CONSTRUCTING EFFICIENT PRODUCTION NETWORKS: A MACHINE LEARNING APPROACH

by

Benjamin Potter

A thesis submitted in conformity with the requirements for the degree of Master of Applied Science
Department of Mechanical and Industrial Engineering
University of Toronto

© Copyright by Benjamin Potter, 2018
Abstract

For production networks with multiple plants and products process flexibility, defined as the ability to build different types of products in the same manufacturing plant, is a key driver of operational efficiency when faced with uncertainty in demand or supply. In their seminal work on process flexibility, Jordan and Graves [1995] considered two central questions faced by the network designer: 1) How much flexibility is enough? 2) Where should flexibility be added? In this work we develop two novel Machine Learning based greedy heuristics that can be used to create efficient production network designs. We show that our heuristics perform at or above the level of the current state-of-the-art heuristics on a number of test settings from the literature. Finally, we introduce a novel application of process flexibility in healthcare operations, and use our heuristics to produce efficient network designs for the radiation therapy unit at Princess Margaret Cancer Centre (Toronto, Canada).
Acknowledgements

I want to thank my supervisor, Tim, for his continued support and mentorship. His thoughtful and dedicated approach to teaching will always be an inspiration. I have truly enjoyed my time at U of T in no small part due to your excellent supervision.

I would also like to thank my committee members Merve and Scott for the valuable feedback they provided on my thesis. Thank you to my collaborators at Princess Margaret: Daniel, Jasmine, and Tom.

My time at U of T would not have been the same without all the members of the Applied Optimization Lab, who made the days in the lab so enjoyable: Philip, Justin, Chris, Neal, Aaron, Islay, Rafid, Ian, Minha, Ben, Jonathan, Clara, Bing, and Nasrin. Thank you to Albert for all the hard work you put into this thesis.

I would like to thank my parents for all the love and support they have provided to me over the years. Anything I accomplish will always be rooted in the foundation you built for me. Lastly, thank you to Islay, for always believing in me, it is the most wonderful gift I have ever received.
Contents

1 Introduction 1

2 Literature Review 4
   2.1 Process Flexibility 4
   2.2 Machine Learning in Optimization 6
   2.3 Operations Research in Radiation Therapy Treatment
      Operations 7
      2.3.1 Patient Flow 7
      2.3.2 Patient Scheduling 7
      2.3.3 Capacity Planning 8

3 Model Development 9
   3.1 Notation and Background 9
   3.2 A Dynamic Programming Perspective for Creating Sparse Designs 11
   3.3 Predicting the Expected Max Flow of a Flexibility Design 16
   3.4 ML-based heuristics 20

4 Evaluation of Existing Methods 24

5 Radiation Therapy Case Study 33
   5.1 Problem Background 33
   5.1.1 Data 34
   5.2 Model Adaptation 37
   5.3 Redesigning the RMP treatment network 40
   5.4 Redesigning the PM Treatment Network with Homogenous LINACs 45
   5.5 Reducing Variability with Even Use Initial Schedules 46
List of Tables

4.1 Flexibility Design Heuristics Expected Max Flow ........................................... 28

5.1 Conceptual correspondence between manufacturing and a radiation therapy clinic .... 34
5.2 Comparison of Network Designs with Heterogenous LINACs ............................. 43
5.3 Difference in Number of Arcs Between 56 arc PSRH Network and Existing PM Network . 44
5.4 Comparison of Network Designs with Homogenous LINACs ............................. 46
5.5 Comparison of Network Designs with Even Use Scheduling Policy ..................... 48

A.1 Patient Groups and Demands ............................................................................. 61
A.1 Patient Groups and Demands ............................................................................. 62
# List of Figures

3.1 Early Closing of The Chain ................................................................. 13
3.2 Heuristic Pathways from time $N-2$ to $N$ ........................................... 15
3.3 Transformation of a Flexibility Design to an Incidence Matrix ................ 17
3.4 Convolutional Neural Network Architecture ........................................ 18
3.5 Predictions of Expected Max Flow for Flexibility Designs ...................... 20
3.6 Steps of the Predict and Search Algorithm ......................................... 21
3.7 Steps of the Predict and Search with Revisionist History Algorithm .......... 23

4.1 Initial Flexibility Designs ................................................................. 26
4.2 Expected Max Flow vs Number of Arcs ............................................... 27
4.3 Pairwise Comparison of Network Designs ........................................... 29
4.4 Distribution of Expected Max Flow for Final Network Designs ................. 30
4.5 Comparison of ML-based Heuristic Parameters ..................................... 31

5.1 Existing PM Treatment Network ......................................................... 36
5.2 Historical Breakdown Lengths ............................................................ 37
5.3 Heterogeneous RMP Simulated Network Data ....................................... 41
5.4 Heuristic Design Performance Progression ......................................... 42
5.5 Variability with respect to The Initial Schedule .................................... 43
5.6 Homogenous LINAC Network Design ................................................ 45
5.7 Comparison of Expected Max Flow w.r.t Initial Scheduling Decisions Even Use of LINACs 48
5.8 Performance Percentage of PSRH-designed 80 arc Network to Fully Flexible Network . . . 49
5.9 Expected Overtime of Treatment Networks ......................................... 50

A.1 Data Sources for RMP Production Network ......................................... 60

B.1 Initial PM Treatment Network ............................................................ 64
Chapter 1

Introduction

Manufacturing process flexibility is a key driver of operational performance when faced with uncertainty in demand or supply in a multiproduct and multiplant production network. Process flexibility has been well-studied in areas ranging from automobile manufacturing [Jordan and Graves, 1995] to call centers [Iravani et al., 2007]. Numerous organizations have adopted process flexibility to better respond to market changes without significantly increasing cost [Simchi-Levi, 2010].

In the manufacturing setting, process flexibility is defined as the ability to build different types of products in the same manufacturing plant or production line. For a production network, a flexible design comprises the set of decisions determining which products each plant should manufacture. In the fully flexible case, every plant is able to manufacture every product. However, incorporating full flexibility into a network can be prohibitively expensive and inefficient. Sparse flexible designs are those that have a small number of product to plant assignments relative to the fully flexible case. Additionally, production networks studied through the lens of process flexibility are primarily classified according to two main characteristics — balance and symmetry. A network is balanced if there are an equal number of plants and products. A network is symmetric if the demand for each product is independently and identically distributed.

In this thesis, we provide the first look at how machine learning (ML) can be used to solve the flexible design problem, a classical problem from the operations research and management science communities. In particular, we develop a new ML-based approach for constructing sparse flexible designs in general (i.e., not necessarily balanced or symmetric) production networks.

When incorporating flexibility into a system, there are two central questions faced by the system designer [Jordan and Graves, 1995]: 1) How much flexibility is enough? 2) Where should flexibility
be added? Much subsequent research has focused on studying these two questions, primarily from a theoretical standpoint, to understand the characteristics of flexible designs [Graves and Tomlin 2003, Akşin and Karaesmen 2007, Chou et al. 2010b, 2011, Bassamboo et al. 2012, Simchi-Levi and Wei 2015, Wang and Zhang 2015, Désir et al. 2016]. With a few notable exceptions (e.g., Deng and Shen [2013]), these foundational theoretical results, such as the optimality of the long chain among all networks with node degree two [Simchi-Levi and Wei 2012], have been established in the case of a symmetric, balanced network.

In the general case of an asymmetric, unbalanced network, the focus tends to be on developing constructive approaches for designing sparse flexible networks [Mak and Shen 2009, Chou et al. 2011, Simchi-Levi and Wei 2015, Feng et al. 2017, Yan et al. 2017, Chen et al. 2018]. While diverse in their specifics, they share a common conceptual framework of computing a proxy for the performance of unseen networks, such as the plant cover index [Simchi-Levi and Wei 2015] or dual prices obtained from a related conic program [Yan et al. 2017], and using those estimates in a greedy approach to augmenting the current design. Accordingly, the performance of these algorithms is largely related to 1) how well the value of a network can be estimated, and 2) the size of the neighborhood of networks that can be evaluated.

The focus of this paper is to add to the rich literature on process flexibility by developing a constructive, ML-based approach to generate sparse flexible designs. Modern ML methods, coupled with the growing availability of high-performance computing resources, are well-suited to addressing the challenges outlined above. In particular, deep neural networks have demonstrate impressive accuracy gains in difficult problems such as speech recognition [Hinton et al. 2012] and image recognition [Krizhevsky et al. 2012]. Furthermore, once a neural network model is trained, it can generate predictions for huge numbers of out-of-sample observations very quickly. These potential advantages suggest that ML-based predictions of network performance may be useful in design heuristics. In particular, we re-cast the problem of creating a sparse flexible design as a dynamic programming problem and use the ML-based predictions as value function approximations. We unlock these advantages by transforming a general network design into an image-like data structure, which allows us to efficiently implement a prediction model. We test our approach against the current state-of-the-art approaches on both symmetric and asymmetric networks that have been studied previously in the literature.

Finally, we introduce a novel application of process flexibility in healthcare and demonstrate our approach on a large, asymmetric and unbalanced cancer treatment network. Radiation therapy (RT), used in over 50% of cancer patients, is a one of the main ways to treat cancer [Delaney et al. 2005]. It typically involves delivering high-energy x-rays to a tumor using a linear accelerator. Linear accelerators...
can be thought of as flexible plants in a production network because, with a few exceptions, they can treat tumors in most parts of the body. Different disease sites (prostate, breast, etc.) form the demand nodes or products. In cancer treatment centers, disease sites may be assigned to a subset of the linear accelerators, indicating which patients are treated on which accelerators. Using data from Princess Margaret Cancer Centre, which is the largest single-site radiation treatment facility in North America and one of the largest in the world, we study flexibility from the perspective of machine downtime (i.e., capacity) uncertainty. In particular, we demonstrate how our ML-based approach can identify sparse networks with comparable performance to Princes Margaret’s existing network. We also demonstrate that an ML-based approach can easily account for clinical constraints that may be difficult to embed within other design approaches. In this thesis we make the following contributions:

1. We propose a dynamic programming formulation to create sparse designs for the process flexibility problem.

2. We develop a novel ML-based heuristic that produces sparse flexibility designs that perform better than, or at the level of, the existing state-of-the-art methods, without relying on the deep theoretical insights that existing heuristics are based upon.

3. We introduce a novel healthcare application of process flexibility that we use to demonstrate the effectiveness of our approach on a large-scale, real-life problem instance and incorporate context-specific constraints. We use our framework to answer relevant clinical questions regarding the design of a sparse treatment network and generate insights that we can extend to the general manufacturing setting.

Overall, the goal of this thesis is to bring together the management science and machine learning communities by demonstrating the potential impact that ML can have on classical management science problems such as flexible system design.
Chapter 2

Literature Review

Our work is related to three streams of literature: 1) process flexibility, 2) the application of machine learning to predict optimal values of optimization problems, and 3) applications of operations research to study radiation therapy treatment operations.

2.1 Process Flexibility

The seminal work of Jordan and Graves [1995] was the first to study the value of partial flexibility in a production system. Through numerical analysis, the authors demonstrated that adding some flexibility to key parts of a system could yield most of the benefits of a fully flexible system. The most celebrated result from this paper comes from the balanced case where the number of plants equal number of products. In particular, the authors discovered that by assigning products to plants such that a path could be traced from any plant or product to any other plant or product by following assignments, the expected sales would be close to that of a fully flexible system. This concept is referred to as chaining, and the structure formed by connecting all products and plants in a Hamiltonian cycle is known as the long chain. While the authors extended their analysis to the unbalanced case, the primary finding is that a bit of flexibility can capture most of the benefits of a fully flexible design, chiefly demonstrated by the effectiveness of the long chain.

Building on the seminal work of Jordan and Graves [1995] there have been two streams of subsequent research. The first includes applications and extensions of the study of process flexibility to more complex systems. The second stream consists of papers that study the value of process flexibility from a theoretical standpoint through rich mathematical analysis. A pertinent subset of this research leverages theoretical insights into the characteristics of flexible designs to develop heuristics that can be used to design flexible
designs in the general case.

In the first stream, researchers have studied process flexibility in a wide range of settings including multi-stage systems [Graves and Tomlin, 2003], queuing systems [Sheikhzadeh et al., 1998, Gurumurthi and Benjaafar, 2004, Iravani et al., 2005, Bassamboo et al., 2012, Tsitsiklis and Xu, 2012], serial production lines [Hopp et al., 2004], call centers [Wallace and Whitt, 2005], and sports analytics [Chan and Fearing, 2018].

From a theoretical standpoint, there have been a number of papers that develop frameworks for better understanding the value of sparse flexibility. Earlier papers approached the problem optimizing for expected performance of the system [Akşin and Karaesmen, 2007, Chou et al., 2010a,b, Simchi-Levi and Wei, 2012]. In particular, Simchi-Levi and Wei [2012] proved that the long chain is the optimal design with respect to expected sales among all 2-flexible designs, those where the degree of each node is two. More recently, analysis has focused on developing theory to understand flexible structures in terms of worst-case performance [Chou et al., 2011, Simchi-Levi and Wei, 2015, Wang and Zhang, 2015, Yan et al., 2017].

Most closely related to the methods presented in this thesis are a set of papers that build, from deep theoretical insights, heuristics for producing designs in the general case. With the notable exception of [Chen et al., 2018], the heuristics proposed in the literature are constructive in nature, greedily adding or subtracting one link at a time to the network in order to create a final design. Adapting the concept of expanders from graph theory, Chou et al. [2011] match product demand to plant capacity to continually improve the worst expansion ratio: the amount of capacity connected to a product and the amount of demand connected to a plant. Simchi-Levi and Wei [2015] choose the next assignment such that it relieves the largest bottleneck in the system. Feng et al. [2017] solve a relaxation of a stochastic integer programming problem and use the optimal solution to determine the next assignment. Finally, Yan et al. [2017] begin with a fully connected network and iteratively delete arcs using dual prices obtained from a related conic program. The authors note, however, that their method can be easily adapted so that it adds links to an initial design. Notably, Deng and Shen [2013] also provide a set of guidelines to consider when creating flexible designs for unbalanced networks.

The existing constructive methods for creating flexible designs are each built upon deep mathematical insights into the process flexibility problem. In this thesis, we are the first work to use an ML-based approach, which does not rely on any theoretical analysis of the process flexibility problem.
2.2 Machine Learning in Optimization

Machine learning, and specifically deep learning, has garnered much attention recently in both the literature and media. Deep learning models have been successfully used to solve a wide variety of difficult problems in image recognition [Krizhevsky et al., 2012], speech recognition [Hinton et al., 2012], language translation [Sutskever et al., 2014], and medical image analysis [Litjens et al., 2017]. We refer the reader to [LeCun et al., 2015] for a comprehensive overview of the recent deep learning literature.

There has been a long history of research conducted at the intersection of ML and Operations Research (OR) solution methods. In the 1980’s and 1990’s there was a considerable effort made to apply ML techniques to solving discrete optimization problems such as the travelling salesman problem [Hopfield and Tank, 1985]. In her review article, Smith [1999] provides a detailed review of this body of literature noting difficulties previous researchers have had in part due to lack of computational resources. After the time of Smith’s review article, this area of research was largely inactive prior to a recent renewed interest sparked by the advancements in ML.

The contemporary research falls into two categories. The first is a revival of techniques that aim to use ML as an alternative to optimization. The second category, which is more closely related to our work, is characterized by the integration of ML into traditional OR methodologies.

Approaches that use ML in lieu of traditional OR solution methods attempt to build models that directly predict the value of the decision variables in the optimal solution. The current research covers a variety of ML methods including reinforcement learning models [Bello et al., 2016, Dai et al., 2017], permutation invariant pooling networks [Kaempfer and Wolf, 2018], and pointer networks [Vinyals et al., 2015]. Closest to our work is [Larsen et al., 2018], who propose a framework for generating descriptions of optimal solutions (ranging from objective value to decision variable values) using only partial information and apply the method to solve a load planning problem in the railway industry. Similar to our application, the authors identify a need for online fast approximation of an optimization problem as the main requirement that dictates an ML approach to the problem.

In our work, we leverage the fast approximation of a stochastic max flow problem to create a less myopic greedy heuristic for a network design problem. There has been a long history of incorporating approximation methods to estimate cost-to-go functions in dynamic programming (DP) [Powell, 2007]. In this thesis, we borrow from the DP framework in the presentation of our heuristics; however, we use a prediction of a given state’s expected max flow in its current form instead of the more traditional cost-to-go functions. Similar prediction models have been developed. [Fischetti and Fraccaro, 2017] predict the objective value of an offshore wind farm layout optimization problem using ML. [Seward, 2017] predicts
Chapter 2. Literature Review

the objective value of a travelling salesman problem, which he then uses to optimize the division of tasks between workers in a warehouse setting.

2.3 Operations Research in Radiation Therapy Treatment

Operations

It is projected that by 2030, 24.6 million new cases of cancer will be reported annually worldwide [Atun et al., 2015]. This is anticipated to increase the demand for radiation therapy, which is currently used to treat or palliate symptoms in 50% of cancer patients [Delaney et al., 2005]. In addition, technological advances aimed at improving treatment outcomes increase the complexity and time taken to deliver treatment, placing additional demands on RT workload [Mou et al., 2011]. These challenges require that RT clinics be able to effectively respond to increased demands in part by improving efficiency. Operations researchers have begun to address these problems by applying OR methodologies to study resource planning in RT clinics. This research has largely focused on three areas: patient flow, patient scheduling, and capacity planning. For a comprehensive review of the literature of applications of operations research to radiotherapy please see Vieira et al. [2016].

2.3.1 Patient Flow

Patients arriving at RT clinics must pass through several preliminary stages before receiving their first treatment. These stages include localization of treatment fields using a CT simulator, RT planning to determine dosage and treatment delivery method, and plan verification. The majority of papers studying the flow of patients between these stages use computer simulation models to identify bottlenecks [Thomas, 2003; Aitkenhead et al., 2012; Price et al., 2013]. Werker et al. [2009] showed that reducing variability in oncologist schedules could lead to a 20% decrease in patients’ waiting time, and Joustra et al. [2012] showed that it was possible to increase the percentage of patients beginning treatment within 21 days from 39% to 92% by increasing capacity in the outpatient department.

2.3.2 Patient Scheduling

Scheduling is the largest area of study within RT clinical operations. Indeed, 18 of the 33 papers reviewed by Vieira et al. [2016] study problems related to patient scheduling. In particular, 12 of the 18 papers study scheduling patients on linear accelerators (LINACs). Kapamara et al. [2006] concluded that scheduling patients for RT treatments is similar to a dynamic stochastic job shop problem. Subsequently,
many papers have studied the patient scheduling problem by formulating integer programming models, for example Conforti et al. [2008], Petrovic and Leite-Rocha [2008], Petrovic et al. [2009], and Burke et al. [2011]. In the majority of cases, these papers focus on gaining operational insights that can help reduce patient wait times by evaluating various scheduling policies using the developed models. Two papers have extended the literature through a stochastic treatment of the patient scheduling problem. Saure et al. [2012] developed an infinite horizon discounted Markov decision process model for creating patient schedules and Legrain et al. [2015] developed an approach that combines stochastic optimization and online optimization.

Fundamentally, the existing RT patient scheduling literature focuses on developing policies or tools at the tactical level that aim to efficiently use a clinic’s resources in their current configuration. In this work, our analysis is conducted at the strategic level, analyzing the match between LINAC capacity and patient demand for RT treatments.

2.3.3 Capacity Planning

Resource capacity planning in radiation therapy has received some attention in the literature Ogulata et al. [2009], Joustra et al. [2010], Bikker et al. [2015], Li et al. [2015]. With the exception of Bikker et al. [2015], these papers study capacity issues related to LINACs, which Li et al. [2015] identified as having the largest impact on patient wait times. In their analysis, previous researchers have used a combination of queueing theory and computer simulation to study the impact of slack capacity (Ogulata et al. [2009]), effect of pooling resources for urgent and regular patients (Joustra et al. [2010]), and the number of required time slots (Li et al. [2015]). A critical difference between the existing literature and our work is that we study the LINAC resource planning problem accounting for the impact of uncertainty in LINAC capacity.
Chapter 3

Model Development

In this chapter, we develop a novel ML-based heuristic approach for producing sparse designs for a production network. Our presentation follows the framework that has been used in the general manufacturing process flexibility literature. A single instance of the problem consists of a set of plants and a set of products, where the demand for each product is modeled by some probability distribution. The products, plants and assignments specifying which products can be manufactured by which plants constitute the production network. Good network designs should maximize demand met while limiting the size of the set of assignments. We refer to these two objectives as efficiency and sparsity, respectively.

3.1 Notation and Background

We consider a system with $m$ plants and $n$ products. We denote the set of plants and products by $I = \{1, \ldots, m\}$ and $J = \{1, \ldots, n\}$, respectively. Let $c = (c_1, \ldots, c_m)$ and $d = (d_1, \ldots, d_n)$ be vectors representing the supply and demand at the plant and product nodes, respectively. We assume demand is uncertain and drawn from some distribution $D$, while supply is fixed. A network design is encoded by a set of arcs $A \subseteq I \times J$—if product $j$ can be manufactured at plant $i$, then $(i, j) \in A$—in a bipartite graph defined on $I$ and $J$. Our objective is to construct a sparse design $A$ that maximizes expected demand met in the production network, which is equivalent to maximizing the expected maximum flow of the corresponding network flow problem. If there are no constraints on the design of the network, then clearly the fully flexible network, $F := I \times J$, is optimal. Our focus on sparse designs means that the number of arcs in a design will be bounded by some constant $N < nm$.

Given an instance of the demand vector $d$, the maximum flow of a design $A$, denoted by $\mu(A, d)$, can be determined by the following linear optimization problem:
\[ \mu(A, d) := \text{maximize} \sum_{(i,j) \in A} f_{ij} \]
subject to \[ \sum_{j: (i,j) \in A} f_{ij} \leq c_i, \quad \forall i \in I, \]
\[ \sum_{i: (i,j) \in A} f_{ij} \leq d_j, \quad \forall j \in J, \]
\[ f_{ij} \geq 0, \quad \forall (i,j) \in A. \] (3.1)

Where \( f_{ij} \) denotes the amount of product \( j \) produced at plant \( i \). The expected value of the max flow of a flexible design \( A \) is
\[ \mathbb{E}_{d \in D}[\mu(A, d)] \] (3.2)
and the problem of maximizing the expected max flow over a set of possible network designs \( \mathfrak{A} \) is written as
\[ \max_{A \in \mathfrak{A}} \mathbb{E}_{d \in D}[\mu(A, d)]. \] (3.3)

Problem (3.3) represents the sparse flexible design problem. A natural approach to formulating problem (3.3) would be as an integer optimization problem. Let \( x_{ij} \) be 1 if \((i,j) \in A\) and 0 otherwise. Approximating the expectation using \( K \) realizations of \( d \) drawn from \( D \), and replicating the flow variables, \( f_{ijk} \) for each demand instance \( k \), we get
\[ \text{maximize} \quad \frac{1}{K} \sum_{k=1}^{K} \sum_{(i,j) \in A} f_{ijk} \]
subject to \[ \sum_{j: (i,j) \in A} f_{ijk} \leq c_i, \quad \forall i \in I, k = 1, \ldots, K, \]
\[ \sum_{i: (i,j) \in A} f_{ijk} \leq d_{jk}, \quad \forall j \in J, k = 1, \ldots, K, \] (3.4)
\[ f_{ijk} \leq c_i x_{ij}, \quad \forall (i,j) \in A, k = 1, \ldots, K, \]
\[ x \in X(\mathfrak{A}), \]
where \( X(\mathfrak{A}) \) is a set of constraints on \( x \) that enforce the possible network configurations allowed by \( \mathfrak{A} \).

Our main focus is on designs with exactly \( N \) arcs, i.e., where \( \mathfrak{A} = \{ A \subseteq \mathcal{F} \mid |A| = N \} \). In this case, we can write \( X(\mathfrak{A}) = \{ x \mid \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij} = N \} \). Other constraints such as limiting the degree to each node or enforcing specific sub-designs can also be modeled easily in the integer optimization paradigm.

We let \( S \) denote the set of \( K \) demand realizations and with a slight abuse of notation we define \( \mu(A, S) \) to be the estimated value of (3.2).
While formulation (3.4) provides an exact approach for the sparse design problem, it is an integer programming problem, which may be difficult to solve for large instances. Feng et al. [2017] found that when the number of demand realizations is large, e.g., $K = 10,000$, integer programming techniques were unable to find satisfactory solutions even for a small networks with 5 plants and 10 products. Thus, much of the literature on creating sparse designs for the general case of an unbalanced production network focuses on developing heuristics that use metrics based on in-depth mathematical analysis to build the network one arc at a time.

### 3.2 A Dynamic Programming Perspective for Creating Sparse Designs

Given that the approaches in the literature to solving formulation (3.3) tend to be constructive in nature, typically adding one arc at a time to an existing network, we propose an alternative formulation of the sparse design problem. Instead of integer optimization, we formulate the sparse network design problem as a dynamic program. We believe that viewing the problem from a dynamic programming (DP) perspective allows us to unify and contrast the existing heuristic approaches within a common framework. Admittedly, the approaches we discuss do not provide direct estimates of the cost-to-go function, and instead are more myopic focusing on maximizing the value of the next state. Moreover, while not considered in this thesis, we will argue that our method could be adapted to more closely fit with the true DP framework by making a minor methodological adjustment to the prediction model developed later.

Since the primary constraint in our design problem is that the number of arcs added is at most $N$, we define the state space at each stage $l$ as a function of the network size, characterized by the number of arcs included in the current design. Without loss of generality in this section, we assume that the initial arc set is empty. Therefore the state space at each stage $l$ is given by:

$$Q_l = \{ A \subseteq \mathcal{F} \mid |A| = l \}.$$  

The possible actions in each state $A_l \in Q_l$ consist of the arcs that can be added to design $A_l$, which we write as $C_{A_l} = \mathcal{F} \setminus A_l$. Each action in $C_{A_l}$ corresponds to a particular arc $(i, j) \not\in A_l$; choosing this action at stage $l$ results in a deterministic transition to the new state $A_{l+1} = A_l \cup (i, j)$. Finally, we define the value of being in state $A_l$ as $J_l(A_l)$.

Viewing the terminal state as a network design with $N$ arcs, we want it to have value equal to
$E_{d \in D}[\mu(A_N^*, d)]$, where $A_N^*$ is a network with $N$ arcs that maximizes the expected max flow. Thus, the optimal sparse network design can in principle be determined by solving the following DP:

$$J_N(A_N) = E_{d \in D}[\mu(A_N^*, d)], \quad A_N \in Q_N,$$

(3.6a)

$$J_l(A_l) = \max_{a_l \in C_{A_l}} J_{l+1}(A_l \cup a_l), \quad A_l \in Q_l, l = 0, \ldots, N - 1.$$  

(3.6b)

Like the integer optimization model, this DP is hard to solve for large-scale instances due to the curse of dimensionality. More fundamentally, we cannot solve this recursion since we do not know $A_N^*$ ahead of time. Nevertheless, we believe this is a useful lens through which to view the sparse flexible design problem. The benefit of the DP formalism is that it allows us to easily represent the design heuristics in the literature, which can generally be classified as greedy policies that add the “best” arc at each stage according to some cost function. In other words, these methods can be thought of as replacing $J_{l+1}(A_{l+1})$ with some approximation $\tilde{J}_{l+1}(A_{l+1})$. These methods differ in their estimation of the value of the next arc to add, which can be seen as different approximations of the value of subsequent flexibility designs, $\tilde{J}_{l+1}(A_{l+1})$. In summary, the existing heuristics boil down to solving the following optimization problem at each stage:

$$\max_{a_l \in C_{A_l}} \tilde{J}_{l+1}(A_l \cup a_l)$$

(3.7)

In other words, at stage $l$, a greedy design heuristic aims to identify an action that improves the surrogate measure of the value function the most.

It is well-known that greedy methods are susceptible to short-sightedness. The process of iteratively adding arcs to the flexibility design problem is especially vulnerable to this trap. In particular, for the balanced and symmetric case, Simchi-Levi and Wei [2012] showed the largest incremental improvement in the expected max flow occurs when an arc is added to form a closed chain. For example, suppose starting from a dedicated production network with 10 plants and 10 products we want to greedily add arcs to get an optimal design with 20 arcs, which we know to be the long chain. At an intermediate step we have added 5 links that form an open chain, but greedily adding the next arc to maximize expected max flow will form a closed chain rather than continuing the open chain (see Figure 3.1). Once the chain is closed, we won’t be able to achieve the long chain. In this way, the extra increase gained by closing the chain can lead short-sighted methods to make early mistakes that drive the final solution away from optimality.

In this thesis, we consider two potential antidotes to the short-sightedness of the greedy heuristic. First, at each stage, we consider a larger neighbourhood of possible designs, equivalent to adding arcs
in batches rather than one at a time. Second, we allow our approach to re-evaluate previously added arcs at each iteration to possibly correct "mistakes". These countermeasures to shortsightedness can be understood in terms of modifications to the optimization problem given in (3.7).

In the DP framework, adding arcs in batches changes the space of possible actions at each stage $l$. Given a batch size, $q$, the action space at stage $l$ includes all possible combinations containing exactly $q$ unique arcs that are not currently part of $A_l$:

$$C^q_{A_l} = \{ (a_l, \ldots, a_{l+q-1}) \mid a_l, \ldots, a_{l+q-1} \in C_{A_l}, a_l \neq a_{l+1} \neq \ldots \neq a_{l+q-1} \}.$$  

Choosing an action $(a_l, \ldots, a_{l+q-1})$ results in a deterministic transition from state $A_l$ to state $A_l \cup \{a_l\} \cup \ldots \cup \{a_{l+q-1}\} \in Q_{l+q}$. Then the optimization problem becomes:

$$\max_{(\{a_l\}, \ldots, \{a_{l+q-1}\}) \in C^q_{A_l}} J_{l+q}(A_l \cup \{a_l\} \cup \ldots \cup \{a_{l+q-1}\}).$$ (3.8)

Solving (3.8) provides more information at each step compared to one-arc greedy heuristics, but the algorithm may still be prone to making short-sighted errors, such as prematurely closing the chaining structure. In fact, short-sighted errors will persist as long as $q < N$. However, a mechanism that allows the algorithm to reverse previous actions provides an opportunity to overcome short-sighted errors. We refer to this concept as *revisionist history*.

To adapt the concept of revisionist history to the DP framework at each stage $l$, we define a set of previously added arcs, i.e., actions taken to arrive at $A_l$, to be $B_l$. We then allow for one arc from $B_l$ to be removed from $A_l$ and be replaced with a batch of $q + 1$ arcs. Consequently at stage $l$ with sets $A_l$...
and $B_l$ we define the set of possible actions to be:

$$C_{A_l}^q(B_l) = \{ (b_l, \alpha_l-1, \ldots, \alpha_l+q-1) \mid b_l \in B_l, \alpha_l-1, \ldots, \alpha_l+q-1 \in C_{A_l}, \alpha_l-1 \neq \alpha_l \neq \ldots \neq \alpha_l+q-1 \}.$$  

Following from above, when action $(b_l, a_l-1, \ldots, a_l+q-1) \in C_{A_l}^q(B_l)$ is taken in state $A_l$, there is a deterministic transition to state $A_l \{b_l\} \cup \{a_l-1\} \cup \ldots \cup \{a_l+q-1\}$. Then the optimization problem becomes:

$$\max_{(b_l, a_l-1, \ldots, a_l+q-1) \in C_{A_l}^q(B_l)} \tilde{J}_{l+q-1}(A_l \{b_l\} \cup \{a_l-1\} \cup \ldots \cup \{a_l+q-1\}).$$  

(3.9)

We refer to the classes of heuristics defined by solving a sequence of optimization problems defined by (3.7), (3.8), and (3.9) as one-arc greedy, batch greedy, and batch greedy with revisionist history, respectively. Figure 3.2 provides a conceptual diagram to illustrate the pathway taken by each of the heuristics from stage $N-2$ to $N$. Each node in the graph represents a design at the corresponding stage. Within each stage, nodes are arranged in descending order of value (expected max flow) from top to bottom and possible actions from each node are indicated by the dashed lines. Each heuristic starts in the same green node at time $N-2$ and ends at the checkered node at time $N$.

As the heuristic pathways in Figure 3.2 suggest, we may expect the batch greedy with revisionist history heuristic to outperform the batch greedy heuristic, which in turn outperforms the one-arc greedy heuristic. This result is formalized in Theorem 1 by considering policies taken by each heuristic to transition from state $A_l$ to state $A_{l+q}$, $q \geq 1$.

**Theorem 1.** Let $\pi^*_G$, $\pi^*_B$, and $\pi^*_{BRH}$ denote the optimal sequence of actions taken to transition from state $A_l$ to $A_{l+q}$ using the one-arc greedy, batch greedy, and batch greedy with revisionist history policies, respectively. Then for any state $A_l$

$$\tilde{J}^*_G(A_l) \leq \tilde{J}^*_B(A_l) \leq \tilde{J}^*_{BRH}(A_l).$$

**Proof.** Suppose $q$ applications of the one-arc greedy policy from a state $A_l$ results in the sequence of actions $\pi^*_G = (a^G_l, a^G_{l+1}, \ldots, a^G_{l+q-1})$, where

$$a^G_{l+i} \in \arg \max_{a_{l+i} \in C_{A_{l+i}}} \tilde{J}_{i+1}(A_{l+i} \cup \{a_{l+i}\}), \quad i = 0, \ldots, q-1.$$  

The policy $\pi^*_G$ is a feasible but not necessarily optimal solution to the batch greedy problem:

$$\max_{(a_l, \ldots, a_{l+q-1}) \in C_{A_l}^q} \tilde{J}_{l+q}(A_l \cup \{a_l\} \cup \ldots \cup \{a_{l+q-1}\}).$$
Chapter 3. Model Development

15

(a) One-arc Greedy

(b) Batch Greedy

(c) Batch Greedy with Revisionist History

Figure 3.2: Heuristic Pathways from time $N - 2$ to $N$
Hence, $\tilde{J}_{\pi^*_B}(A_l) \leq \tilde{J}_{\pi^*_G}(A_l)$. Similarly, given the optimal batch greedy policy of size $q$ at state $A_l$, $\pi^*_B = (a_l^{\pi^*_B}, a_{l+1}^{\pi^*_B}, \ldots, a_{l+q-1}^{\pi^*_B})$, we can construct a feasible solution to the batch greedy with revisionist history heuristic with batch size $q + 1$,

$$\max_{(b_l, a_{l-1}, \ldots, a_{l+q-1}) \in C_{A_l}(B_l)} \tilde{J}_{l+q}(A_l \setminus \{b_l\} \cup \{a_{l-1}\} \cup \ldots \cup \{a_{l+q-1}\}),$$

by choosing $\pi_{BRH} = (b_l, a_{l-1}, a_l^{\pi^*_B}, a_{l+1}^{\pi^*_B}, \ldots, a_{l+q-1}^{\pi^*_B})$ where $b_l = \alpha_{l-1}$. Since $\pi_{BRH}$ is a feasible solution it follows for the optimal solution $\pi^*_{BRH}$ that

$$\tilde{J}_{\pi^*_G}(A_l) \leq \tilde{J}_{\pi^*_B}(A_l) \leq \tilde{J}_{\pi^*_{BRH}}(A_l).$$

While the batch greedy and batch greedy with revisionist history approaches may be able to identify better sparse designs, they also dramatically increase the number of designs that need to be evaluated at each step in the algorithm. In order to make these approaches tractable, we need an efficient method for network design evaluation. Fortunately, modern machine learning methods can help fill this gap. In particular, we develop a Convolutional Neural Network, which we show can accurately predict $\mu(A, S)$ for an unseen $A$. In our heuristics, we use $\mu(A, S)$ as an approximation of $J_{l+1}(A_{l+1})$.

### 3.3 Predicting the Expected Max Flow of a Flexibility Design

Consider an instance of the network design problem with $m$ plants, $n$ products, a demand distribution $D$, a capacity vector $c$, an initial design $A_0$, and $N$ the number of arcs in the final design. Note that for any flexibility design, the aforementioned values are constant. Therefore, the variation between different designs is entirely captured by the arc set, $A$. Moreover, for a fixed set of demand samples, $S$, we are interested in learning a function that maps network designs to expected max flow:

$$\mu : \{A \mid A \subseteq \mathcal{F}\} \rightarrow \mathbb{R}. \quad (3.10)$$

To train a prediction model that can accurately approximate this function, we follow a supervised learning approach by using a set of randomly generated network designs (inputs) and their corresponding expected max flow values (outputs). Notably, in this setting we are able to generate arbitrarily many input-output pairs only limited by computational resources and time considerations. Further, generating
input-output pairs is a highly parallelizable task that can be spread across multiple computational resources.

Recently, modern deep learning methods such as neural networks have shown impressive accuracy on a wide range of problems, especially when there is a large amount of data available for training [LeCun et al., 2015]. Once trained, neural networks are able to make a large number of predictions quickly especially when run using graphics processing units (GPUs). As a result, neural networks are well suited for the flexibility design problem studied in this thesis.

To train a neural network that predicts the expected max flow of a flexible design we need an efficient representation of the flexibility design. Because of the bipartite nature of the graph, the arc set $\mathcal{A}$ can be represented by an $m \times n$ incidence matrix, denoted by $X$ where the element $x_{ij}$ is 1 if $(i,j) \in \mathcal{A}$ and 0 otherwise. The incidence matrix representation of a flexibility design allows us to treat it as an “image” of $m$ by $n$ pixels. Figure 3.3 illustrates the translation from a network design with 10 plants and 10 products to the corresponding incidence matrix.

![Figure 3.3: Transformation of a Flexibility Design to an Incidence Matrix](image)

For prediction problems that have image-like inputs, a special type of neural network architecture called convolutional neural networks (CNN), has been shown to be particularly effective. In this instance the image-like input offers a convenient format for taking advantage of tools that are readily available to implement CNN models. In this thesis, we use a relatively simple CNN architecture with two hidden layers. The first layer is a convolutional layer with 1024 filters. We then flatten the convolutional layer and add a fully connected layer of 128 hidden units. The final layer is a single unit that outputs a real number estimating $\mu(\mathcal{A},S)$. Figure 3.4 illustrates the model architecture.

During development of our model, we found we were able to improve the prediction accuracy by making a number of adjustments to better adapt the architecture to our specific problem. Generally,
convolutional filters try and exploit feature locality, i.e., only neighbouring pixels have meaningful predictive patterns. However, we found this to not always be the case in our problem. For example, in Figure 3.3(b) the arcs represented by pixels $x_{18}$ and $x_{88}$ are relatively far apart, but could greatly influence another since both arcs are adjacent to demand node 8. As a result, we found the prediction accuracy of our model was much better if we used filters that were the same size as the input image. We note here that this choice makes the convolutional layer of our network equivalent to a standard fully connected layer. Despite the equivalence between the convolutional layer and the simpler fully connected layer, we find it more convenient to maintain the convolutional architecture representation as it is a more natural fit with the image-like input. Further, we also found that initializing the filters to be orthogonal matrices with no bias terms improved our model accuracy. While we did not conduct an exhaustive search of other possible weight initializations, this suggests that there may be some spatial structure the CNN is able to exploit.

To train the CNN we generate a data set consisting of flexibility designs and their corresponding expected max flows. Since our end goal is to identify a sparse flexible design with $N$ arcs, we only require the CNN to have accurate performance predictions for networks with at most $N$ arcs that contain the initial design $A_0$ as a subgraph. Based on experimental results, we found that prediction models trained on data sets containing network designs with more than $N$ arcs performed better when used by the heuristics we present in the next section. As a rule of thumb we include networks in the generated data set that contain up to $1.2N$ arcs. To randomly generate a design $A_r$ for $r = 1, \ldots, R$, we start with the initial configuration $A_0$. For each arc not in $A_0$, we add it to the design with probability $p_{\text{arc}}$, where $p_{\text{arc}}$ is drawn from a uniform distribution on the interval $[0, \frac{1.2N-|A_0|}{mn}]$. We repeat this procedure to generate $R$ random flexibility designs, $A_r$, and create a training set, $T$, by computing each $\mu(A, S)$ with a fixed sample $S$:

$$T = \left\{ \left( A_r, \frac{\sum_{d^k \in S} \mu(A_r, d^k)}{K} \right) \mid r = 1, \ldots, R \right\}. \quad (3.11)$$
During the development of our model we withheld validation and testing sets from the training set, \( T \). The validation set was used to tune the model parameters, such as number of filters and number of hidden units. Once we determined an appropriate set of parameters we evaluate the model performance on the test set.

To train the CNN we use the Keras Python package with a Tensorflow back end running the adam optimizer \cite{Kingma2014}. Throughout this work all prediction models were trained using an NVIDIA GeForce GTX 1080 Ti GPU. We also leverage the GPU’s ability to run in parallel by making predictions in batches. In our application, large prediction errors in \( \mu(A, S) \) can have particularly adverse effects on our heuristic’s performance, so to help limit the size of the largest prediction errors, we use a root mean square error loss function.

To demonstrate the accuracy of the CNN predictions we consider a balanced, symmetric network with 10 products and 10 plants. We initialize \( A_0 \) to be the dedicated network with connections between product \( i \) and plant \( i \) only, \( i = 1, \ldots, n \). We assume each of 5000 demand realizations are drawn from a normal distribution with a mean of 100 and standard deviation of 40, truncated at two standard deviations. Each plant has a production capacity of 100. This is the same setup as the first example from \cite{Jordan1995}. We refer to this particular instance of the flexibility design problem as the Jordan and Graves (JG) problem. We generate a set of 20,000 flexibility designs that range from 11 to 25 arcs. We train the CNN on 90\% of the data over 250 epochs, with 10\% left out for testing. For comparison, we also show the error for a naive predictor equal to the average value in the training set and a linear regression (LR) model that predicts the expected max flow purely based on the number of arcs in the design.

The results shown in Figure 3.5 show the significant predictive power of our CNN model compared to the naive predictors. We see that the magnitude of the CNN’s largest prediction errors is relatively small compared to the naive predictors, this characteristic is especially important in our later applications of the CNN prediction model. We also note that the variation of expected max flow values increases with the number of arcs. Intuitively, this makes sense as with more arcs added there is a larger discrepancy between good and bad network designs.

Note that the size of the training set is actually very small compared to the approximately \( 10^{15} \) flexibility designs that contain the dedicated network and have between 11 and 25 arcs (not accounting for symmetry). Nevertheless, the CNN is still able to make accurate predictions. The prediction of \( \mu(A, S) \) is significantly accelerated by taking advantage of advances in GPU technology. In our experiments we found that the CNN approach produces a speed up between 10,000 and 150,000 times compared to evaluating a design using a simulation and optimization procedure.
Finally, in the development of a DP framework in Section 3.2 we mentioned that our method, in particular the prediction model, may be able to be adapted to better fit with a traditional cost-to-go definition. To make this change the training set $T$ would need to be adjusted so that it consisted of input-output pairs, where the output was a true estimate of the cost-to-go function for some predetermined network size $N$. However, this consideration is beyond the scope of this thesis and left for future work. In the following section we use the CNN predictions in lieu of the traditional cost-to-go approximation and develop two heuristics for creating flexible designs.

### 3.4 ML-based heuristics

In this section we bring together the batch greedy and batch greedy with revisionist history frameworks developed in Section 3.2 with the CNN prediction model developed in Section 3.3 to derive two ML-based heuristics. In particular, the CNN’s accelerated evaluation of $\mu(A, S)$ enables us to consider much larger neighbourhoods at each stage in a constructive approach to designing flexible networks.

First, we adapt the batch greedy framework. We let $A_l$ denote the current network and add batches of $q$ arcs, by considering all possible combinations of arcs that could be added to $A_l$, which translates to a set of $\binom{C_{A_l}}{q}$ networks to evaluate. We then predict the expected max flow of candidate networks in batches using the CNN to take advantage of the GPU’s efficient parallel computation properties. To help mitigate the effect of prediction errors we use the prediction results to produce a short list of the top $r$ candidates, which we then evaluate using a simulation and optimization procedure with a fixed sample $S$. We refer to $r$ as the search size. We can repeat this procedure iteratively until the network contains a predetermined number of arcs $N$. We refer to this method as the Predict and Search (PS) heuristic and provide the details below as Algorithm 1.
**Algorithm 1** Predict and Search

1: predict: evaluate design by CNN
2: solve: evaluate design by simulation and optimization
3: \( A \leftarrow A_0 \)
4: while \(|A| < N\) do
5: \( X = \{\} \)
6: for \( \Omega \in \binom{A}{q} \) do
7: \( A_\Omega = A \cup \Omega \)
8: \( X . \text{append}(A_\Omega) \)
9: \( \text{predictions} = \text{predict}(X) \)
10: \( \text{candidates} = \arg \max \{\text{predictions}, r\} \)
11: \( A \leftarrow \arg \max \{\text{solve}(x) \text{ for } x \text{ in candidates}\} \)

To demonstrate the functionality of the PS heuristic we use the prediction model from Section 3.3 to iteratively add 10 arcs to the dedicated network from the JG problem using a batch size of \( q = 2 \). Figure 3.6 illustrates the network at each intermediate step, with the added links highlighted in green.

![Figure 3.6: Steps of the Predict and Search Algorithm](image)

It is clear that the PS algorithm constructs a sub-optimal flexibility design, since we know from Simchi-Levi and Wei [2012] that the optimal design in this setting is the long chain. Examining the intermediate steps we see that at step 2, the PS algorithm adds arcs to form the following closed chain:


This is problematic since a closed chain at an intermediate step makes it impossible to achieve the
long chain in the future. The flexible algorithmic design allows us to extend the batch size to help mitigate short-sighted errors, but unless we are able to choose \( q = N \) the possibility of short-sighted errors still exist.

The revisionist history framework presented in Section 3.2 offers a method to help mitigate short-sighted errors. In the context of flexibility design problems this equates to removing a previously added arc at a future iteration. Algorithmically we implement this concept at each iteration by looping over the arcs that have previously been added, temporarily removing them one at a time. For each removed arc, we run the PS procedure. Then we choose the best arc to remove and the best arcs to add. We refer to this heuristic as Predict and Search with Revisionist History (PSRH) given as Algorithm 2. We provide the details below:

**Algorithm 2** Predict and Search with Revisionist History

1: Run Predict and Search Algorithm to add a single batch of links denoted \( b_0 \)
2: \( A \leftarrow A_0 \cup \{b_0\} \)
3: \( \text{added_arcs} = [b_0] \)
4: \( X = [\ ] \)
5: while \(|A| < N\) do
6: \( \text{for arc in added_arcs do} \)
7: \( X = [\ ] \)
8: \( \text{for } \Omega \in \binom{C_A}{q} \text{ do} \)
9: \( A_\Omega = (A \backslash \{\text{arc}\}) \cup \Omega \)
10: \( X.\text{append}(A_\Omega) \)
11: \( \text{predictions} = \text{predict}(X) \)
12: \( \text{candidates} += \text{arg max}(\text{predictions}) \)
13: \( A = \text{arg max}[ \text{solve}(x) \text{ for } x \text{ in candidates}] \)
14: \( \text{added_arcs} = A \backslash A_0 \)

To demonstrate the PSRH heuristic we run the algorithm (with \( q = r = 2 \)) to add ten arcs to the dedicated network from the JG problem. The design pathway is illustrated in Figure 3.7.

Similar to the PS algorithm, the PSRH algorithm forms closed chains in intermediate steps. For example at Step 2 there is a closed chain that follows the path:

plant 2 → product 3 → plant 9 → product 9 → plant 10 → product 10 → plant 2.

However, the ability to remove previously added links allows the heuristic to correct this short-sighted error by removing the arc (2, 10) to open the chain and then adding (2, 8) and (8, 10) to form a longer chain:

plant 2 → product 2 → plant 9 → product 9 → plant 10 → product 10 → plant 8 → product 8 → plant 2.
Figure 3.7: Steps of the Predict and Search with Revisionist History Algorithm

Ultimately this results in the final flexibility design achieving the optimal long chain structure. Overall these methods have shown promising results, and in the following section we further demonstrate their performance by comparing them to existing methods in the literature.
Chapter 4

Evaluation of Existing Methods

In this chapter, we compare the PS and PSRH heuristics to state-of-the-art heuristics presented in the literature. With the exception of Yan et al. [2017], the heuristics we use for comparison all follow the same general framework starting with an initial design, $A_0$, and iteratively adding arcs. The heuristic from Yan et al. [2017] begins with the fully flexible network and deletes arcs until the design contains a specified number of arcs. However, the authors also present a version of their heuristic that can be used to build up a sparse structure by greedily adding the most effective arc, which is the version that we implement in this comparison.

The heuristics that we consider from the literature are listed below, along with a brief explanation of the method. We refer the reader to the source papers for additional details.

- **Expander [Chou et al., 2011]**: use the concept of graph expanders to connect a small group of products to a large group of plants based on their respective demand and capacity with the goal of building a highly connected graph. The results presented in the paper only considers a single node at a time due to the computational challenges of considering larger subsets. Arcs are added to an existing design by computing an expansion ratio for each node. For demand nodes (products) the expansion ratio is computed by dividing the sum of the plant capacities the product is connected to by its own expected demand. For supply nodes (plants), the expansion ratio is computed by dividing the sum the product expected demands that the plant is connected by its own capacity. Thus, small ratios indicate nodes are under-utilized and an arc should be added to connect the most unmet demand to the most unused capacity.

- **UW/W-PCI [Simchi-Levi and Wei, 2015]**: approximate the tightness of bottlenecks containing a fixed number of products and add arcs to relieve the tightest bottlenecks. The tightness
Chapter 4. Evaluation of Existing Methods

of bottlenecks is determined by computing a series of plant cover indices (PCI), which are the minimum plant capacities required to create a vertex cover, given the vertex cover contains exactly \( k \) products. Then summing the PCI indices, the heuristic gains the advantage of considering all the bottlenecks. The summation is a weighted sum of the PCI values. In their work the authors present two versions of the PCI based heuristic based on different weighting functions used to sum the PCI indices, which they refer to as the W-PCI and UW-PCI heuristics.

- **MDEP [Feng et al., 2017]**: extend the results from [Mak and Shen, 2009]. The authors develop a two stage heuristic for constructively adding arcs based on the stochastic IP formulation. During the first stage, they solve a LP relaxation of the assignment problem allowing additional arcs to be added for a fixed cost. The relaxed optimization problem gives a fractional value between 0 and 1 for each candidate arc. The next arc to be added is the one that corresponds to the largest fractional value.

- **DVBH [Yan et al., 2017]**: begin with a fully flexible arc and iteratively delete arcs until the network design is sufficiently sparse. To determine the next arc to delete the authors add a quadratic constraint relating to sensitivity analysis of the max-flow problem. To solve the constrained problem they reformulate the problem into a copositive problem. The next arc to delete is determined by considering the shadow price and removing the arc that worsens the objective value the least. The authors provide no guidelines for adapting this method such that the design will contain a predetermined initial design. However, they do provide a constructive version of their heuristic that selects the next arc by choosing the arc that improves the objective value the most. At each stage this requires checking all possible arcs that can be added by solving the reformulated problem for each arc. The authors warn that the constructive method requires much greater computation than the arc deletion method; however, to easily incorporate the initial designs we must follow the constructive approach.

We note that three of these four heuristics were developed with worst-case performance as the goal. However, there is growing evidence in the literature that suggests a strong correlation between heuristic performance in the worst-case and in expected value [Simchi-Levi and Wei, 2015, Yan et al., 2017]. Further, our approach could be easily adapted to optimize for the worst-case by adjusting the measure \( \mu(A, S) \) in the training data generation step.

For purpose of comparison, we evaluate the expected max flow of flexibility designs generated by the different heuristics over three test instances of the flexibility design problem described in [Simchi-Levi and Wei, 2015]. The test instances increase in generality from balanced and symmetric to unbalanced...
Chapter 4. Evaluation of Existing Methods

Figure 4.1: Initial Flexibility Designs

and symmetric to unbalanced and asymmetric. For the first test setting, we consider a network with 10 products and 10 plants where demand for each product is assumed to be i.i.d and drawn from a normal distribution with mean 1 and a standard deviation of 0.5. We treat any negative samples of demand to be 0. The capacity of each plant is 1, so the total capacity and expected demand in the system are equal. The second setting is a network with 14 products and 7 plants. The demand distribution for each product is the same as the previous case and capacities are chosen to be $c_1 = c_2 = 3$, $c_3 = c_4 = c_5 = 2$, $c_6 = c_7 = 1$. Finally, in the third test setting we consider an unbalanced system with asymmetric demand distributions. The mean demand for each product $i$, $\mu_i$ is first drawn from a uniform distribution between 0.5 and 1.5. Then for each product we draw samples from a normal distribution with mean $\mu_i$ and standard deviation 0.5 truncated at 0. Capacities of each plant are chosen such that for each plant the total capacity is equal to the sum of the adjacent demands:

$$
c_1 = \sum_{i=1}^{3} \mu_i, \quad c_2 = \sum_{i=4}^{6} \mu_i, \quad c_3 = \sum_{i=7}^{8} \mu_i, \quad c_4 = \sum_{i=9}^{10} \mu_i, \quad c_5 = \sum_{i=11}^{12} \mu_i, \quad c_6 = \mu_{13}, \text{ and } c_7 = \mu_{14}.
$$

The initial network for each of the test settings is shown in Figure 4.1 and in each case we iteratively add 10 links. For each of the test settings we generate 50,000 random networks following the procedure in Section 3.3. We withhold 5,000 samples for testing and train a CNN on 40,500 samples using the remaining 4,500 for validation. Figure 4.2 illustrates how the expected max flow increases from the initial design for each heuristic in both absolute terms and relative to the expected max flow of the fully flexible network.
Chapter 4. Evaluation of Existing Methods

In each test setting, the design produced by the PSRH heuristic dominates the designs produced by all other methods from the initial design until all 10 links are added. Notably, in test setting 1 the PSRH
heuristic is able to recreate the long chain. Furthermore, there is variation in the relative performance of the existing methods over the different test settings. For example the MDEP method performs well in test setting 1 but produces comparably inefficient designs in test settings 2 and 3. Table 4.1 summarizes the expected max flow of the terminal designs produced by each of the heuristics. We also include the expected max flow over the initial network design (Initial), the fully connected network (Full Flexibility), and the maximum and average expected max flow from all the designs in the training set with the same number of arcs as the terminal designs denoted by Best Random and Average Random, respectively, as baselines.

Table 4.1: Flexibility Design Heuristics Expected Max Flow

<table>
<thead>
<tr>
<th>Heuristic</th>
<th>Test Setting 1</th>
<th>Test Setting 2</th>
<th>Test Setting 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline Networks</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Initial</td>
<td>8.04</td>
<td>12.12</td>
<td>13.19</td>
</tr>
<tr>
<td>Average Random</td>
<td>8.68</td>
<td>12.86</td>
<td>13.91</td>
</tr>
<tr>
<td>Best Random</td>
<td>9.04</td>
<td>13.21</td>
<td>14.26</td>
</tr>
<tr>
<td>Full Flexibility</td>
<td>9.40</td>
<td>13.30</td>
<td>14.34</td>
</tr>
<tr>
<td>Expander</td>
<td>8.62</td>
<td>13.08</td>
<td>14.30</td>
</tr>
<tr>
<td>W-PCI</td>
<td>9.06</td>
<td>13.25</td>
<td>14.27</td>
</tr>
<tr>
<td>UW-PCI</td>
<td>9.00</td>
<td>13.23</td>
<td>14.27</td>
</tr>
<tr>
<td>MDEP</td>
<td>9.23</td>
<td>12.91</td>
<td>14.13</td>
</tr>
<tr>
<td>Dual</td>
<td>8.74</td>
<td>13.09</td>
<td>14.16</td>
</tr>
<tr>
<td>ML-based Heuristics</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PS</td>
<td>9.03</td>
<td>13.21</td>
<td>14.32</td>
</tr>
<tr>
<td>PSRH</td>
<td>9.24</td>
<td>13.25</td>
<td>14.32</td>
</tr>
</tbody>
</table>

To test if there is a significant difference in the expected max flow of networks produced by the various heuristics we conduct the same pairwise statistical significance test used by Simchi-Levi and Wei [2015]. For each pair of networks, $A_1$ and $A_2$, we compute the mean of the differences, $M$, and the standard error, $SE$, over a sample of 20,000 demand vectors where

$$M = \frac{\sum_{k=1}^{20000}(\mu(A_1, d^k) - \mu(A_1, d^k))}{20000}, \text{ and}$$

$$SE = \sqrt{\frac{\sum_{k=1}^{20000}(\mu(A_1, d^k) - \mu(A_1, d^k))^2}{(20000)(19999)}}.$$

Figure 4.3 summarizes the result of the statistical significance test for each of the test settings. A check mark indicates that the design produced by the corresponding row heuristic has a statistically
significant higher expected max flow than the design from the corresponding column heuristic. Notably, for each of the test settings, the PSRH heuristic outperforms the other heuristics with the exception of MDEP in test setting 1 and PS in test setting 3.

Finally, we evaluate the worst-case performance of the networks produced by the heuristics. We plot the percentiles of the expected max flow over the samples of demand vectors. Figure 4.4 shows that, similar to average case performance, the PS and PSRH heuristics create network designs that perform well across all percentiles.

Overall, we find that the PSRH heuristic is able to produce efficient designs for problems ranging from balanced and symmetric to unbalanced and asymmetric. The ML-based heuristics we developed achieve similar or better performance compared to state-of-the-art methods from the literature that were derived from deep mathematical insights. Moreover, the power of the ML-based heuristics was limited by the choice of conservative values for batch and search size. However, as illustrated in Figure 4.5, the performance gains from increasing the heuristic parameters values is limited. This is especially true in the case of the PSRH heuristic.

Here we have demonstrated that the PSRH heuristic is an effective method for producing efficient network designs in each of the test settings. However, when considering the overall performance of the PS, PSRH, and existing heuristics we must take into account the time required to run the heuristics. For both PS and PSRH there is significant overhead required before the heuristics can be run. In particular, we must first generate a training set and then train the CNN at the core of the heuristics. For test settings 1, 2, and 3 generating the training set took between 3 and 4 hours when run in parallel on 2 Intel Xeon e5-2630 CPUs. In addition training the CNN for each test setting takes approximately 10 minutes. However, once trained the CNN was able to make predictions for 1000 network designs in 50
Figure 4.4: Distribution of Expected Max Flow for Final Network Designs
Chapter 4. Evaluation of Existing Methods

Figure 4.5: Comparison of ML-based Heuristic Parameters
milliseconds compared to the 4 minutes required to evaluate the 1000 networks using the simulation and optimization approach. This represents almost a 50,000 times speed up in network design evaluation. In test setting 1 this upfront cost allows the PSRH heuristic to evaluate over 150,000 network designs during a single run of the algorithm. In comparison, the existing heuristics run relatively quickly; the Expander, W-PCI, and UW-PCI heuristics run in less than 1 second. The MDEP and DVBH heuristics run in under 2 minutes and 2 hours respectively. We note that while the ML-based heuristics proposed are more computationally expensive than the existing heuristics they are still tractable in the problem instances we consider in this thesis.

In the following chapter we demonstrate the generalizability and effectiveness of our heuristic in a highly unbalanced and asymmetric healthcare production network, incorporating additional clinical constraints that capture unique aspects of the problem. In our analysis, we run the ML-based heuristics with batch size and search size both equal to 2.
Chapter 5

Radiation Therapy Case Study

In this section, we introduce a novel application of process flexibility in healthcare operations. In particular, we study the assignments of different types of patients requiring radiation therapy to available treatment machines (linear accelerators, or LINACs) with the aim of minimizing overtime in the operations of the cancer center. This case study is based on the operations of Princess Margaret Cancer Centre (PM) in Toronto, Canada. First, we adapt the general process flexibility framework to model problem-specific constraints arising in the operation of PM’s radiation therapy unit. We then demonstrate the application of our network design heuristics in three computational studies: 1) redesigning the patient assignment policies to reduce the number of connections in the network, 2) determining the value of operating a fleet of homogeneous LINACs, and 3) assessing the impact of spreading patient treatments evenly across all available LINACs. Finally, we perform a sensitivity analysis of our results with respect to clinic-specific model parameters.

5.1 Problem Background

The Radiation Medicine Program (RMP) at PM is the largest single-site radiation treatment facility in North America and one of the largest in the world, treating approximately 400 patients per day. Our analysis builds on a conceptual correspondence between a radiation therapy treatment center and the general manufacturing networks studied in early chapters.

We model the RMP’s operation as a treatment network comprising patients (products) requiring RT treatment and LINACs (plants) capable of delivering them. We assume for simplicity that demand is constant. Although there is some daily variation in demand, several factors suggest that this assumption is reasonable: (1) the typical treatment course for a patient lasts several weeks, (2) the number of new
Table 5.1: Conceptual correspondence between manufacturing and a radiation therapy clinic.

<table>
<thead>
<tr>
<th>Manufacturing</th>
<th>Radiation Therapy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plants</td>
<td>Linear Accelerators</td>
</tr>
<tr>
<td>Products</td>
<td>Patient Types</td>
</tr>
<tr>
<td>Capacity</td>
<td>Time available for treatment each day over all LINACs</td>
</tr>
<tr>
<td>Demand for each product</td>
<td>Time required to complete all treatments each day</td>
</tr>
<tr>
<td>Capacity uncertainty</td>
<td>Unexpected LINAC breakdowns</td>
</tr>
</tbody>
</table>

patients starting treatment each day is small, (3) treatments are scheduled well in advance, and (4) it is uncommon for patients to miss treatment. On the other hand, there is non-trivial uncertainty in daily treatment capacity due to the possibility of unexpected machine breakdowns. Thus, in this application, we reverse the typical setup where capacity is considered fixed and demand is uncertain. However, we can still apply the network design approaches in this setting because of the symmetry between capacity and demand, as evident in model (3.1). In Table ?? we summarize the correspondence between the general manufacturing and radiation therapy settings.

5.1.1 Data

In this section we detail the approach taken to model PM’s radiation therapy unit as a treatment network. First, we consider the supply and demand nodes. Second, we describe the methodology used to determine the distribution of the capacity vector and the fixed demand vector.

PM has 15 LINACs that are used for delivering radiation therapy treatments. Each LINACs is one of three different models: Elekta Infinity, Varian Truebeam, Varian iX. Supply nodes in the treatment network each represent a single LINAC, which we refer to each supply node as a treatment unit.

On the demand side of the network are the patients requiring treatment. These patients are designated as receiving either curative or palliative courses of treatment. At PM curative patients vary by the technique used to deliver their treatment. In our analysis we consider 168 distinct treatment techniques used at PM, which provides a low-level distinction between different patients. For example, some techniques vary only by the dose of radiation delivered to the patient. Other techniques vary by how the radiation is delivered to the patient (a continuous arc around the patient vs. from a discrete number of angles). In this work, we group together treatment techniques that share common technical requirements for delivery. To approximate the overlap of requirements across the various treatment techniques within each disease site, we assume that treatment techniques assigned to the same subset of LINACs in PM’s current configuration share common requirements for delivery. This assumption results in 35 curative patient groupings, or patient types. A detailed summary of the treatment techniques within each of the
35 patient types is given in Table A.1 in Appendix A. We assume that all palliative patients are the same, making no distinction between different types of treatment required. We make this simplifying assumption for two reasons. The first is that guidelines dictate palliative patients receive treatment within 7 days of the intent to treat decision, compared to 14 days for curative patients. This time constraint puts pressure on patient schedulers to assign palliative patients to the first available LINAC. Second, the treatment of palliative patients tends to be more complex, especially when a patient’s disease has metastasized, making it difficult to delineate between different patient types. We assume that palliative patients are capable of being treated on any of the LINACs. Historical data confirms that treatments for palliative patients were spread across all LINACs. Together curative and palliative patients form 36 distinct demand nodes. Figure 5.1 illustrates the LINACs and patient groupings that form the nodes of the treatment network along with the 136 arcs between them in the existing PM configuration.

To estimate the distribution of capacity vectors, \( \text{C} \), we assume each LINAC’s regular operating hours are between 8:00AM to 6:30PM (630 minutes). LINACs may experience unexpected service disruptions caused by a range of malfunctions from mechanical failures to software issues, which result in a reduction of available operating hours of the machine. When the therapists operating a treatment unit determine the LINAC has a problem that prevents it from delivering further treatments, they record the time of the breakdown using PM’s AQUA system. Breakdowns can be detected by the morning quality assurance test or during regular operation. The therapists also record the time the LINAC comes back online and is available to resume treatment delivery. The amount of machine downtime can vary from a few minutes to an entire day.

We obtained breakdown data from the AQUA system that included all instances of a treatment unit being listed “out of service” due to “Emergency Servicing” between July 2015 and July 2017, covering 538 operating days. Across all treatment units, we observed that on a given day an individual treatment unit has a 9.03% chance of any type of breakdown. The distribution of breakdown lengths in the historical data is shown in Figure 5.2. Note that the distribution is “U-shaped”, with a 23.5% likelihood of a breakdown lasting at most 15 minutes and a 9.9% likelihood of an all-day breakdown. We assume each LINAC is identical and independent with respect to the likelihood and duration of breakdowns.

To generate a single instance of the capacity vector, \( \text{c} \), we first simulate a yes/no breakdown outcome (with probability of breakdown 0.0903) and conditional on a breakdown, we draw from the historical breakdown duration distribution.

As mentioned above, we assume that demand for each of the patient groupings is fixed. Over the last five years, about 31% of total treatment time has been spent on palliative patients. To determine the demand for each of the 35 curative patient groupings, we obtained a summary of 3,437 courses of
curative treatments delivered between April 1, 2015 and March 31, 2016. For each patient grouping we first determine the demand for each treatment technique by multiplying the number of times it was used by the total number of minutes required to deliver the full course of treatment (see Table A.1). We then sum the demand of all treatment techniques associated with that patient group.

Figure 5.1: Existing PM Treatment Network

In summary, we have modeled PM radiation therapy clinic in terms of the process flexibility problem by determining the set of supply nodes (LINACs), demand nodes (patient groupings), and arc set (LINAC to patient group assignments). Figure A.1 in Appendix A summarizes what data sources were used to construct the model of the existing PM treatment network.
In this section, we adapt the general process flexibility framework to reflect the clinical operation at PM. Let $I$ be the set of all LINACs and $J$ be the set of all patient types. Parameters $\bar{c}_i$ and $d_j$ denote the maximum capacity LINAC $i$ and demand generated by patient type $j$ (both in minutes of treatment time), respectively. Let $f_{ij}$ denote the amount of demand of patient type $j$ treated on LINAC $i$ in minutes.

Next, we describe three specific clinical considerations that we incorporate into our model and data generation process. The first issue pertains to the capacity and demand data. In the clinic, all patients scheduled to receive RT on a given day must be treated that day. So while unmet demand would result in lost sales in the manufacturing setting, here, LINACs would run overtime if needed to ensure all demand is met. We denote the amount of overtime of LINAC $i$ by the decision variable $o_i$. Knowing that all patients scheduled must be treated and that there is a chance of machine breakdown, the clinic purposely underbooks. That is, they schedule so that total treatment time required on any given day is strictly less than total available capacity. We let $\alpha \in [0, 1)$ represent the excess capacity as a proportion of the total available capacity. So $\alpha = 0.1$ means that total demand scheduled is 90% of the available capacity. Later, we will examine the effect of varying $\alpha$ by scaling the total demand according to the equation:
\[ \sum_{j \in J} d_j = (1 - \alpha) \sum_{i \in I} \bar{c}_i, \]

The other two considerations, both related to machine breakdown, affect the model. An initial schedule (assignment of patients to LINACs) is always made well in advance of the treatment day. However, machine breakdowns on the treatment day effectively lower the available capacity, possibly leading to re-assignment of some patients to different LINACs. While it may be possible in the manufacturing context for as much production as needed to be moved to other factories, in our application, we assume there is a limit to how many patients can be “re-shuffled”. This limitation is motivated by operational realities and preferences. First, patients typically see the same therapist on every visit and therapists tend to work on the same treatment machines. Handing off a patient to other therapists or moving to other machines is generally not preferred by the patients or therapists, respectively, unless the downtime is expected to be significant. For example, handing off a patient to another treatment unit requires significant effort to coordinate between several care teams. We incorporate this idea by adding a constraint on the flow that limits how different the post-breakdown flow vector can be from a given initial flow vector (initial schedule), using a parameter \( \Gamma \).

The other issue is that, in practice, not all treatment machines are identical. In particular, PM treatment machines vary across the three different types of LINACs they operate. Different types of LINACs use different software to generate treatment plans. Currently, a treatment plan created for one type of LINAC cannot be easily ported to another. Thus, following a machine breakdown, a patient must move to a LINAC of the same type as the original one. This constraint is also included in the model by partitioning the set of LINACs \( I \) into subsets that group together LINACs of the same type.

Given the clinical considerations described above and an initial schedule denoted by \( \bar{f} \), we use the following model to optimize patient-LINAC assignments to minimize overtime after a breakdown has occurred.
minimize \( \sum_{i \in I} o_i \)

subject to \( \sum_{j : (i,j) \in A} f_{ij} \leq c_i + o_i, \quad \forall i \in I, \)
\( \sum_{i : (i,j) \in A} f_{ij} = d_j, \quad \forall j \in J, \)
\( \frac{\sum_{(i,j) \in A} |f_{ij} - \bar{f}_{ij}|}{2\sum_{(i,j) \in A} \bar{f}_{ij}} \leq \Gamma, \)
\( \sum_{i \in I_s : (i,j) \in A} f_{ij} = \sum_{i \in I_s : (i,j) \in A} \bar{f}_{ij}, \quad \forall j \in J, s = 1, 2, 3, \)
\( f_{ij} \geq 0, \quad \forall (i, j) \in A. \)

The capacity vector \( c \) is generated using the simulation procedure outlined in Section 5.1.1. The third constraint limits re-shuffling following a breakdown. In particular, it requires that when creating the post-breakdown schedule the total proportion of treatment time that can be moved to a different LINAC is no more than \( \Gamma \). Effectively, \( \Gamma = 0.1 \) allows up to 10% of treatment time to be scheduled on a LINAC different from the one it was initially scheduled on. We note that to implement this constraint we first linearized the absolute value by incorporating a set of dummy variables for each \((i, j) \in A\). The fourth constraint deals with the LINAC heterogeneity issue by requiring, for each disease site, that the total treatment time delivered on each of the three LINAC types be equal before and after the breakdown.

Given an optimal solution \((F^*, o^*)\) to problem (5.1), we compute
\[
\nu(A, c) = \sum_{(i,j) \in A} f_{ij}^* - \sum_{i \in U} o_i^*,
\]
which is the total treatment time delivered during regular operating hours. Similar to the general case, we are concerned with maximizing the expected max flow over the distribution of capacities \( C \) subject to some design constraints \( A \):
\[
\max_{A \in \mathcal{A}} \mathbb{E}_{c \in C} [\nu(A, c)].
\]

We approximate the value of \( \mathbb{E}_{c \in C} [\nu(A, c)] \) using the sampling approach described in Section 3.1. Namely, we draw \( K \) capacity vectors from the distribution \( C \) and solve (5.1) for each realization. We employ a slight abuse of notation and define the average \( \nu(A, S) \) over the set \( S \) of \( K \) samples to be
\[
\nu(A, S) = \frac{1}{K} \sum_{c \in S} \nu(A, c).
\]
An initial schedule, \( \bar{f} \), can be generated by solving the following optimization model:

\[
\begin{align*}
\text{minimize} & \quad \sum_{i \in I} o_i + \sum_{(i,j) \in A} \xi_{ij} f_{ij} \\
\text{subject to} & \quad \sum_{j : (i,j) \in A} f_{ij} \leq \bar{c}_i + o_i, \quad \forall i \in U, \\
& \quad \sum_{i : (i,j) \in A} f_{ij} = d_j, \quad \forall j \in V, \\
& \quad f_{ij} \geq 0, \quad \forall (i,j) \in A.
\end{align*}
\] (5.4)

The auxiliary objective function in model (5.4) is used to randomize the output flow vector among the multiple optimal solutions that all minimize total overtime. Each parameter \( \xi_{ij} \) is drawn randomly and independently from a uniform distribution on \([0.01,0.5]\). This randomization is important since the value of \( \nu(A,c) \) is dependent on \( \bar{f} \). To generate the training set of network designs and corresponding expected max flow values, we generate 10 random initial schedules using (5.4). For each network design, we partition \( S \) into 10 equally sized subsets and compute \( \nu(A,c) \) using the corresponding initial schedule. The full details are given as Algorithm 3.

**Algorithm 3 Simulation and Optimization for Evaluating Treatment Networks**

1: \( S \): Sample of capacity vectors \( c \)
2: \( S = S_0 \cup S_1 \cup \ldots \cup S_9 \) such that \( |S_i| = |S| / 10 \) \( i = 0, \ldots, 9 \)
3: for \( i \) in range(10) do
4: solve (5.4) with random \( \xi_{ij} \)'s to generate \( \bar{f} \)
5: \( \nu_i(A,S_i) = \frac{1}{|S_i|} \sum_{c \in S_i} \nu(A,c) \) by solving (5.1)
6: \( \nu(A,S) = \frac{1}{10} \sum_{i=0}^9 \nu_i(A,S_i) \)

### 5.3 Redesigning the RMP treatment network

In this section, we use the heuristics that we previously developed to investigate the impact of redesigning the RMP treatment network. For all the numerical experiments, unless stated otherwise, we consider the regular capacity of all 15 LINACs to be 630 minutes, totalling 9450 minutes of daily capacity. There are 36 patient types, which are described in Section 5.1.1. Our base case assumes that total demand is equal to 97.5% of the total capacity, or 9213.75 minutes, corresponding to \( \alpha = 0.025 \). After a breakdown, we allow up to 5% of treatment time to be reassigned to a new LINAC, i.e., \( \Gamma = 0.05 \). To evaluate network performance, we used a fixed sample, \( S \), of 7500 breakdown scenarios generated following the procedure in Section 5.1.1. All heuristics use an initial network design where each patient type is assigned to exactly one LINAC for a total of 36 arcs. For patient types where the RMP scheduling guidelines
include a primary or preferred LINAC, we choose this LINAC. The remaining patient types are assigned
to a LINAC chosen from the subset of LINACs that they are assigned to in the existing RMP network,
in such a way as to balance the number of patient types initially assigned to each LINAC. The initial
PM treatment network (where each patient type is assigned to exactly one LINAC) is shown in Figure
B.1 in Appendix B.

To train the CNN, we follow the procedure described in Section 3.3. We generate 250,000 random
networks that have between 36 and 100 arcs, each containing the initial network design described above.
We calculate the expected max flow of each network following the procedure in Section 5.1.1 90% of the
data is used for model training and 10 % is withheld for testing. Figure 5.3 illustrates the distribution
of the training set, testing set, and prediction errors by the number of arcs.

![Figure 5.3: Heterogeneous RMP Simulated Network Data](image)

We applied our PS and PSRH heuristics to the initial network design using a batch size of 2 and a
search size of 2, informed by the results from Chapter 4. For comparison, we also apply the Expander
heuristic [Chou et al., 2011] and W-PCI and UW-PCI heuristics [Simchi-Levi and Wei, 2015]. For the
MDEP heuristic [Feng et al., 2017], we found that the largest sample of capacity vectors we were able
to run the heuristic with was too small to produce comparable results. We did not implement the
DVBH heuristic [Yan et al., 2017] because it requires the solution of a co-positive program, which for
our problem instance, was too large to solve.

Figure 5.4 shows the performance of the networks designed by the heuristics, for network sizes ranging
from 36 to 80 arcs. The vast majority of increase in expected max flow comes early as patient groupings
with relatively large demand nodes, e.g. Palliative Group 1, are assigned to additional LINACs. Early
on, between 36 and 42 arcs, the Expander, PS, and PSRH heuristics outperform the PCI-based heuristics,
since they are able to more effectively match large demands to available LINACs. Between 42 and 54
arcs, the Expander, PS, and PSRH heuristics produce designs that approach the expected max flow
of the fully flexible network before the performance begins to level off. However, both the Expander
and PSRH heuristics reach this point much earlier than the PS heuristic, which achieves only marginal increase in performance between 44 and 48 arcs. In this range, the PS heuristic continues to add arcs to the patient groups with the largest demands — adding 2 arcs to Palliative Group 1 and 1 arc to Head & Neck Group 1 — suggesting that the prediction model overestimates the benefit of adding arcs to the largest demand nodes. In contrast, the PSRH heuristic’s more conservative approach to adding arcs allows it to avoid this pitfall. Interestingly, without any specialized information about the clinical constraints, the Expander produces designs that perform similar to those produced by the PSRH heuristic during the entire design process. Simchi-Levi and Wei 2015 noted in their analysis that the Expander performed well when the network was unbalanced with asymmetric demands, which is the case for PM’s treatment network. The results here seem to provide further evidence that the Expander produces impressive results in these settings.

Figure 5.4: Heuristic Design Performance Progression

The designs produced by the PSRH heuristic with 46, 56, and 80 arcs are included in the Appendix B for illustrative purposes. We note that when the network is very sparse (fewer than 8 arcs added to the initial design), the expected max flow values that result from Algorithm 3 with the 10 different initial schedules were all the same. In other words, when the network is too sparse, the initial schedule has no impact on the performance of the network after breakdown. But, when a sufficient number of arcs are in the network, the initial schedule has a nontrivial impact on the performance of the network.

In Figure 5.5, we estimate the distribution of the expected max flow value for the PSRH designs over 100 runs of Algorithm 3. What we see is that performance is sensitive to the random initial schedule.
when the network design is sufficiently flexible. This suggests that with re-shuffling constraints, the initial schedule becomes an important consideration for the process flexibility problem in our clinical context. In the following sections, we try to reduce the sensitivity to the initial schedule by making PM-specific adjustments to the treatment network, which can later be extended to the general process flexibility setting.

![Variability with respect to The Initial Schedule](image)

(a) PSRH Design Performance Progression  (b) Tail of PSRH Design Performance Progression

Figure 5.5: Variability with respect to The Initial Schedule

Next, we look a bit deeper into the PSRH-designed networks. Specifically, we compare the networks with 46, 56, and 80 arcs produced by PSRH to the current PM treatment network and the fully flexible network. To account for the variability with respect to the initial schedule, we run Algorithm 3 100 times for each of the networks.

<table>
<thead>
<tr>
<th>Network</th>
<th>Expected Max Flow</th>
<th>Standard Deviation</th>
<th>Number of Arcs</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSRH</td>
<td>9057.93</td>
<td>9.07</td>
<td>46</td>
</tr>
<tr>
<td>PSRH</td>
<td>9098.74</td>
<td>18.46</td>
<td>56</td>
</tr>
<tr>
<td>PSRH</td>
<td>9101.80</td>
<td>17.89</td>
<td>80</td>
</tr>
<tr>
<td>Existing PM</td>
<td>9105.76</td>
<td>18.62</td>
<td>136</td>
</tr>
<tr>
<td>Fully Flexible</td>
<td>9110.98</td>
<td>17.10</td>
<td>540</td>
</tr>
</tbody>
</table>

Table 5.2: Comparison of Network Designs with Heterogenous LINACs

Notably, the PSRH heuristic produces a network design that achieves 99.97% of the expected max flow of the existing RMP treatment network using just 56 arcs (less than two arcs per patient type on average). Table 5.3 details the difference in arcs between the PSRH-designed 56 arc network and the 136 arc existing RMP network. Interestingly, two patient types, Breast Group 2 and Head & Neck Group 1, have more arcs in the redesigned network than they did in the existing RMP network. This suggests that these patient types are under connected in the existing PM network. Notably, patients within Breast Group 2 require that the assigned LINAC be equipped with an Active Breathing Control (ABC)
device suggesting that an investment in additional ABC devices could allow for increased specialization for other patient types. In contrast the number of LINACs assigned to treat Breast Group 3 and 4 is significantly lower in the redesigned network. Another important distinction between the two networks is that the number of patient types assigned to only one LINAC dramatically increases in the redesigned network from 3 to 29. This dramatic increase suggests that the majority of the benefits associated with flexibility can be had by focusing on only a few patient types. However, in practice, having patients assigned to only one LINAC may be problematic, especially on those days with long service disruptions.

Table 5.3: Difference in Number of Arcs Between 56 arc PSRH Network and Existing PM Network

<table>
<thead>
<tr>
<th>Patient Type</th>
<th>Num. Arcs</th>
<th>Patient Type</th>
<th>Num. Arcs</th>
<th>Patient Type</th>
<th>Num. Arcs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breast Group 1</td>
<td>-1</td>
<td>Eye Group 4</td>
<td>-3</td>
<td>Lymphoma Group 3</td>
<td>-1</td>
</tr>
<tr>
<td>Breast Group 2</td>
<td>-3</td>
<td>GI Group 1</td>
<td>-2</td>
<td>Lymphoma Group 4</td>
<td>-1</td>
</tr>
<tr>
<td>Breast Group 3</td>
<td>-11</td>
<td>GI Group 2</td>
<td>-4</td>
<td>Lymphoma Group 5</td>
<td>-1</td>
</tr>
<tr>
<td>Breast Group 4</td>
<td>-9</td>
<td>GU Group 1</td>
<td>-2</td>
<td>Paediatrics Group 2</td>
<td>-1</td>
</tr>
<tr>
<td>CNS Group 1</td>
<td>-2</td>
<td>GU Group 2</td>
<td>-2</td>
<td>Paediatrics Group 2</td>
<td>0</td>
</tr>
<tr>
<td>CNS Group 2</td>
<td>-1</td>
<td>Gyna Group 1</td>
<td>0</td>
<td>Paediatrics Group 3</td>
<td>-1</td>
</tr>
<tr>
<td>CNS Group 3</td>
<td>-3</td>
<td>Gyna Group 2</td>
<td>-2</td>
<td>Palliative Group 1</td>
<td>-7</td>
</tr>
<tr>
<td>Endocrine Group 1</td>
<td>-3</td>
<td>Head &amp; Neck Group 1</td>
<td>-1</td>
<td>Sarcoma Group 1</td>
<td>-3</td>
</tr>
<tr>
<td>Endocrine Group 2</td>
<td>-1</td>
<td>Lung Group 1</td>
<td>-2</td>
<td>Sarcoma Group 2</td>
<td>-1</td>
</tr>
<tr>
<td>Eye Group 1</td>
<td>-1</td>
<td>Lung Group 2</td>
<td>-7</td>
<td>Sarcoma Group 3</td>
<td>-1</td>
</tr>
<tr>
<td>Eye Group 2</td>
<td>-2</td>
<td>Lymphoma Group 1</td>
<td>-2</td>
<td>Sarcoma Group 4</td>
<td>-2</td>
</tr>
<tr>
<td>Eye Group 3</td>
<td>0</td>
<td>Lymphoma Group 2</td>
<td>-2</td>
<td>Skin Group 1</td>
<td>-3</td>
</tr>
</tbody>
</table>

Overall, we were able to use the PSRH heuristic to design a treatment network that closely replicated the performance of both the existing PM and fully flexible networks, but with many fewer links. Further, the experimental results show that the expected overtime of a fixed network is dependent on the initial patient scheduling decisions. This is an important result when considering networks with limited re-shuffling, and may be relevant in many real world problems where the initial production decisions must be made in advance of any observed uncertainty. In particular, while out of scope of this thesis, this observation suggests that a two-stage optimization model can further improve post-breakdown performance, where the first stage optimizes the initial schedule and the second stage optimizes the recourse scheduling decisions following breakdown. We note that before the PS and PSRH heuristics are run we must first generate the training set and train the CNN, which each take approximately 47 hours and 8 hours respectively. However, we were then able to run the PSRH heuristic in under 20 hours, which predicted the expected max flow of over 90 million network designs. In contrast evaluating 90 million networks with the standard simulation and optimization approach would have taken almost 2 years.
5.4 Redesigning the PM Treatment Network with Homogenous LINACs

In this section, we investigate the impact that operating multiple models of LINACs has on the design of the treatment network, the expected max flow, and sensitivity to initial schedules. To do so, we drop the fourth constraint from problem (5.1) and repeat the analysis from the previous section.

We follow the same procedure from Section 5.3 to train a prediction model. Figure 5.6 summarizes the results from the data generation, prediction model training, and network design process.

Comparing the results from Figure 5.6(d) to those from Figure 5.4, we see that the Expander, PS, and PSRH heuristics again outperform the plant-cover index heuristics. However, one important difference between the homogeneous and heterogenous LINAC cases is that the expected max flow is approximately 30 minutes higher with homogeneous LINACs for the PSRH designed network with 80 arcs. This increase in max flow corresponds to a 27% reduction in overtime. Another difference, summarized in Table 5.4,
is that the expected max flow in the homogeneous LINAC case is much less variable with respect to the initial schedule. The reduction in variability suggests that the initial schedule is a less important consideration for production networks with homogeneous production resources.

Similar to the heterogeneous case, we find that the PSRH algorithm is able to produce network designs that have a similar expected max flow compared to the existing PM treatment network and the fully flexible network. Further, the intermediate networks with 46 and 56 arcs perform comparatively well achieving 99.2% and 99.9% of the expected max flow of the fully flexible network, respectively. These results are summarized in Table 5.4.

Table 5.4: Comparison of Network Designs with Homogenous LINACs

<table>
<thead>
<tr>
<th>Network</th>
<th>Expected Max Flow</th>
<th>Standard Deviation</th>
<th>Number of Arcs</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSRH Designed</td>
<td>9055.28</td>
<td>0.66</td>
<td>46</td>
</tr>
<tr>
<td>PSRH Designed</td>
<td>9119.59</td>
<td>4.21</td>
<td>56</td>
</tr>
<tr>
<td>PSRH Designed</td>
<td>9131.37</td>
<td>2.08</td>
<td>80</td>
</tr>
<tr>
<td>Existing PM</td>
<td>9130.93</td>
<td>1.80</td>
<td>136</td>
</tr>
<tr>
<td>Fully Flexible</td>
<td>9139.05</td>
<td>0.34</td>
<td>540</td>
</tr>
</tbody>
</table>

We may expect that the PSRH algorithm will be able to intuit the distinction in LINAC models through the prediction model training step. If this were the case, we would expect the PSRH algorithm to manifest this understanding by avoiding assigning patient types to more than one type of LINAC. However, as Figure B.8 in Appendix B shows, there is no obvious display of this understanding.

In practice, the decision to have only one type of LINAC versus a mix of LINAC types operating in a cancer center is a complex one. Going with one type of LINAC may give the center some buying power and the ability to negotiate volume discounts or reduce maintenance costs. This thesis also shows that there is clearly a benefit in terms of the potential to reduce overtime costs due to extra flexibility. However, there are also reasons to follow a portfolio approach, such as to avoid being locked into one technology. A middle ground is the potential for software solutions that allow treatments planned one on system to be delivered on another. Effectively, this type of solution would turn heterogeneous LINACs into homogeneous ones, without actually having to modify the hardware. The results in this section show that such a solution, which would be much cheaper to implement than modifying hardware, can have substantial value due to the added flexibility in the system.

5.5 Reducing Variability with Even Use Initial Schedules

In the previous section, we found that operating a fleet of homogeneous LINACs can potentially decrease the expected daily overtime by 27%. We discussed how achieving homogeneity would require
either hardware or software investments. In this section, we investigate easy-to-implement policies for creating initial patient schedules that can close the gap in performance between the heterogeneous and homogeneous settings, and which can be implemented without any of the extra investments mentioned previously. In particular, we adjust the initial patient scheduling model by adding a new objective that aims to evenly distribute treatments across all LINACs.

Suppose we add the following constraint to model (5.4):

\[
\delta \leq \bar{c}_i - \sum_{j: (i,j)} f_{ij}, \quad \forall i \in I,
\]

(5.5)

where \(\delta\) is a continuous variable that represents the minimum slack capacity across all LINACs. We then adjust the objective of (5.4) to maximize \(\delta\) as the secondary objective while maintaining the primary objective of minimizing overtime, and keeping random schedule generation term as a tertiary objective:

\[
\text{minimize} \quad M_1 \sum_{i \in U} o_i - M_2 \delta + \sum_{(i,j) \in A} \xi_{ij} f_{ij},
\]

(5.6)

Note that by definition \(\delta\) increases (5.6) in the presence of any overtime, and only reduces (5.6) if overtime is zero. To insure that the random schedule generation term does not cause unnecessary overtime, or lessen the value of \(\delta\) we choose \(M_1 = M_2 = 100 \times \sum_{j \in J} d_j\) such that the potential reduced cost from using an arc with the smallest possible cost, \(\xi_{ij} = 0.01\) is guaranteed to be smaller then any potential cost incurred by an increase in the overtime or \(\delta\) objective terms.

Making the above adjustments to model (5.4) and reevaluating the networks along the design pathway for the heterogeneous case, we find that the variability with respect to the initial schedule is greatly reduced. In Figure 5.7 we estimate the distribution of expected max flow by evaluating each network 100 times using Algorithm 3. In Figure 5.7(b) we see that for the networks that have between 46 and 55 arcs there is still significant variability. However, once the network becomes flexible enough, the variability of the expected max flow values significantly decreases. Interestingly, the PSRH-designed network with 56 arcs has the lowest variability of any network and slightly higher expected max flow than the subsequent networks. The performance of the 56 arc network suggests that there are networks with just the right amount of flexibility to minimize the variance and achieve most of the performance of the fully flexible network. Table 5.5 summarizes the results with a comparison against the current PM network and the fully flexible network.

Overall, we find that for the PSRH-designed 80 arc network, an initial schedule that more evenly uses LINAC resources results in a 13-minute increase in the expected max flow, corresponding to an
Chapter 5. Radiation Therapy Case Study

48

(a) Full PSRH Design Pathway (b) Tail of PSRH Design Pathway

Figure 5.7: Comparison of Expected Max Flow w.r.t Initial Scheduling Decisions Even Use of LINACs

Table 5.5: Comparison of Network Designs with Even Use Scheduling Policy

<table>
<thead>
<tr>
<th>Network</th>
<th>Expected Max Flow</th>
<th>Standard Deviation</th>
<th>Number of Arcs</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSRH</td>
<td>9058.20</td>
<td>8.73</td>
<td>46</td>
</tr>
<tr>
<td>PSRH</td>
<td>9111.16</td>
<td>0.34</td>
<td>56</td>
</tr>
<tr>
<td>PSRH</td>
<td>9114.63</td>
<td>2.44</td>
<td>80</td>
</tr>
<tr>
<td>Existing PM</td>
<td>9116.21</td>
<td>2.78</td>
<td>136</td>
</tr>
<tr>
<td>Fully Flexible</td>
<td>9118.65</td>
<td>5.14</td>
<td>540</td>
</tr>
</tbody>
</table>

11.5% decrease in daily overtime and a 15-minute, or 86.4%, decrease in the standard deviation. Thus, by modifying the patient scheduling policy, which is an inexpensive intervention, we can capture almost 40% of the difference in overtime between a homogeneous and heterogeneous LINAC network.

5.6 Sensitivity Analysis

The analysis presented in the previous sections was conducted with fixed model parameters $\alpha$ and $\Gamma$. Through our sensitivity analyses, we find that networks designed with one set of parameters tend to perform well for other parameter values. To illustrate this effect, we compare the performance of the PSRH-designed 80 arc network to that of the fully flexible network. We assume that the fleet of LINACs is heterogeneous. To mitigate variability in the results for each $(\alpha, \Gamma)$ parameter pair considered, we evaluate the networks 100 times each using Algorithm 3. Figure 5.8 illustrates the ratio of performance for each parameter pair, on average the PSRH-designed network achieved 99.91% of the expected max flow of the fully flexible network.

Figure 5.9 illustrates the performance of the PSRH and fully flexible networks for a range of $\alpha$ and $\Gamma$ values. We can identify three important operational insights from this figure. The first is that when the total demand equals the total supply in the system ($\alpha = 0$), there is no recourse that reduces overtime.
by shuffling patients. This is intuitively clear since when operating at full capacity, any reduction in regular time supply will cause a one-to-one increase in required overtime. Second, for all levels of $\alpha$, practitioners can achieve the majority of the possible reduction in overtime by reassigning at most 5% of patients. Finally, our model suggests that scheduling excess capacity to reduce overtime may lead to an inefficient use of resources. For example, even with a large re-assignment budget ($\Gamma = 0.2$), reducing expected overtime by 1 minute requires a 9 minute reduction in the number of scheduled treatments.
Figure 5.9: Expected Overtime of Treatment Networks
In this thesis, we have explored the potential impact that machine learning-based methods can have when applied to the classical management science problem of flexible system design. The contributions we make in this thesis are two-fold. First, we proposed novel ML-based design heuristics for the process flexibility problem and unify the existing methods using a common dynamic programming framework. Second, in the case study, we garnered several operational insights.

In the development of our methods, we first propose a novel dynamic programming reformulation of the flexible network design problem. This reformulation allows us to view the existing greedy heuristic approaches within a standardized framework, and provides a convenient frame of reference for discussing the strategies we use in our heuristics to mitigate the short-sightedness that is characteristic of traditional greedy approaches. In particular, we consider adding arcs in batches and reevaluating previously added arcs, which form the basis of the heuristics we refer to as Predict and Search (PS) and Predict and Search with Revisionist History (PSRH). At the core of our approaches is a convolutional neural network (CNN) that significantly accelerates the evaluation of large numbers of candidate designs. During the CNN training procedure we exploited the bipartite structure of the process flexibility problem to efficiently represent a network design as a compact incidence matrix. To evaluate the PS and PSRH heuristics we compared their performance on several test settings from the literature. Notably, the PSRH heuristic was able to recreate the celebrated long chain design for a balanced network.

We then introduce a novel application of process flexibility in healthcare operations by adapting the traditional process flexibility framework to model the clinical operations of the radiation therapy treatment center at Princess Margaret Cancer Centre. This case study highlighted the flexibility of our heuristics, easily incorporating problem specific side constraints. We use our heuristics to show that
Princess Margaret Cancer Centre’s radiation therapy unit can achieve similar performance as the current network with 43% fewer arcs. In our analysis, we found that treatment network designs are sensitive to initial schedules in the presence of the re-shuffling constraints. We showed that this sensitivity can be mostly mitigated by investing in a fleet of homogeneous LINACs, or partially mitigated at a lower cost by scheduling patients such that all LINACs are used equally. In the general manufacturing setting, our results suggest that initial production decisions are an important consideration when they are made before uncertainty in demand or supply levels is observed and if recourse decisions are constrained by the initial decisions. We showed slack capacity, which is a common approach for dealing with machine breakdown, is an inefficient mechanism for reducing expected overtime. To capture the majority of reduction in overtime due to slack capacity up to 7.5%, the re-shuffling budget only needs to be 5% of the total demand. Finally, we conducted a sensitivity analysis of the model parameters that showed the PSRH heuristic produces network designs that tend to perform well across a range of possible parameter values.

Overall, our machine learning-based heuristics performed at or above the level of the existing heuristics, without requiring deep theoretical insights that have been leveraged by previous approaches. However, we note that our approach may be limited for instances of the process flexibility problem that are much larger than that in our case study. Large problem instances present two distinct computational challenges. First, as the underlying network grows more data points need to be generated to train the CNN that is able to make accurate predictions. Second, for larger problem instances the number of possible combinations of two arcs grows quickly. This is especially problematic for the PSRH heuristic since previously added arcs need to be revisited as well.

Based on the work in this thesis we believe there are a number of possibilities for future work. First, the CNN at the core of the PS and PSRH heuristics could be adapted to predict a true cost-to-go function for each network design instead of expected max flow. This adaptation could be incorporated into our current method by generating a set of input-output pairs consisting of network designs and estimated cost-to-go value, and training the CNN on these pairs. Further, the methods presented in this work aimed to demonstrate the potential of incorporating machine learning-based models into operations research solution methods. As such we only considered basic greedy approaches, but believe that the general methodology of exploiting fast neural network predictions could be applied to a variety of more advanced OR methods.

In the case study we found that the initial schedule was an important consideration when creating network designs. In our later experiments, we tried to mitigate the impact of the initial schedule through some simple methods. However, it should be possible to formulate this problem as a two-stage
stochastic optimization problem, considering the initial patient scheduling and network design decisions together. Another avenue for further work is to consider applications of ML-based heuristics in other process flexibility design problems. This thesis nevertheless serves as a first attempt at applying modern ML-based methods to the classical problem of process flexibility.
Bibliography


Geoffrey E. Hinton, Li Deng, Dong Yu, George E. Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, 
for acoustic modeling in speech recognition: The shared views of four research groups. IEEE Signal 


Appendix A

Data Sources

Figure A.1: Data Sources for RMP Production Network
Table A.1: Patient Groups and Demands

<table>
<thead>
<tr>
<th>Group</th>
<th>Technique</th>
<th>Num. Courses Delivered</th>
<th>Minutes per Course</th>
<th>Total Demand (mins)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Breast Group 1</strong></td>
<td>QUICKSTART - 2-Field Tangent Breast</td>
<td>12</td>
<td>250</td>
<td>3000</td>
</tr>
<tr>
<td></td>
<td>QUICKSTART - 2-Field Tangent Breast +/- ABC</td>
<td>19</td>
<td>250</td>
<td>4750</td>
</tr>
<tr>
<td></td>
<td>2-Field Tangents (ABC)</td>
<td>117</td>
<td>250</td>
<td>29250</td>
</tr>
<tr>
<td></td>
<td>2-Field Tangents + Boost (ABC)</td>
<td>72</td>
<td>250</td>
<td>18000</td>
</tr>
<tr>
<td></td>
<td>2-Field Tangents (40/15) (ABC)</td>
<td>1</td>
<td>250</td>
<td>250</td>
</tr>
<tr>
<td></td>
<td>2-Field Tangents (42 4/16) + Boost (ABC)</td>
<td>6</td>
<td>250</td>
<td>1500</td>
</tr>
<tr>
<td></td>
<td>2-Field Tangents (42 4/16) + Boost (ABC)</td>
<td>4</td>
<td>250</td>
<td>1000</td>
</tr>
<tr>
<td><strong>Breast Group 2</strong></td>
<td>4F-Tangent Bilateral Breast (ABC)</td>
<td>2</td>
<td>500</td>
<td>1000</td>
</tr>
<tr>
<td></td>
<td>3 or 4-Field Acym + Boost (ABC)</td>
<td>3</td>
<td>510</td>
<td>16780</td>
</tr>
<tr>
<td></td>
<td>4F Bilateral Breast</td>
<td>103</td>
<td>1050</td>
<td>245</td>
</tr>
<tr>
<td></td>
<td>ABC- LT 4F Acym + IMN</td>
<td>5</td>
<td>510</td>
<td>2550</td>
</tr>
<tr>
<td><strong>Breast Group 3</strong></td>
<td>4F-Tangent Bilateral Breast (ABC)</td>
<td>2</td>
<td>500</td>
<td>1000</td>
</tr>
<tr>
<td></td>
<td>3 or 4-Field Acym (Non-ABC)</td>
<td>79</td>
<td>510</td>
<td>40290</td>
</tr>
<tr>
<td><strong>Breast Group 4</strong></td>
<td>LT 4F Acym + IMN with ABC (Non-ABC)</td>
<td>2</td>
<td>500</td>
<td>1000</td>
</tr>
<tr>
<td><strong>CNS Group 1</strong></td>
<td>Brain Base of Skull w/ MRI</td>
<td>35</td>
<td>185</td>
<td>6475</td>
</tr>
<tr>
<td></td>
<td>Brain w/ MRI</td>
<td>177</td>
<td>185</td>
<td>32745</td>
</tr>
<tr>
<td><strong>CNS Group 2</strong></td>
<td>Stereotactic IMRT Base of Skull / MRI</td>
<td>2</td>
<td>60</td>
<td>120</td>
</tr>
<tr>
<td></td>
<td>Stereotactic IMRT Brain / MRI</td>
<td>1</td>
<td>60</td>
<td>60</td>
</tr>
<tr>
<td><strong>Endocrine Group 1</strong></td>
<td>Thyroid and Thyroid bed IMRT</td>
<td>21</td>
<td>605</td>
<td>12705</td>
</tr>
<tr>
<td><strong>Endocrine Group 2</strong></td>
<td>Stereotactic IMRT with Eye Device / MRI</td>
<td>5</td>
<td>190</td>
<td>950</td>
</tr>
<tr>
<td><strong>Eye Group 1</strong></td>
<td>Electrom (uni- or bilateral)</td>
<td>7</td>
<td>240</td>
<td>1680</td>
</tr>
<tr>
<td><strong>Eye Group 2</strong></td>
<td>IMRT (non-stereotactic)</td>
<td>17</td>
<td>460</td>
<td>7820</td>
</tr>
<tr>
<td><strong>Eye Group 3</strong></td>
<td>Stereotactic IMRT with Eye Device +/- MRI</td>
<td>11</td>
<td>190</td>
<td>2090</td>
</tr>
<tr>
<td><strong>Eye Group 4</strong></td>
<td>Stereotactic IMRT with Eye Device / MRI</td>
<td>11</td>
<td>190</td>
<td>2090</td>
</tr>
<tr>
<td><strong>GI Group 1</strong></td>
<td>SBRT Out-of-Town - same day teaching/simulation</td>
<td>7</td>
<td>210</td>
<td>1480</td>
</tr>
<tr>
<td><strong>GI Group 2</strong></td>
<td>EBRT - EBRT for MRI Guided HDR Boost Study</td>
<td>9</td>
<td>470</td>
<td>4230</td>
</tr>
<tr>
<td><strong>GU Group 1</strong></td>
<td>Bladder with fiducial marker (Lipiodol)</td>
<td>3</td>
<td>670</td>
<td>2010</td>
</tr>
<tr>
<td><strong>GU Group 2</strong></td>
<td>EBRT - EBRT for MRI Guided HDR Boost Study</td>
<td>9</td>
<td>470</td>
<td>4230</td>
</tr>
<tr>
<td><strong>Gynec Group 1</strong></td>
<td>Direct Perineum / Ingino-Femoral</td>
<td>11</td>
<td>360</td>
<td>390</td>
</tr>
<tr>
<td><strong>Gynec Group 2</strong></td>
<td>EBRT - Prostate</td>
<td>2</td>
<td>670</td>
<td>1340</td>
</tr>
<tr>
<td><strong>Head &amp; Neck Group 1</strong></td>
<td>EBRT - Prostate Target Study (Arm 2)</td>
<td>12</td>
<td>580</td>
<td>6960</td>
</tr>
<tr>
<td><strong>Palin Alone</strong></td>
<td>EBRT - Prostate</td>
<td>15</td>
<td>580</td>
<td>8700</td>
</tr>
<tr>
<td><strong>Pelvis</strong></td>
<td>EBRT - Prostate</td>
<td>44</td>
<td>600</td>
<td>26400</td>
</tr>
<tr>
<td><strong>Pelvis + Prostate Bed</strong></td>
<td>EBRT - Prostate</td>
<td>26</td>
<td>611</td>
<td>15886</td>
</tr>
<tr>
<td><strong>Prostate</strong></td>
<td>EBRT - Prostate</td>
<td>2</td>
<td>985</td>
<td>1970</td>
</tr>
<tr>
<td><strong>Prostate</strong></td>
<td>EBRT - Prostate</td>
<td>1</td>
<td>595</td>
<td>595</td>
</tr>
<tr>
<td><strong>Prostate</strong></td>
<td>EBRT - Prostate</td>
<td>111</td>
<td>595</td>
<td>66045</td>
</tr>
<tr>
<td><strong>Prostate</strong></td>
<td>EBRT - Prostate</td>
<td>6</td>
<td>510</td>
<td>3060</td>
</tr>
<tr>
<td><strong>Pelvis</strong></td>
<td>EBRT - Prostate</td>
<td>49</td>
<td>505</td>
<td>24745</td>
</tr>
<tr>
<td><strong>Pelvis + MR Sim for gross recurrence</strong></td>
<td>EBRT - Prostate</td>
<td>8</td>
<td>670</td>
<td>5360</td>
</tr>
<tr>
<td><strong>Seminoma</strong></td>
<td>EBRT - Prostate</td>
<td>4</td>
<td>595</td>
<td>2380</td>
</tr>
<tr>
<td><strong>Endocrine Group 1</strong></td>
<td>EBRT - Prostate</td>
<td>5</td>
<td>310</td>
<td>1550</td>
</tr>
<tr>
<td><strong>Endocrine Group 2</strong></td>
<td>EBRT - Prostate occult/MRI</td>
<td>11</td>
<td>360</td>
<td>390</td>
</tr>
<tr>
<td><strong>Eye Group 1</strong></td>
<td>EBRT - Prostate</td>
<td>11</td>
<td>360</td>
<td>390</td>
</tr>
<tr>
<td><strong>Eye Group 2</strong></td>
<td>EBRT - Prostate</td>
<td>13</td>
<td>360</td>
<td>390</td>
</tr>
<tr>
<td><strong>Prostate</strong></td>
<td>EBRT - Prostate</td>
<td>13</td>
<td>360</td>
<td>390</td>
</tr>
<tr>
<td><strong>Prostate</strong></td>
<td>EBRT - Prostate</td>
<td>1</td>
<td>360</td>
<td>360</td>
</tr>
<tr>
<td><strong>Prostate</strong></td>
<td>EBRT - Prostate</td>
<td>11</td>
<td>360</td>
<td>360</td>
</tr>
<tr>
<td><strong>Prostate</strong></td>
<td>EBRT - Prostate</td>
<td>1</td>
<td>360</td>
<td>360</td>
</tr>
<tr>
<td><strong>Prostate</strong></td>
<td>EBRT - Prostate</td>
<td>11</td>
<td>360</td>
<td>360</td>
</tr>
</tbody>
</table>
### Appendix A. Data Sources

Table A.1: Patient Groups and Demands

<table>
<thead>
<tr>
<th>Group</th>
<th>Technique</th>
<th>Num. Courses Delivered</th>
<th>Minutes per Course</th>
<th>Total Demand (mins)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMRT 1- or 2-phase</td>
<td></td>
<td>419</td>
<td>880</td>
<td>368720</td>
</tr>
<tr>
<td>IMRT 1- or 2-Phase w with fusion MRI</td>
<td></td>
<td>150</td>
<td>1005</td>
<td>150750</td>
</tr>
<tr>
<td>IMRT DAHANCA 70Gy/35/6 weeks</td>
<td></td>
<td>10</td>
<td>880</td>
<td>8800</td>
</tr>
<tr>
<td>IMRT-BID Daily / HARDWINS w with fusion MRI</td>
<td></td>
<td>3</td>
<td>1005</td>
<td>3015</td>
</tr>
<tr>
<td>IMRT-BID Daily/HARDWINS</td>
<td></td>
<td>5</td>
<td>1005</td>
<td>5025</td>
</tr>
<tr>
<td>Lung with ABC</td>
<td></td>
<td>1</td>
<td>610</td>
<td>610</td>
</tr>
<tr>
<td>Mesothelioma - 50 Gy / 25 Rx</td>
<td></td>
<td>2</td>
<td>885</td>
<td>1770</td>
</tr>
<tr>
<td>Mesothelioma- 25Gy/5 + concomitant boost to GTV (5Gy/5)</td>
<td></td>
<td>17</td>
<td>185</td>
<td>3145</td>
</tr>
<tr>
<td>NSCLC, Single Fraction</td>
<td></td>
<td>49</td>
<td>510</td>
<td>24990</td>
</tr>
<tr>
<td>NSCLC - 66 Gy / 33 Rx</td>
<td></td>
<td>45</td>
<td>510</td>
<td>22950</td>
</tr>
<tr>
<td>Other</td>
<td></td>
<td>18</td>
<td>395</td>
<td>7110</td>
</tr>
<tr>
<td>Pre-Op POP - 45 Gy / 25 Rx</td>
<td></td>
<td>2</td>
<td>370</td>
<td>740</td>
</tr>
<tr>
<td>SBRT LUNG - 54Gy / 3 Rx</td>
<td></td>
<td>5</td>
<td>180</td>
<td>900</td>
</tr>
<tr>
<td>Lung Group 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SBRT Lung</td>
<td></td>
<td>55</td>
<td>180</td>
<td>9900</td>
</tr>
<tr>
<td>SBRT Lung - 48 Gy / 4 Rx</td>
<td></td>
<td>52</td>
<td>180</td>
<td>9360</td>
</tr>
<tr>
<td>SBRT Lung - 60 Gy / 8 Rx - DAILY XRT</td>
<td></td>
<td>19</td>
<td>180</td>
<td>3420</td>
</tr>
<tr>
<td>SCLC</td>
<td></td>
<td>14</td>
<td>465</td>
<td>6510</td>
</tr>
<tr>
<td>SCLC - 40Gy 15 Gy / 15 Rx (3 -bid)</td>
<td></td>
<td>5</td>
<td>465</td>
<td>2325</td>
</tr>
<tr>
<td>SCLC - 45 Gy / 30 Rx BID</td>
<td></td>
<td>4</td>
<td>465</td>
<td>1860</td>
</tr>
<tr>
<td>Thorax POP - 30 Gy / 10 Rx</td>
<td></td>
<td>2</td>
<td>90</td>
<td>180</td>
</tr>
<tr>
<td>Thymoma</td>
<td></td>
<td>4</td>
<td>390</td>
<td>1560</td>
</tr>
<tr>
<td>FAZA Lung Activated Study</td>
<td></td>
<td>3</td>
<td>90</td>
<td>270</td>
</tr>
<tr>
<td>Lung Group 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FAZA/FLT PET Lung Activated Study</td>
<td></td>
<td>1</td>
<td>90</td>
<td>90</td>
</tr>
<tr>
<td>PCI POP - 25 Gy / 16 Rx</td>
<td></td>
<td>4</td>
<td>90</td>
<td>360</td>
</tr>
<tr>
<td>PCI/Whole Brain</td>
<td></td>
<td>3</td>
<td>90</td>
<td>270</td>
</tr>
<tr>
<td>Lymphoma Group 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Breast IMRT</td>
<td></td>
<td>1</td>
<td>240</td>
<td>240</td>
</tr>
<tr>
<td>Other - IMRT</td>
<td></td>
<td>45</td>
<td>310</td>
<td>13950</td>
</tr>
<tr>
<td>Lymphoma Group 2</td>
<td>Abdomen - 4DCT</td>
<td>6</td>
<td>310</td>
<td>1860</td>
</tr>
<tr>
<td>Abdomen - Simple technique/POP</td>
<td></td>
<td>3</td>
<td>240</td>
<td>720</td>
</tr>
<tr>
<td>Mantle Mod No ABC</td>
<td></td>
<td>1</td>
<td>240</td>
<td>240</td>
</tr>
<tr>
<td>CSI - Spine Whole CNS</td>
<td></td>
<td>3</td>
<td>465</td>
<td>1395</td>
</tr>
<tr>
<td>Gastric Lymphomas</td>
<td></td>
<td>6</td>
<td>310</td>
<td>1860</td>
</tr>
<tr>
<td>Lymphoma Group 3</td>
<td>IMRT Head &amp; Neck</td>
<td>14</td>
<td>240</td>
<td>3360</td>
</tr>
<tr>
<td>IMRT Head &amp; Neck with Mask</td>
<td></td>
<td>21</td>
<td>240</td>
<td>5040</td>
</tr>
<tr>
<td>Other - Simple technique/POP</td>
<td></td>
<td>2</td>
<td>240</td>
<td>480</td>
</tr>
<tr>
<td>Mantle Mod. ABC</td>
<td></td>
<td>19</td>
<td>615</td>
<td>11685</td>
</tr>
<tr>
<td>Mantle Mod. ABC with Contrast</td>
<td></td>
<td>14</td>
<td>615</td>
<td>8610</td>
</tr>
<tr>
<td>TBI - 2 Fractions / Day</td>
<td></td>
<td>64</td>
<td>225</td>
<td>14400</td>
</tr>
<tr>
<td>TBI - 6 Fractions Start in the AM</td>
<td></td>
<td>3</td>
<td>240</td>
<td>720</td>
</tr>
<tr>
<td>TBI - 6 Fractions Start in the AM - LINAC</td>
<td></td>
<td>7</td>
<td>240</td>
<td>1680</td>
</tr>
<tr>
<td>Lymphoma Group 5</td>
<td>TBI - Single Fraction</td>
<td>20</td>
<td>75</td>
<td>1500</td>
</tr>
<tr>
<td>TBI - Single Fraction LINAC</td>
<td></td>
<td>33</td>
<td>75</td>
<td>2475</td>
</tr>
<tr>
<td>IMRT (CNS)</td>
<td></td>
<td>30</td>
<td>980</td>
<td>29400</td>
</tr>
<tr>
<td>Paediatrics Group 1</td>
<td>IMRT (Other)</td>
<td>19</td>
<td>1132.5</td>
<td>21517.5</td>
</tr>
<tr>
<td>POP (CNS &amp; Other)</td>
<td></td>
<td>12</td>
<td>360</td>
<td>5760</td>
</tr>
<tr>
<td>Whole CNS</td>
<td></td>
<td>12</td>
<td>805</td>
<td>9660</td>
</tr>
<tr>
<td>Paediatrics Group 2</td>
<td>lymphoma ABC (IMRT/POP)</td>
<td>5</td>
<td>705</td>
<td>4250</td>
</tr>
<tr>
<td>Paediatrics Group 3</td>
<td>TBI</td>
<td></td>
<td>16</td>
<td>360</td>
</tr>
<tr>
<td>4DCT Retroperitoneal Sarcoma Treatment</td>
<td></td>
<td>16</td>
<td>370</td>
<td>9120</td>
</tr>
<tr>
<td>CT w Fusion MRI pre or post op</td>
<td></td>
<td>8</td>
<td>510</td>
<td>4080</td>
</tr>
<tr>
<td>Lower Extremity IMRT 25 Fractions</td>
<td></td>
<td>59</td>
<td>625</td>
<td>36875</td>
</tr>
<tr>
<td>Lower Extremity IMRT Post-op w Possible Boost - 33Fx</td>
<td></td>
<td>9</td>
<td>670</td>
<td>6030</td>
</tr>
<tr>
<td>Lower Extremity IMRT/VMAT 25 Fractions</td>
<td></td>
<td>8</td>
<td>625</td>
<td>5000</td>
</tr>
<tr>
<td>Upper Extremity - IMRT Post-op w Possible Boost - 33 Fx</td>
<td></td>
<td>8</td>
<td>510</td>
<td>3060</td>
</tr>
<tr>
<td>Upper Extremity IMRT 25 fractions</td>
<td></td>
<td>34</td>
<td>510</td>
<td>17340</td>
</tr>
<tr>
<td>Skin Group 1</td>
<td>CT SIM &gt; All Other Body Locations</td>
<td></td>
<td>6</td>
<td>875</td>
</tr>
<tr>
<td>CT SIM &gt; Axilla/SCF</td>
<td></td>
<td>7</td>
<td>400</td>
<td>2800</td>
</tr>
<tr>
<td>CT SIM &gt; H-N Immobilization</td>
<td></td>
<td>2</td>
<td>400</td>
<td>800</td>
</tr>
<tr>
<td>CT SIM &gt; H-N Immobilization &gt; Complex Bolus</td>
<td></td>
<td>2</td>
<td>400</td>
<td>800</td>
</tr>
</tbody>
</table>
Appendix B

Treatment Network Designs

Here we include figures illustrating different treatment network designs considered in the analysis in Chapter 5.
Figure B.1: Initial PM Treatment Network
Figure B.2: PSRH Designed 80 Arc Heterogeneous Linac Network
Figure B.3: PSRH Designed 56 Arc Heterogeneous Linac Network
Figure B.4: PSRH Designed 46 Arc Heterogeneous Linac Network
Figure B.5: PSRH Designed 80 Arc Homogenous Linac Network
Figure B.6: PSRH Designed 56 Arc Homogenous Linac Network
Figure B.7: PSRH Designed 46 Arc Homogenous Linac Network
Appendix B. Treatment Network Designs

(a) 80 Arc Homogeneous Linac Network

(b) 80 Arc Heterogeneous Linac Network

(c) 56 Arc Homogeneous Linac Network

(d) 56 Arc Heterogeneous Linac Network

(e) 46 Arc Homogeneous Linac Network

(f) 46 Arc Heterogeneous Linac Network

Figure B.8: Patient Types Assigned to More Than One Linac