantzig outlines several types of combinatorial optimisation problems, one of which is the *knapsack problem*. This problem relates to finding the most valuable selection of objects that can be carried in a knapsack, subject to a total weight limit, given a selection of candidate items, each having a weight and a value. A whole range of knapsack-type problems have been researched in the intervening period. Detailed descriptions of the most common problems and the research carried out over the half century following Dantzig's paper are given by Korte and Vygen (2012) and Martello and Toth (1990).

One knapsack-type problem that may be of particular relevance is the bin-packing problem. Martello and Toth (1990) describes the bin packing problem as one in which there are a number of items with associated weights and a number of bins with associated capacities. The aim is to each item to a bin

so that the weight capacity of the bin is not exceeded and the number of bins is minimised. A number of algorithms for both exact and approximate solutions of the bin-pakcing problem have been developed. Korte and Vygen (2012) state that the bin-packing problem is strongly **NP**-hard, indicating that efforts should be taken to minimise the sample space when dealing with optimising vessel selection.

Bettinelli *et al.* (2010) investigates a particular variant of the bin packing problem where there is a minimum filling constraint. This is an important consideration in vessel selection, usually some minimum fill level, *e.g.* 20–30% is defined so that the impeller in the vessel remains sumbmerged during the mixing process and to reduce the volume of cleaning solutions or water required to clean the vessel as a fraction of the volume of buffer produced.

In terms of process scheduling methodologies, a number of papers exist in the field of chemical engineering (Ahmed and Sahinidis, 2000),

Ahmed and Sahinidis (2000) states that the general process planning problem is also **NP**-hard.

A detailed synopsis of scheduling methodologies applicable to the chemical and process industries is given by Harjunkoski *et al.* (2014).

Chapter 3

Data

> Just as the largest library, badly arranged, is not so useful as a very moderate one that is well arranged, so the greatest amount of knowledge, if not elaborated by our own thoughts, is worth much less than a far smaller volume that has been abundantly and repeatedly though over. For only by universally combining what we know, by comparing every truth with every other, do we fully assimilate our own knowledge and get it into our power.
>
> — Arthur Schopenhauer, *On Thinking for Oneself*

3.1 Introduction

This section will outline what the input data might look like, including data sources. It will outline what form the output data and metrics should take.

For modeling a single process, a typical dataset consists of three distinct files. The first file is a table of data relating to the available selection of vessels. The second file is a table of data relating to parameters specific to each buffer. The third file comprises a collection of global parameters that apply to all vessels and/or buffers.

One particular example, which is based on obfuscated data from a large-scale facility, is used throughout this chapter to illustrate the format of the data.

3.2 Vessel Data

A vessel dataset will contain entries for M vessels ($M \in \mathbb{N}$).

Typically, when designing a large-scale production facility, buffer preparation vessel volumes range from 1,000 l to 30,000 l.

When ordering such a vessel, the stated size of the vessel is usually a nominal volume, which will differ from the liquid fill volume and may also differ from the maximum working volume. It is usual to round the stated or nominal volume to the nearest 1,000 l.

For the purposes of this study, it is assumed that, for each vessel $m \in M$, the specified vessel volume, V_m is the maximum working volume (in litres) of the vessel, which is taken to be the maximum volume of buffer which can be prepared in that vessel.

Vessels will also have a minimum working volume. It is usual to assume a minimum fill ratio of about 30% of the vessel volume. This limitation arises due to the minimum agitation volume of the impeller in the vessel. For the purposes of this simulation, the minimum fill ratio is a global parameter. It may be possible to specify a value of minimum fill ratio for every vessel size, but this level of detail is typically neither required nor available. As a result, the minimum fill ratio is dealt with in the parameters section.

For each vessel, $m \in M$, a (relative) cost, c_m, must also be defined. Note that absolute costs are not required to find the vessel selection that minimises costs. Vessel costs tend not to scale linearly with volume.

Sample vessel data is given in Table 3.1. In this dataset, the cost datum for each vessel size has been estimated by raising the vessel volume (in litres) to a power of 0.6 and rounding to two decimal places. Note that the vessel volumes, V_m, are larger than the nominal volumes indicated in the 'names' column. The contents of the 'names' column are treated as identifiers and are not used in any calculation.

TODO: Alignment in table

Table 3.1: Vessel data for large-scale example

names	volumes V_m (l)	costs c_m (–)
2000 l	2222	95.64
4000 l	4444	144.96
5000 l	5556	165.72
8000 l	8889	219.71
10000 l	11111	251.19
12000 l	13333	280.23
15000 l	16667	320.37
20000 l	22222	380.73
25000 l	27778	435.28

3.3 Buffer Data

A buffer dataset will contain entries for N buffers ($N \in \mathbb{N}$). For each buffer, $n \in N$, the dataset will contain, at a minimum, its required preparation volume (in litres), U_m.

If nothing was known about the scheduling of the prodcution process, a simple simulation could be carried out without scheduling. Such a simulation would assume a fixed duration for all operations in the preparation procedure and would require knowledge of the cycle time and an additional parameter representing the maximum utilisation ratio of the preparation vessels (see Section 3.4).

For a complete model, with scheduling, we also need to know when each buffer is first required by the process, and the duration for which it is required.

For each buffer, $n \in N$, its time of first use, $t_{USE,n}$, is the time when the process draws the first drop of buffer from its hold vessel. This time is relative to some batch datum, e.g. the start of the batch or the start of downstream processing. The choice of datum is unimportant once it is consistently used.

For each buffer, $n \in N$, its duration of first use, $\Delta t_{USE,n}$, is the duration from when the process draws the first drop of buffer from its hold vessel to when

it finishes drawing the last drop of buffer from the same vessel. Note that a process may draw buffer discontinuously from a hold vessel, e.g. a cleaning buffer may be drawn from its hold vessel for a few minutes at the end of a chromatography procedure and may not be required again until near the end of another chromatography procedure a day or two later, but may not be required at all in the intervening period. Note the use of the general convention that *durations* are denoted by Δt, whereas *times* (relative to some datum) are denoted by t. It is assumed that all times and durations are in hours.

For each buffer, n, $\Delta t_{USE,n}$ must be sufficiently less than the cycle time so that there is opportunity to *turn around* its hold vessel (i.e. sufficient time must exist after use of the buffer to clean and sterilise the vessel, receive the subsequent batch of buffer and hold for a sufficient duration before use of that buffer is required by the subsequent batch).

Sample buffer data is given in Table 3.2.

Table 3.2: Buffer data for large-scale example

names	required volumes U_n (l)	use start times $t_{USE,n}$ (h)	use durations $\Delta t_{USE,n}$ (h)
Buffer #1	19676.0	733.87	48.99
Buffer #2	18528.0	644.18	36.95
Buffer #3	17346.0	684.60	49.27
Buffer #4	15055.0	764.96	28.01
Buffer #5	13896.0	628.54	52.59
Buffer #6	10267.0	728.99	53.12
Buffer #7	10070.0	677.94	20.86
Buffer #8	6797.0	728.99	53.12
Buffer #9	6716.0	612.90	70.40
Buffer #10	4632.0	645.50	35.63
Buffer #11	3780.0	729.34	43.19
Buffer #12	3694.0	678.92	42.17

3.4 Parameters

In addition to the buffer-specific and vessel-specific data detailed above, some global parameters are required to specify the problem. For the large-scale example, these parameters are tabulated in Table 3.3 and are described in more detail hereafter.

Table 3.3: Global parameters for large-scale example

symbol	short description	value	unit
λ	process cycle time	96.0	h
$\Delta t_{PREP,PRE}$	prep pre duration	11.0	h
$\Delta t_{PREP,POST}$	prep post duration	2.5	h
$\Delta t_{TRANSFER}$	transfer duration	2.2	h
$\Delta t_{HOLD,PRE}$	hold pre duration	9.0	h
$\Delta t_{HOLD,POST}$	hold post duration	1.0	h
$\Delta t_{HOLD,MIN}$	minimum hold duration	12.0	h
$\Delta t_{HOLD,MAX}$	maximum hold duration	60.0	h
$r_{MINFILL}$	minimum fill ratio	0.27	–
r_{UTIL}	maximum utilisation ratio	0.7	–

TODO: Describe parameters

Chapter 4

Methodology

> We have some freedom in setting up our personal standards of beauty, but it is especially nice when the things we regard as beautiful are also regarded by other people as useful.
>
> — Donald Knuth, *Computer Programming as an Art*

4.1 Introduction

At first glance, the pathway to solving the vessel selection problem was unclear. There are two aspects to the problem; scheduling and selection. The selection problem appears to be a close relative of a number of classic **NP**-hard problems in the "bin-packing" genre. For the scheduling aspect of the problem, initial concepts involved finding an API to one of the commercial scheduling packages. One intial concept included developing an algorithm to intelligently guess possible vessel selections, and pass these into a scheduling model which could be repeatedly be solved using a commercial scheduling package to find an optimium selection.

On completion of the module on linear programming, it became apparent that many bin-packing type problems are most efficiently solved using linear programming techniques. Additionally, linear programming could be used to model scheduling constraints.

The vessel selection problem was thus formulated as a MILP problem. Firstly, the problem was described mathematically. Next, attention was focussed on the method of solution. There are many available linear programming solvers, both proprietary and open source. Many proprietary solvers have free academic licenses. The XPRESS solver from FICO was familiar to the author from the course module in linear programming, but the documentation is poor and the licensing system required connection to university servers which proved unreliable remotely. The version avaialable through the university also had no python API (one is avaialble in more recent versions), which was a strong preference for output data analysis and plotting.

The CPLEX solver from IBM was investigated. The quality of the documentation is excellent, and there is a full python API which also has excellent documentation.

The MILP problem was thus constructed in python and then solved via the CPLEX python API. It was apparent that comparitively few lines of the code involved interface with the API. With a view to making this research as useful as possible, open-source alternatives were also investigated. In the end, the PuLP python library was used as a bridge between problem generation and problem solution. PuLP is an open-source python library which acts as an API to several (MI)LP solvers, both commercial and open source, allowing the possibility of a fully open-source toolchain for solving the problem.

4.2 Slots

To model the probem as an MILP problem, we must introduce the concept of *slots*. Noting that there is a buffer hold vessel for each buffer, it is evident that a feasible but inefficient solution could involve installing a dedicated, suitably sized, preparation vessel for each hold vessel. Any solution involving more than this number of preparation vessels would not be optimal. As a result, we have an upper limit of N on the number of required preparation vessels. The model is thus constructed with P *slots*, where

$$P = N \tag{4.2.1}$$

and a *slot* is a notional space which may be occupied by a vessel, or may be empty. Note that slots do not have any physical significance, i.e. in a physical plant, floor sapce is assgined to physical vessels, rather than notional slots.

4.3 Objective Function

The vessel selection problem may be described as a series of linear constraints. These constraints are applied to find the optimium value of an objective function, which we seek to minimise. The primary objective function is the total cost of vessels. We find this by summing the vessel costs for all vessels present in slots. Recall that the vessel data set contains entries for M vessel sizes, each of which has an associated cost, c_m.

We now introduce the first decision variable, $\boldsymbol{y}_{mp}$. This a binary decision variable with dimensions $M \times P$. Note that binary decision variables may only take the values 0 and 1, i.e.

$$\boldsymbol{y}_{mp} \in \{0, 1\} \quad \forall m \in M \quad \forall p \in P \tag{4.3.1}$$

The possible states of $\boldsymbol{y}_{mp}$ may be described thus:

$$\boldsymbol{y}_{mp} = \begin{cases} 1 \implies \text{instance of vessel } m \text{ in slot } p \\ 0 \implies \text{otherwise} \end{cases} \tag{4.3.2}$$

We can now describe the objective function, which gives the objective function value, $\boldsymbol{Z}$. The objective is to minimise $\boldsymbol{Z}$.

$$\boldsymbol{Z} = \sum_{m \in M} \sum_{p \in P} c_m \boldsymbol{y}_{mp} \tag{4.3.3}$$

4.4 Basic Model

A small number of constraints need to be applied to arrive at the simplest variant of the problem. Additional constraints may then be added to make the model more detailed or realistic.

4.4.1 Buffers Dedicated to Slots

The first constraint to be added is the limitation that a buffer must be prepared in exactly one slot. This constraint means that we must preprare each buffer once per cycle and also that the buffer is always prepared in the same slot, i.e. slot selection for every cycle is identical.

We introduce a new binary decision variable, $\boldsymbol{x}_{np}$, which takes a value of 1 if buffer n is prepared in slot p, and takes a value of 0 otherwise, i.e.

$$\boldsymbol{x}_{np} \in \{0, 1\} \quad \forall n \in N \quad \forall p \in P \tag{4.4.1}$$

Note that $\boldsymbol{x}_{np}$ has dimensions of $N \times P$. The possible states of $\boldsymbol{x}_{np}$ may be described thus:

$$\boldsymbol{x}_{np} = \begin{cases} 1 \implies \text{buffer } n \text{ prepared in slot } p \\ 0 \implies \text{otherwise} \end{cases} \tag{4.4.2}$$

The following constraint can now be defined:

$$\sum_{p \in P} \boldsymbol{x}_{np} = 1 \quad \forall n \in N \tag{4.4.3}$$

4.4.2 Vessel Instances Dedicated to Slots

The second constraint to be added is the requirement that at most one vessel may inhabit a given slot. This is not directly analagous to the first constraint; it is possible to use the same *sized* vessel in many slots, but a maximum of one vessel *instance* may inhabit any given slot. Note that this inequality allows for the possibility of unused slots, i.e. the number of occupied slots (and hence the number of preparation vessels) may be less than the number of available slots (and hence the number of buffers).

$$\sum_{m \in M} \boldsymbol{y}_{mp} \leq 1 \quad \forall p \in P \tag{4.4.4}$$

4.4.3 Vessel Capacity

The third constraint is the requirement that, if a vessel is in a given slot, it has an appropriate volume to prepare all buffers assigned to the slot. There are

two facets to the above statement. Firstly, we cannot produce buffer volumes greater than the preparation vessel maximum working volume. Additionally, vessels have a minimum fill ratio, $r_{MINFILL}$, ususally in the range 10 - 30 %. The minimum fill ratio is generally a function of the minimum stir volume of a vessel's impeller.

More complicated rules may be applied, such as the ability to make an excess of buffer to prevent the requirement for adding a smaller vessel, but such approaches are beyond the scope of this basic model.

Recall that for each vessel size, $m \in M$, we have defined a (maximum) vessel volume, V_m. Also, for each buffer, $n \in N$, we have defined the required volume, U_n.

The maximum vessel capacity constraint is:

$$U_n \boldsymbol{x}_{np} - \sum_{m \in M} V_m \boldsymbol{y}_{mp} \leq 0 \quad \forall n \in N, \quad \forall p \in P \tag{4.4.5}$$

The minimum vessel capacity constraint is slightly more complex, in that the *big-M* method must be utilised. The value used for *big-M* is V_{MAX}, where:

$$V_{MAX} = \max\left(V_m \quad \forall m \in M\right) \tag{4.4.6}$$

The minimum vessel capacity constraint is defined as:

$$V_{MAX} \boldsymbol{x}_{np} + r_{MINFILL} \sum_{m \in M} V_m \boldsymbol{y}_{mp} \leq U_n + V_{MAX} \quad \forall n \in N, \quad \forall p \in P \tag{4.4.7}$$

4.4.4 Preparation Vessel Utilisation

In the absence of detailed scheduling data (i.e. knowledge of when the buffers are used by the process and their durations of use), an allowance can be made for unknown scheduling constraints by limiting the maximum allowed utilisation of the preparation vessels. Note that this will result in a crude approximation but it is often the case that detailed timing information is not available or cannot be predicted with sufficient accuracy. We thus introduce the parameter r_{UTIL}, the maximum utilisation ratio. The total duration of all preparation

procedures in a given slot must not be above $r_{UTIL}\lambda$, where λ is the cycle time of the process.

The total duration of a single preparation procedure, $\Delta t_{PREP,TOTAL}$, is given by:

$$\Delta t_{PREP,TOTAL} = \Delta t_{PREP,PRE} + \Delta t_{TRANSFER} + \Delta t_{PREP,POST} \tag{4.4.8}$$

Note that, for the basic case, it is only strictly necessary to know $\Delta t_{PREP,TOTAL}$; the constituent durations do not need to be known individually.

The following constraint is now defined:

$$\Delta t_{PREP,TOTAL} \sum_{n \in N} \boldsymbol{x}_{np} \leq r_{UTIL}\lambda \quad \forall p \in P \tag{4.4.9}$$

4.4.5 Basic Model Summary

The basic model is summarised below.

Minimise:

$$\boldsymbol{Z} = \sum_{m \in M} \sum_{p \in P} c_m \boldsymbol{y}_{mp} \tag{4.3.3}$$

Subject to:

$$\sum_{p \in P} \boldsymbol{x}_{np} = 1 \quad \forall n \in N \tag{4.4.3}$$

$$\sum_{m \in M} \boldsymbol{y}_{mp} \leq 1 \quad \forall p \in P \tag{4.4.4}$$

$$U_n \boldsymbol{x}_{np} - \sum_{m \in M} V_m \boldsymbol{y}_{mp} \leq 0 \quad \forall n \in N, \quad \forall p \in P \tag{4.4.5}$$

$$V_{MAX} \boldsymbol{x}_{np} + r_{MINFILL} \sum_{m \in M} V_m \boldsymbol{y}_{mp} \leq U_n + V_{MAX} \quad \forall n \in N, \quad \forall p \in P \tag{4.4.7}$$

$$\Delta t_{PREP,TOTAL} \sum_{n \in N} \boldsymbol{x}_{np} \leq r_{UTIL}\lambda \quad \forall p \in P \tag{4.4.9}$$

Where:

$$\boldsymbol{x}_{np} \in \{0, 1\} \quad \forall n \in N \quad \forall p \in P \tag{4.4.1}$$

$$\boldsymbol{y}_{mp} \in \{0, 1\} \quad \forall m \in M \quad \forall p \in P \tag{4.3.1}$$

4.5 Complete Model

For a more accurate appraisal of vessel requirements, more data are required on scheduling. Specifically, data are required on the duration of use of each buffer, along with data on the time of first use of each buffer, relative to some fixed point in a batch (e.g. batch start). Given this information, it is possible to constrain the problem so that the individual preparations are scheduled correctly.

For each buffer, constraints covering both the scheduling of the preparation procedures and the scheduling of the hold procedures must be added. The scheduling of the hold procedures is quite straightforward; since the hold vessels are dedicated, we simple need to ensure that the total duration of each hold procedure is not greater than the cycle time. The scheduling of the preparation procedures is more complex, despite these procedures being of fixed duration.

4.5.1 Hold Procedure Duration

We now intoduce the buffer hold time decision variable, $\boldsymbol{z}_n$, which has dimensions of N:

$$\Delta t_{HOLD,MIN} \leq \boldsymbol{z}_n \leq \Delta t_{HOLD,MAX}; \quad \boldsymbol{z}_n \in \mathbb{R} \quad \forall n \in N \tag{4.5.1}$$

This is the only real-valued variable in the model.

The first constraint required for the complete model is the limitation that the total duration of each hold procedure must not be greater than the cycle time. If this constraint is not observed, a hold procedure in a given batch may not have finished before the hold procedure for the next batch is due to start.

$$\boldsymbol{z}_n \leq \lambda - (\Delta t_{HOLD,PRE} + \Delta t_{TRANSFER} + \Delta t_{USE,n} + \Delta t_{HOLD,POST}) \quad \forall n \in N \tag{4.5.2}$$

4.5.2 Introducing the Scheduling Constraint

The scheduling constraint may be descibed quite simply: Ensure that no two preparation operations overlap in time in a given slot.

Since we have a constant preparation duration, $\Delta t_{PREP,TOTAL}$, this constraint may be expressed, *for any two distinct buffers that are made in the same slot*, by the following:

$$|(t_{USE,k} - \boldsymbol{z}_k) - (t_{USE,n} - \boldsymbol{z}_n)| \leq \Delta t_{PREP,TOTAL} \quad \forall n \in N, \quad \forall k \in N, k > n \tag{4.5.3}$$

Note that the range of k is limited to $k > n$ to prevent duplication of constraints. The above formula is not yet in a format that can be applied in a MILP. Firstly, it was noted that the constraints only apply to two buffers which happen to be made in the same slot. Secondly, absolute value expressions are not valid in linear programming constraints. To overcome these issues, several additional constraints and variables must be introduced.

4.5.3 Pairs of Distinct Buffers Prepared in a Particular Slot

Firstly, we want to specify a binary variable which indicates if two distinct buffers are made in the same slot. This, in turn requires an additional binary variable which indicates if two distinct buffers are made in a *particular* slot. The latter binary variable, $\boldsymbol{w}_{nkp}$ is defined below.

$$\boldsymbol{w}_{nkp} \in \{0, 1\} \quad \forall n \in N \quad \forall k \in N \quad \forall p \in P \tag{4.5.4}$$

Note that the above variable has dimensions $N \times N \times P$, but we don't need each constituent binary variable to describe the problem. Since w_{nnp} is not required and $w_{nkp} = w_{knp}$, we only need to describe constraints where $k > n$.

$$\begin{aligned} \boldsymbol{x}_{np} + \boldsymbol{x}_{kp} - 2\boldsymbol{w}_{nkp} &\geq 0 \\ \boldsymbol{x}_{np} + \boldsymbol{x}_{kp} - \ \boldsymbol{w}_{nkp} &\leq 1 \end{aligned} \qquad \forall n \in N, \quad \forall k \in N, k > n \tag{4.5.5}$$

A truth table for the above constraints is given below. In the table below, $\boldsymbol{w}^{(1)}_{nkp}$ refers to the first inequality above and $\boldsymbol{w}^{(2)}_{nkp}$ refers to the second. Applying both inequalities is equivalent to performing a logical-and, i.e. $\boldsymbol{w}_{nkp} = \boldsymbol{w}^{(1)}_{nkp} \wedge \boldsymbol{w}^{(2)}_{nkp}$.

Table 4.1: Truth table for $\boldsymbol{w}_{nkp}$

$\boldsymbol{x}_{np}$	$\boldsymbol{x}_{kp}$	$\boldsymbol{w}^{(1)}_{nkp}$	$\boldsymbol{w}^{(2)}_{nkp}$	$\boldsymbol{w}_{nkp} = \boldsymbol{w}^{(1)}_{nkp} \wedge \boldsymbol{w}^{(2)}_{nkp}$
0	0	0	$\{0,1\}$	0
0	1	0	$\{0,1\}$	0
1	0	0	$\{0,1\}$	0
1	1	$\{0,1\}$	1	1

4.5.4 Pairs of Distinct Buffers Prepared in the Same Slot

Given the above constraint, we can now define a binary variable, $\boldsymbol{v}_{nk}$ which indicates if two distinct buffers are made in the same slot.

$$\boldsymbol{v}_{nk} \in \{0,1\} \quad \forall n \in N \quad \forall k \in N \tag{4.5.6}$$

This new variable is introduced via the following constraint:

$$\sum_{p \in P} \boldsymbol{w}_{nkp} - \boldsymbol{v}_{nk} \leq 0 \quad \forall n \in N, \quad \forall k \in N, k > n \tag{4.5.7}$$

The truth table for the above constraint, for a given n and k is detailed below.

Table 4.2: Truth table for $\boldsymbol{v}_{nk}$

$\boldsymbol{w}_{nk0}$	$\boldsymbol{w}_{nk1}$	$\cdots$	$\boldsymbol{w}_{nkP}$	$\boldsymbol{v}_{nk}$
0	0	$\cdots$	0	0
0	0	$\cdots$	1	0
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
1	1	$\cdots$	0	0
1	1	$\cdots$	1	1

4.5.5 Order of Preparation of a Pair of Distinct Buffers

Recall that we still cannot apply our scheduling constraint due to the presence of an absolute value expression in the equation. The absolute value expression

may be though of as representing a pair of constraints, e.g. $|\alpha - \beta| \geq \gamma$ is essentially shorthand for $\alpha - \beta \geq \gamma \quad \vee \quad \beta - \alpha \geq \gamma$. **TODO: Not strictly true... if statement ... implement case equation without a reference number** We now need to remove the logical-or ($\vee$) from the above pair of inequalities. This can be done by using the *big-M* method, whereby a large constant, $\mathbb{M}$, is used to force selection of one or other of the constraints based on the value of an additional binary. In our case, the absolute value function represents two cases. In one case, buffer n is prepared before another buffer k, and in the alternate case, buffer k is prepared after buffer n. We therefore define a binary decision variable, $\boldsymbol{u}_{nk}$:

$$\boldsymbol{u}_{nk} \in \{0, 1\} \quad \forall n \in N \quad \forall k \in N \tag{4.5.8}$$

$\boldsymbol{u}_{nk} = 0$ if n is prepared before k and $\boldsymbol{u}_{nk} = 1$ if k is prepared before n. In the edge case where the buffers are prepared at precisely the same time, the binary may take either value. **TODO: Implement case eqution here, similar to those for x and y**

We are not yet ready to define the constraint which governs the value of $\boldsymbol{u}_{nk}$. The reason for this is that it is difficult to define if one event happens before or after another when they occur repeatedly in a cyclic process. Recall that for each buffer, the scheduling data consist of a use start time and a use duration. From the point of view of the model, we are only concerned with the steady-state cyclic case, so we use a modified use start time:

$$t^*_{USE,n} = t_{USE,n} \mod \lambda \quad \forall n \in N \tag{4.5.9}$$

We want to now rigorously define whether an event, α, occurs after another event, β, iff $\alpha \mod \lambda > \beta \mod \lambda$. **TODO: double-check this logic** We are concerned with the timing of our preparation procedures. Recall that all preparation procedures have the same durations. Thus, when deciding which of two such procedures occurs first, we can use $t^*_{USE,n} - \boldsymbol{z}_n$ to mark the timing of the preparation procedure of a given buffer n. Note that this may take a negative value, which must be corrected by adding a factor of λ to bring the vaue back into the single-cycle range, as we can't use a modulo expression in

an MILP constraint. Another binary variable, $\boldsymbol{q}_n$ is introduced to indicate if $t^*_{USE,n} - \boldsymbol{z}_n < 0$:

$$\boldsymbol{q}_n \in \{0,1\} \quad \forall n \in N \tag{4.5.10}$$

Note that the value of $\boldsymbol{q}_n$ is undefined if $t^*_{USE,n} - \boldsymbol{z}_n = 0$. The above definition of $\boldsymbol{q}_n$ is captured in the following pair of constraints:

$$\begin{aligned} &\boldsymbol{z}_n - \lambda \boldsymbol{q}_n \leq t^*_{USE,n} \\ &\boldsymbol{z}_n - \lambda \boldsymbol{q}_n \geq t^*_{USE,n} - \lambda \end{aligned} \qquad \forall n \in N \tag{4.5.11}$$

TODO: Explain that q_n is used to ensure () remains between 0 and λ

The truth table for $\boldsymbol{q}_n$ is given below.

Table 4.3: Truth table for $\boldsymbol{q}_n$

$t^*_{USE,n} - \boldsymbol{z}_n$	$\boldsymbol{q}_n$
<0	0
0	$\{0,1\}$
>0	1

Having incorporated $\boldsymbol{q}_n$, we can now implement the pair of constraints which, given two distinct buffers, indicate which is prepared first.

$$\begin{aligned} &\boldsymbol{z}_n - \boldsymbol{z}_k - \lambda \boldsymbol{q}_n + \lambda \boldsymbol{q}_k + \lambda \boldsymbol{u}_{nk} \geq t^*_{USE,n} - t^*_{USE,k} \\ &\boldsymbol{z}_n - \boldsymbol{z}_k - \lambda \boldsymbol{q}_n + \lambda \boldsymbol{q}_k + \lambda \boldsymbol{u}_{nk} \leq t^*_{USE,n} - t^*_{USE,k} + \lambda \\ &\quad \forall n \in N, \quad \forall k \in N, k > n \end{aligned} \tag{4.5.12}$$

Table 4.4: Truth table for $\boldsymbol{u}_{nk}$

$\left(t^*_{USE,k} + \lambda \boldsymbol{q}_k - \boldsymbol{z}_k\right) - \left(t^*_{USE,n} + \lambda \boldsymbol{q}_n - \boldsymbol{z}_n\right)$	$\boldsymbol{u}^{(1)}_{nk}$	$\boldsymbol{u}^{(2)}_{nk}$	$\boldsymbol{u}_{nk}$
<0	$\{0,1\}$	0	0
0	$\{0,1\}$	$\{0,1\}$	$\{0,1\}$
>0	1	$\{0,1\}$	1

4.5.6 Preparation Scheduling

With the above constraint, we can finally implement the scheduling constraint. This makes use of the *big-M* method, with $\mathbb{M} = 2\lambda$.

$$\begin{aligned} \boldsymbol{z}_n - \boldsymbol{z}_k + 2\lambda\boldsymbol{u}_{nk} - 2\lambda\boldsymbol{v}_{nk} \geq \quad & t^*_{USE,n} - t^*_{USE,k} + \Delta t_{PREP,TOTAL} - 2\lambda \\ -\boldsymbol{z}_n + \boldsymbol{z}_k - 2\lambda\boldsymbol{u}_{nk} - 2\lambda\boldsymbol{v}_{nk} \geq & -t^*_{USE,n} + t^*_{USE,k} + \Delta t_{PREP,TOTAL} - 4\lambda \\ & \forall n \in N, \quad \forall k \in N, k > n \end{aligned} \tag{4.5.13}$$

Truth table:

Table 4.5: Active scheduling constraints based on values of $\boldsymbol{u}_{nk}$ and $\boldsymbol{v}_{nk}$

$\boldsymbol{u}_{nk}$	$\boldsymbol{v}_{nk}$	applicable scheduling constraint
0	0	none (buffers are not prepared in same slot)
0	1	$\left(t^*_{USE,k} - \boldsymbol{z}_k\right) - \left(t^*_{USE,n} - \boldsymbol{z}_n\right) \geq \Delta t_{PREP,TOTAL}$
1	0	none (buffers are not prepared in same slot)
1	1	$\left(t^*_{USE,n} - \boldsymbol{z}_n\right) - \left(t^*_{USE,k} - \boldsymbol{z}_k\right) \geq \Delta t_{PREP,TOTAL}$

4.5.7 Complete Model Summary

The complete model is summarised below.

Minimise:

$$\boldsymbol{Z} = \sum_{m \in M} \sum_{p \in P} c_m \boldsymbol{y}_{mp} \tag{4.3.3}$$

Subject to:

$$\sum_{p \in P} \boldsymbol{x}_{np} = 1 \quad \forall n \in N \tag{4.4.3}$$

$$\sum_{m \in M} \boldsymbol{y}_{mp} \leq 1 \quad \forall p \in P \tag{4.4.4}$$

$$U_n \boldsymbol{x}_{np} - \sum_{m \in M} V_m \boldsymbol{y}_{mp} \leq 0 \quad \forall n \in N, \quad \forall p \in P \tag{4.4.5}$$

$$V_{MAX}\boldsymbol{x}_{np} + r_{MINFILL}\sum_{m\in M} V_m \boldsymbol{y}_{mp} \leq U_n + V_{MAX} \quad \forall n \in N, \quad \forall p \in P \tag{4.4.7}$$

$$\Delta t_{PREP,TOTAL}\sum_{n\in N} \boldsymbol{x}_{np} \leq r_{UTIL}\lambda \quad \forall p \in P \tag{4.4.9}$$

$$\boldsymbol{z}_n \leq \lambda - (\Delta t_{HOLD,PRE} + \Delta t_{TRANSFER} + \Delta t_{USE,n} + \Delta t_{HOLD,POST}) \quad \forall n \in N \tag{4.5.2}$$

$$\begin{aligned} \boldsymbol{x}_{np} + \boldsymbol{x}_{kp} - 2\boldsymbol{w}_{nkp} &\geq 0 \\ \boldsymbol{x}_{np} + \boldsymbol{x}_{kp} - \boldsymbol{w}_{nkp} &\leq 1 \end{aligned} \quad \forall n \in N, \quad \forall k \in N, k > n \tag{4.5.5}$$

$$\sum_{p\in P} \boldsymbol{w}_{nkp} - \boldsymbol{v}_{nk} \leq 0 \quad \forall n \in N, \quad \forall k \in N, k > n \tag{4.5.7}$$

$$\begin{aligned} \boldsymbol{z}_n - \lambda \boldsymbol{q}_n &\leq t^*_{USE,n} \\ \boldsymbol{z}_n - \lambda \boldsymbol{q}_n &\geq t^*_{USE,n} - \lambda \end{aligned} \quad \forall n \in N \tag{4.5.11}$$

$$\begin{aligned} \boldsymbol{z}_n - \boldsymbol{z}_k - \lambda \boldsymbol{q}_n + \lambda \boldsymbol{q}_k + \lambda \boldsymbol{u}_{nk} &\geq t^*_{USE,n} - t^*_{USE,k} \\ \boldsymbol{z}_n - \boldsymbol{z}_k - \lambda \boldsymbol{q}_n + \lambda \boldsymbol{q}_k + \lambda \boldsymbol{u}_{nk} &\leq t^*_{USE,n} - t^*_{USE,k} + \lambda \\ \forall n \in N, \quad \forall k \in N, k > n & \end{aligned} \tag{4.5.12}$$

$$\begin{aligned} \boldsymbol{z}_n - \boldsymbol{z}_k + 2\lambda \boldsymbol{u}_{nk} - 2\lambda \boldsymbol{v}_{nk} &\geq t^*_{USE,n} - t^*_{USE,k} + \Delta t_{PREP,TOTAL} - 2\lambda \\ -\boldsymbol{z}_n + \boldsymbol{z}_k - 2\lambda \boldsymbol{u}_{nk} - 2\lambda \boldsymbol{v}_{nk} &\geq -t^*_{USE,n} + t^*_{USE,k} + \Delta t_{PREP,TOTAL} - 4\lambda \\ \forall n \in N, \quad \forall k \in N, k > n & \end{aligned} \tag{4.5.13}$$

Where:

$$\boldsymbol{q}_n \in \{0, 1\} \quad \forall n \in N \tag{4.5.10}$$

$$\boldsymbol{u}_{nk} \in \{0,1\} \quad \forall n \in N \quad \forall k \in N \tag{4.5.14}$$

$$\boldsymbol{v}_{nk} \in \{0,1\} \quad \forall n \in N \quad \forall k \in N \tag{4.5.6}$$

$$\boldsymbol{w}_{nkp} \in \{0,1\} \quad \forall n \in N \quad \forall k \in N \quad \forall p \in P \tag{4.5.4}$$

$$\boldsymbol{x}_{np} \in \{0,1\} \quad \forall n \in N \quad \forall p \in P \tag{4.4.1}$$

$$\boldsymbol{y}_{mp} \in \{0,1\} \quad \forall m \in M \quad \forall p \in P \tag{4.3.1}$$

$$\Delta t_{HOLD,MIN} \leq \boldsymbol{z}_n \leq \Delta t_{HOLD,MAX}; \quad \boldsymbol{z}_n \in \mathbb{R} \quad \forall n \in N \tag{4.5.1}$$

4.6 Implementation

4.6.1 Code and APIs etc

4.6.2 Goal programming

4.6.3 Plotting

Chapter 5

Results

> Bosh! Stephen said rudely. A man of genius makes no mistakes. His errors are volitional and are the portals of discovery.
>
> — James Joyce, *Ulysses*

5.1 Introduction

The equipment time utilisation plot for the sample data given in Chapter 3 is given in Figure 5.1.

TODO: generate more datasets

TODO: some stuff on complexity

TODO: time taken as a function of input size and as a function of solver choice

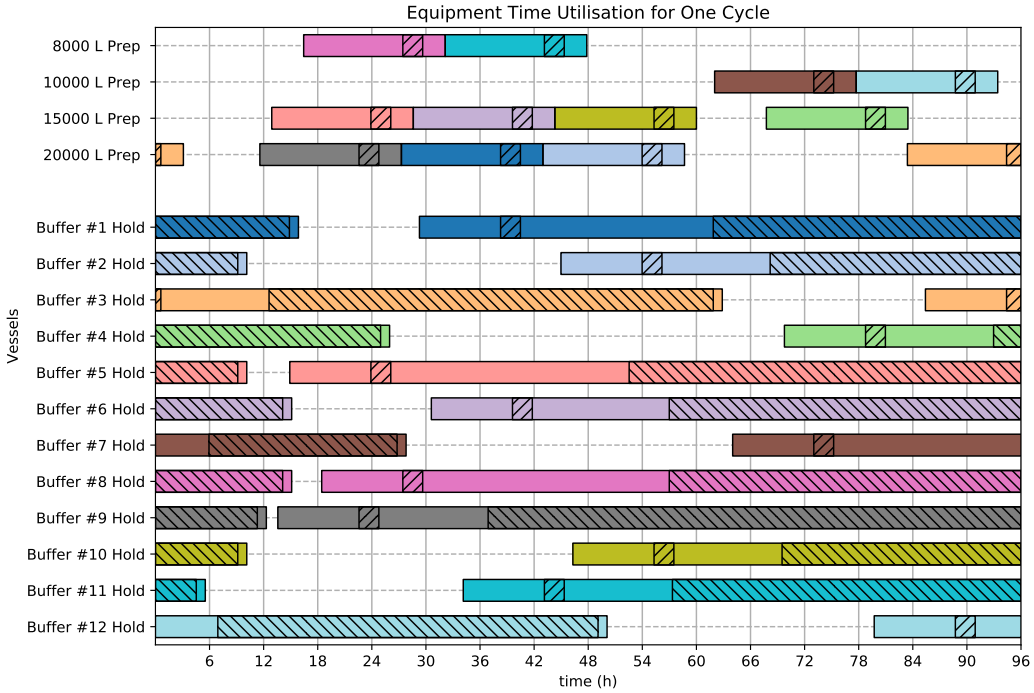

Figure 5.1: Equipment Time Utilisation for Large-Scale Example

Chapter 6

Discussion

6.1 Introduction

In this chapter we examine ...

Chapter 7

Conclusions

7.1 Introduction

The significance of ...

Detailed tables

Xyz

Program code

Xyz etc

Glossary

Entries are listed in alphabetical order.

Bibliography

Ahmed, S. and N. V. Sahinidis. 2000. Analytical investigations of the process planning problem. *Computers & Chemical Engineering*, **23**(11–12): 1605–1621. doi:10.1016/S0098-1354(99)00312-9.

Al-Otaibi, G. A., M. D. Stewart *et al.*. 2004. Simulation model determines optimal tank farm design. *Oil & gas journal*, **102**(7): 50–55.

Bettinelli, A., A. Ceselli and G. Righini. 2010. A branch-and-price algorithm for the variable size bin packing problem with minimum filling constraint. *Annals of Operations Research*, **179**(1): 221–241. doi:10.1007/s10479-008-0452-9.

Cavin, L., U. Fischer, F. Glover and K. Hungerbühler. 2004. Multi-objective process design in multi-purpose batch plants using a tabu search optimization algorithm. *Computers & Chemical Engineering*, **28**(4): 459–478. doi:10.1016/j.compchemeng.2003.07.002.

Cavin, L., U. Fischer, A. Mošať and K. Hungerbühler. 2005. Batch process optimization in a multipurpose plant using tabu search with a design-space diversification. *Computers & Chemical Engineering*, **29**(8): 1770–1786. doi:10.1016/j.compchemeng.2005.02.039.

Dantzig, G. B. 1957. Discrete-variable extremum problems. *Operations Research*, **5**(2): 266–277. doi:10.1287/opre.5.2.266.

Dedieu, S., L. Pibouleau, C. Azzaro-Pantel and S. Domenech. 2003. Design and retrofit of multiobjective batch plants via a multicriteria genetic algorithm.

Computers & Chemical Engineering, **27**(12): 1723–1740. doi:10.1016/S0098-1354(03)00155-8.

Dietz, A., A. Aguilar-Lasserre, C. Azzaro-Pantel, L. Pibouleau and S. Domenech. 2008. A fuzzy multiobjective algorithm for multiproduct batch plant: Application to protein production. *Computers & Chemical Engineering*, **32**(1–2): 292–306. doi:10.1016/j.compchemeng.2007.05.011.

Harjunkoski, I., C. T. Maravelias, P. Bongers, P. M. Castro, S. Engell, I. E. Grossmann, J. Hooker, C. Méndez, G. Sand and J. Wassick. 2014. Scope for industrial applications of production scheduling models and solution methods. *Computers & Chemical Engineering*, **62**: 161–193. doi:10.1016/j.compchemeng.2013.12.001.

Korte, B. and J. Vygen. 2012. *Combinatorial Optimization: Theory and Algorithms*. Springer-Verlag, Berlin, Heidelberg. doi:10.1007/978-3-642-24488-9.

Martello, S. and P. Toth. 1990. *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley & Sons, Inc., New York, NY, USA. ISBN 0-471-92420-2.

Otto, R., A. Santagostino and U. Schrader. 2014. Rapid growth in biopharma: Challenges and opportunities. online. Accessed 8th June 2016.
URL `http://www.mckinsey.com/industries/pharmaceuticals-and-medical-products/our-insights/rapid-growth-in-biopharma`

Petrides, D., D. Carmichael, C. Siletti and A. Koulouris. 2014. Biopharmaceutical process optimization with simulation and scheduling tools. *Bioengineering*, **1**(4): 154–187. doi:10.3390/bioengineering1040154.

Schaschke, C. 2014. *A Dictionary of Chemical Engineering*. Oxford University Press, Oxford. doi:10.1093/acref/9780199651450.001.0001.

Sharda, B. and A. Vazquez, 2009. Evaluating capacity and expansion opportunities at tank farm: a decision support system using discrete event simulation. In: *Simulation Conference (WSC), Proceedings of the 2009 Winter*, pages 2218–2224. IEEE.

Stewart, M. D. and L. D. Trierwiler. 2005. Simulating optimal tank farm design. *PTQ Magazine*, (2).

Terrazas-Moreno, S., I. E. Grossmann and J. M. Wassick. 2012. A mixed-integer linear programming model for optimizing the scheduling and assignment of tank farm operations. *Industrial & Engineering Chemistry Research*, **51**(18): 6441–6454. doi:10.1021/ie202217v.

Toumi, A., C. Jurgens, C. Jungo, B. Maier, V. Papavasileiou and D. Petrides. 2010. Design and optimization of a large scale biopharmaceutical facility using process simulation and scheduling tools. *Pharmaceutical Engineering*, **30**(2): 1–9.

List of Notation

Symbol	Description	Ref
c_m	(Relative) cost of vessel m	3.2
k	Secondary buffer index ($k \in N, k \neq n$)	4.5.3
m	Vessel size index ($m \in M$)	3.2
n	Buffer index ($n \in N$)	3.3
p	Slot index ($p \in P$)	4.2
$\boldsymbol{q}_n$	Buffer n hold operation starts in cycle prior to the one in which it finishes	4.5.5
$r_{MINFILL}$	Vessel minimum fill ratio	3.4
r_{UTIL}	Preparation slot maximum utilisation ratio	3.4
$t_{USE,n}$	Buffer n time of first use, relative to batch start	3.3
$t^*_{USE,n}$	Buffer n time of first use, relative to batch start, modulo cycle time	4.5.5
$\Delta t_{HOLD,MAX}$	Maximum allowable buffer hold duration	3.4
$\Delta t_{HOLD,MIN}$	Minimum allowable buffer hold duration	3.4
$\Delta t_{HOLD,POST}$	Duration of post-use operations in buffer hold procedures	3.4
$\Delta t_{HOLD,PRE}$	Duration of operations prior to receiving buffer in buffer hold procedures	3.4
$\Delta t_{PREP,PRE}$	Duration of operations prior to transferring out buffer in buffer preparation procedures	3.4
$\Delta t_{PREP,POST}$	Duration of operations post transferring out buffer in buffer preparation procedures	3.4
$\Delta t_{PREP,TOTAL}$	Total duration of buffer preparation procedure	4.4.4

Symbol	Description	Ref
$\Delta t_{USE,n}$	Duration of use of buffer n	3.3
$\Delta t_{TRANSFER}$	Duration of transfer from buffer preparation vessel to buffer hold vessel	3.4
$\boldsymbol{u}_{nk}$	Boolean: Buffer k is prepared before buffer n	4.5.5
$\boldsymbol{v}_{nk}$	Boolean: Distinct buffers n and k are both made in the same slot	4.5.4
$\boldsymbol{w}_{nkp}$	Boolean: Distinct buffers n and k are both made in slot p	4.5.3
$\boldsymbol{x}_{np}$	Boolean: Buffer n is in slot p	4.4.1
$\boldsymbol{y}_{mp}$	Boolean: Vessel size m is in slot p	4.3
$\boldsymbol{z}_{n}$	Buffer n hold duration	4.5.1
M	Number of vessel sizes	3.2
N	Number of buffers	3.3
P	Number of slots	4.2
U_n	Volume of buffer n to be preapred	3.3
V_m	Maximum working volume of vessel size m	3.2
V_{MAX}	Largest maximum working volume of available vessel sizes	4.4.3
$\boldsymbol{Z}$	Primary objective function value (minimal total vessel cost)	4.3
λ	Process cycle time (start–to–start duration)	3.4

Notes:

Decision variables and objective function values are in ***bold*** text.

Subscripts in lower-case represent indices. Subscripts in upper-case are descriptive. Both subscript types may be used simultaneously, e.g. in the case of $t_{USE,n}$.