NTNU
Norwegian University of
Science and Technology

Multiobjective Evolutionary Surgery Scheduling under Uncertainty

Jacob Henrik Nyman

Master of Science in Computer Science
Submission date: June 2018
Supervisor: Kazi S. N. Ripon, IDI

Norwegian University of Science and Technology
Department of Computer Science

To mom and dad

Summary

Surgery scheduling is the task of allocating resources to surgical procedures over time. Because hospitals are dynamic environments governed by uncertainty, difficult priorities and careful coordination of scarce resources, machine scheduling algorithms are generally not directly applicable to surgery scheduling problems. This thesis attempts to develop new genetic operators that can integrate the power of evolutionary machine scheduling algorithms with the dynamics that are present in real life hospitals. Our most important contribution is a fast simulation approach for evaluating surgery schedules under surgery duration uncertainty. We implement the developed genetic operators and apply the strength pareto evolutionary algorithm 2 and non-dominated-sorting genetic algorithm II to the multiobjective surgery admission planning problem. In this problem, patients are selected from a patient waiting list and scheduled into operating rooms over a weekly period in a way that maximizes resource utilization and minimizes patient waiting time on the waiting list. The algorithm output is a set of detailed and compatible surgeon and operating room schedules that are robust against surgery duration variability. Our analysis exemplifies how significant overtime reduction can be achieved by incorporating uncertainty in the optimization process. Because handling multiple objectives and uncertainty simultaneously quickly becomes complicated, we acknowledge that hospital managers preference regarding performance trade-offs and risk should be integrated in the future.

Sammendrag

Operasjonsplanlegging handler om å sette av resurser til å gjennomføre kirurgiske operasjoner over tid. Siden sykehus er dynamiske miljøer preget av usikkerhet og tøffe prioriteringer, kan ikke klassiske planleggingsalgoritmer brukes på operasjonsplanleggingsproblemer uten modifikasjoner. Denne oppgaven prøver å utvikle nye genetiske mekanismer for å løse realistiske operasjonsplanleggingsproblemer ved hjelp av evolusjonære algoritmer. Vårt viktigste bidrag er en rask metode for å simulere styrken til operasjonsplaner når operasjonenes varighet er usikker. Vi implementerer de utviklede genetiske mekanismene i to kjente evolusjære algoritmer, "strength Pareto evolutionary algorithm 2" og "non-dominated-sorting genetic algorithm II", for å løse et operasjonsplanleggingsproblem med flere samtidige målsetninger. Problemet baserer seg på en liste med ventende pasienter som skal opereres i løpet av en ukentlig periode, hvor målsetningene er å redusere pasientenes ventetid, samt å bruke leger og operasjonssaler så godt som mulig. Algoritmene produserer planer for leger og operasjonssaler som er lite utsatt for variasjoner i operasjonenes varighet. Vi viser, med eksempler, at overtiden for leger og operasjonssaler kan reduseres betraktelig ved å ta hensyn til usikkerheten under løsningsprosessen. Siden det å håndtere motstridende målsetninger og usikkerhet samtidig fort blir komplisert, anerkjenner vi at sykehusledere burde involveres i framtiden for å integrere preferanser når det kommer til risiko og avveininger mellom de ulike målsetningene.

Preface

This thesis is the final delivery in a five year Master's Degree in Computer Science attained at the Norwegian University of Science and Technology in Trondheim, Norway. The presented work was conducted during the spring of 2018 and extends a project on the same subject performed during the fall of 2017 (Nyman, 2017). Other relevant work from the same author includes a Master's Thesis submitted within the field of applied economics and optimization (Nyman, 2016) and a conference paper for the IEEE Congress on Evolutionary Computation 2018 (Nyman and Ripon, 2018).

Any credit should be shared with my inspiring supervisor, Kazi Shah Nawaz Ripon, who has eagerly supported and guided me throughout the project period. Appropriate thanks should be given to my close friends and family, who always encourage my social and scientific aspirations. A special thanks to my girlfriend, Ingrid, who always believes in me.

Table of Contents

Summary i

Sammendrag ii

Preface iii

Table of Contents vii

List of Tables ix

List of Figures xii

1 Introduction 1
- 1.1 Thesis Objectives . . . 3
- 1.2 Research Questions . . . 3
- 1.3 Structure of Report . . . 3

2 Background and Basic Theory 5
- 2.1 The Surgery Scheduling Process . . . 5
- 2.2 Taxonomic Description of Surgery Scheduling Problems . . . 6
  - 2.2.1 Patient Characteristics . . . 6
  - 2.2.2 Performance Measures . . . 6
  - 2.2.3 Decision Delineations . . . 9
  - 2.2.4 Uncertainty . . . 10
  - 2.2.5 Research Methodology . . . 11
  - 2.2.6 Applicability of Research . . . 11
- 2.3 Conceptual Description of the Problem Under Investigation . . . 11
- 2.4 Machine Scheduling Problems . . . 12
  - 2.4.1 A Selection of Well-known Machine Scheduling Problems . . . 13
  - 2.4.2 Genetic Algorithms for Solving Scheduling Problems . . . 14
- 2.5 Multiobjective Optimization . . . 15

2.5.1 Conflicting Objectives and Dominance . . . 15
2.5.2 The Goal of Multiobjective Optimization . . . 16
2.5.3 Traditional Approaches to Multiobjective Optimization . . . 17
2.5.4 Multiobjective Evolutionary Algorithms (MOEAs) . . . 18
2.5.5 Measuring Performance of MOEAs . . . 18
2.6 The Wilcoxon Signed-Rank Test . . . 23
2.7 Optimization under Uncertainty . . . 24
2.7.1 Notions of Uncertainty in the Optimization Process . . . 24
2.7.2 Classical Approaches to Optimization Under Uncertainty . . . 25
2.7.3 Evolutionary Optimization Under Uncertainty . . . 25
2.7.4 MOEAs in Uncertain Environments . . . 26

3 Literature Review **29**
3.1 Heuristic Approaches to Surgery Scheduling . . . 29
3.1.1 Rule-based Heuristics . . . 30
3.1.2 Metaheuristics . . . 30
3.1.3 Multiobjective Heuristics . . . 30
3.2 Multi-stage and Multi-resource Problems . . . 31
3.3 Surgery Scheduling under Duration Uncertainty . . . 32
3.3.1 Stochastic Programs . . . 32
3.3.2 Chance-constrained Optimization . . . 32
3.3.3 Robust Optimization . . . 33
3.3.4 Multiobjective Surgery Scheduling under Duration Uncertainty . 33

4 Detailed Problem Formulation **35**
4.1 Model Assumptions . . . 35
4.2 Mathematical Problem Formulation . . . 36

5 Implemented Algorithms **41**
5.1 Genetic Submechanisms . . . 41
5.1.1 Solution Representation . . . 41
5.1.2 Decoding . . . 42
5.1.3 Fitness Evaluation . . . 44
5.1.4 Crossover . . . 46
5.1.5 Mutation . . . 48
5.1.6 Population Initialization . . . 50
5.2 SPEA2 . . . 51
5.3 NSGA-II . . . 53
5.4 Hybrid GA for Singleobjective Optimization . . . 55
5.4.1 Variable Neighbourhood Decent . . . 57
5.4.2 Hybridizing a Simple GA . . . 57

6 Experimental Results **59**
6.1 Test Data . . . 59
6.1.1 Parameters . . . 60
6.1.2 Instance Generation . . . 60

6.1.3 Surgery Properties 62
6.2 Experimental Setup 62
6.2.1 Calibration of Hyperparameters 63
6.3 Preliminary Analysis 63
6.3.1 Solution Space and Pareto Front Properties 64
6.3.2 Degree of Conflict Between Objectives 64
6.3.3 The Effect of Uncertainty 66
6.4 Comparison of SPEA2 and NSGA-II 69
6.4.1 Wilcoxon Signed-Rank Test Results 69
6.5 Dealing With Uncertainty 71
6.5.1 Explicit Averaging 72
6.5.2 Comparison with a Deterministic Approach 76
6.6 Discussion 78
6.7 Further Work 80

7 Conclusion **83**

Bibliography **85**

List of Tables

6.1 Nine different instances 62
6.2 MOEA performance measures summary 70
6.3 Critical values for the wilcoxon signed rank test 70
6.4 Comparison under uncertainty 71
6.5 Estimate of computation time (in minutes) needed for implicit averaging . 73

List of Figures

2.1 The daily problem 12
2.2 Solutions in multiobjective optimization 16

4.1 Model notation 40

5.1 Two-vector chromosome representation 42
5.2 Idle time intervals 43
5.3 Decoded chromosome 44
5.4 Realized schedule 45
5.5 Crossover operation 47
5.6 Parent 1 47
5.7 Parent 2 48
5.8 Child 48
5.9 Swap mutation operators 49
5.10 Room-day swap mutation 49
5.11 Precedence mutation 50
5.12 Renew room-day mutation 50
5.13 NSGA-II sorting procedures 53
5.14 Partition into fronts 55

6.1 Instance classes created from a real-life master surgery schedule 61
6.2 Optimized operating room overtime for $I_1^{(2)}$ 65
6.3 Optimized operating room idle time for $I_1^{(2)}$ 66
6.4 Optimized surgeon idle time for $I_1^{(2)}$ 67
6.5 Optimized waiting time for $I_1^{(2)}$ 67
6.6 Optimized weighted sum for $I_1^{(2)}$ 68
6.7 Realized schedule of $I_1^{(1)}$ optimized with weighted sum approach 68
6.8 Explicit averaging 75
6.9 Deterministic vs. stochastic solutions 77

6.10 Comparison of waiting time objective using deterministic and stochastic approach 78

Chapter 1

Introduction

Surgery scheduling is an operational task that requires coordination of scarce resources, thoughtful prioritization of conflicting interests and constant response to unexpected events such as delays and cancellations (Cardoen et al., 2010). Labour-intensive scheduling activities are critical in most hospitals and directly linked to the rate at which patients receive treatment and the quality of care provided to each patient. Optimization of the surgery scheduling process is a topic of increasing interest as it can improve schedule quality in order to better utilize surgeons, nurses, equipment and operating rooms. Thus, high quality surgery schedules have the potential of improving hospitals core business without incurring costs associated with upscaling of resources.

Hospitals maintain a list of patients in need of surgery (Riise and Burke, 2011). Each surgery must be performed by a qualified surgeon in an operating room that facilitates the required equipment. In order to simplify and control the scheduling process, it is common to adhere to a master surgery schedule; a periodic time table that reserves time slots in operating rooms for certain surgical subspecialties (Riise and Burke, 2011). Most surgeries may be planned for well in advance so as to ensure smooth coordination of staff (e.g. surgeons, nurses, anesthesiologists), facilities (e.g. operating rooms, post-anesthesia care unit, intensive care unit, hospital beds) and equipment. However, carefully designed schedules are continuously disrupted by unpredictable events such as the arrival of emergency patients and delays in the operating room (Cardoen et al., 2010). It is up to the schedulers to balance the need of patients waiting for surgery against the schedule density, risk of overtime and surgery cancellations. This tedious coordination, prioritization and risk evaluation has been subject to automation in recent years.

Challenges identified in recent surgery schedule optimization literature include
(i) how to delineate the scheduling problem in a computationally tractable manner,
(ii) how to cope with multiple conflicting objectives and
(iii) how to manage the uncertainty inherent in surgery durations.
This thesis aims to deal with these challenges by applying two state of the art multiob-

jective evolutionary algorithms (MOEAs), namely the strength pareto evolutionary algorithm 2 (SPEA2) (Zitzler et al., 2001) and the non domination sorting genetic algorithm II (NSGA-II) (Deb et al., 2002), to a new variation of the surgery admission planning problem (Riise and Burke, 2011). In the presented problem, patients from a patient waiting list are to be scheduled in operating rooms over a period of one or several days. There are multiple operating rooms available, and surgeons may switch between them in order to meet equipment requirements or tighten the overall schedule. We minimize five objectives:

(i) patient waiting time on the waiting list,
(ii) surgeon overtime,
(iii) surgeon idle time,
(iv) operating room overtime and
(v) operating room idle time.

Each solution is a detailed schedule for operating rooms and surgeons that should lead to high performance even if surgeries last longer or shorter than expected. As in general multiobjective optimization, the goal is to present a diverse set of high-quality trade-off solutions to the decision maker within a reasonable amount of computation time.

The motivation of this research is to narrow the gap between theoretical surgery scheduling analysis and actual use of surgery scheduling algorithms. We address how the SPEA2 and NSGA-II can be applied with new genetic mechanisms to solve surgery scheduling problem instances that are realistic in terms of problem size, multiobjectivity and uncertainty. While multiobjectivity and uncertainty have been considered separately in many recent papers on surgery scheduling, few consider them in combination. Thus, our approach rather uniquely integrates two well-acknowledged complications in an already complicated problem. The underlying assumption is that surgery scheduling is still performed manually in many hospitals, partly because surgery scheduling algorithms fail to accurately capture the dynamics in real hospitals. Our hope is that the presented work may contribute to more manageable daily operations at hospitals in general.

We assess the quality of the implemented MOEAs by solving instances based on the situation in a real-life Norwegian hospital. Because optimal solutions to the investigated instances are not known, the study is a comparative one, where several well-known performance measures (Ripon et al., 2007; Yen and He, 2014) are used to compare Pareto-optimality, distribution, and spread of the SPEA2 and NSGA-II. As is common in surgery schedulig literature (Addis et al., 2014), we model duration uncertainty by assuming log-normal distributions for surgery durations. We use a relatively simple approach for evaluating fitness under uncertainty during evolution in the MOEAs (Jin and Branke, 2005), namely explicit averaging using Monte-Carlo simulation. Furthermore, we investigate the benefit of incorporating uncertainty during optimization.

We model the surgery problem mathematically and modify instances from previous work on the same problem (Nyman, 2017). All algorithms are implemented in Java and the experiments are run on an Asus laptop with a 0.80 GHz Intel multi core processor with 8 GB of RAM. Our most important contribution is the development of new genetic mechanisms for representing, decoding, mutating and crossing over surgery schedules in an efficient manner. Our analysis revolves around how uncertainty may efficiently be incorporated in multiobjective search.

1.1 Thesis Objectives

In order to facilitate an analysis of SPEA2 and NSGA-II applied to the surgery admission planning problem, the following objectives will be covered:

- Model a new variation of the surgery admission planning problem.
- Develop new genetic mechanisms that enable the presented problem to be solved with evolutionary algorithms.
- Generate realistically sized instances to experiment with.
- Implement and compare SPEA2 and NSGA-II with Monte-Carlo simulation.

1.2 Research Questions

This study focuses on efficiently finding multiple distinguished high-quality trade-off solutions to the surgery admission planning problem that are resistant to surgery duration variability. Our aim is to contribute to answering the larger research question:

Can algorithms efficiently solve realistic instances of surgery scheduling problems?

To put our contribution in perspective, we will review the field of surgery scheduling and shed light upon a set of more detailed questions:

- What performance measures are appropriate to include in a realistic surgery scheduling problem?
- Are MOEAs appropriate for solving multiobjective surgery scheduling problems?
- How should sources of uncertainty be modelled and accounted for in surgery scheduling algorithms?

This work is intended to be a contribution to both surgery scheduling literature and general literature on population based methods for complex problem solving. This thesis is not a case study of the mentioned Norwegian hospital. The hospital serves as a context and provider of data that enables generation of realistic problem instances.

1.3 Structure of Report

The thesis is organized as follows. Chapter 2 gives an introduction to surgery scheduling and outlines basic theory on general scheduling, multiobjective optimization and optimization under uncertainty. In Chapter 3, we present an updated literature review on solution methods for solving varieties of surgery scheduling problems. Chapter 4 details the mathematical problem formulation, and Chapter 5 describes the implemented algorithms. We describe the instances and analyze the results in Chapter 6 before giving our recommendations for future research. Finally, we conclude the thesis in Chapter 7.

Chapter 2

Background and Basic Theory

This chapter gives an introduction to surgery scheduling and fundamental theory for understanding the methods used in this thesis. We include a description of the surgery scheduling process, a taxonomic description of surgery scheduling problems and a conceptual description of the problem under investigation. Theory on general scheduling problems, multiobjective optimization and optimization under uncertainty is provided in the last three sections of this chapter.

2.1 The Surgery Scheduling Process

The surgery scheduling process is described in various ways, but may be generalized as follows (Riise and Burke, 2011). The surgery scheduling process starts when the surgery of a patient is acknowledged by a certain hospital and ends when the patient leaves the hospital after completed surgery. The set of detailed activities and resources needed to complete the surgery process varies among hospitals, and among patients within the same hospital. Typically, once surgery is acknowledged, the patient is assigned a surgeon and put on the patient waiting list. The schedulers look for an appropriate surgery date and inform the patient once a date is set. Next, the room and exact starting time and duration for the surgery may be decided. Decisions regarding day, operating room and time interval for surgery may be dependent on a master surgery schedule. The master surgery schedule reserves time in operating rooms for certain surgical subspecialties throughout a time period of e.g. one week. The use of a master surgery schedule simplifies the coordination of resources needed to perform detailed activities on the day of surgery. Daily coordination activities include personnel rostering of surgeons, anesthesiologists and nurses, facility scheduling of hospital beds, operating rooms, post-anesthesia care unit and intensive care unit, as well as coordination of equipment such as prostheses and surgical tools. A patient may be involved with some or all of these resources before the surgery is completed.

2.2 Taxonomic Description of Surgery Scheduling Problems

Operational studies on surgery scheduling use different terminology for describing particular problems and problem scopes. In the following, we present how surgery scheduling problem varieties are unusually distinguished in literature. Surgery scheduling manuscripts may be categorized using six descriptive fields as defined in (Cardoen et al., 2010):
(i) patient characteristics,
(ii) performance measures,
(iii) decision delineation,
(iv) uncertainty,
(v) research methodology and
(vi) applicability of research.
We elaborate each of these fields in the following subsections.

2.2.1 Patient Characteristics

When dealing with a surgery scheduling problem, it is essential to known if patients are
(i) elective or non-elective and
(ii) inpatients or outpatients.
Elective patients may be planned for well in advance because their condition will not worsen over time if surgery is not performed. On the other hand, non-elective patients are in need of immediate or urgent care and postponement of surgery may critically worsen the patients condition. The arrival of non-elective emergency or urgency cases is unpredictable and requires swift mobilization of necessary resources. When the same set of resources are allocated to the treatment of elective and non-elective patients, this mobilization often interferes with the elective schedule. Therefore, it is not uncommon to separate treatment of elective and non-elective patients completely. Alternatively, schedulers can add slack in the elective patient schedule and practice a rescheduling policy for fast and efficient re-coordination of resources.

The distinction between inpatients and outpatients is made between patients that must be hospitalized and patients that arrive and leave at the day of surgery, respectively. When scheduling surgery for inpatients, other resources such as hospital bed capacity and nurses must be considered in addition to the resources needed to complete the surgery. Furthermore, outpatient surgeries are usually shorter and less variable than inpatient surgeries.

2.2.2 Performance Measures

Schedule quality may be evaluated using different performance measures. Specific performance measures mentioned in (Cardoen et al., 2010) include waiting time, utilization, throughput, leveling, makespan, patient deferrals, preferences and financial measures. We

summarize the interpretation of these performance measures in the context of surgery scheduling in the following overview, and subsequently discuss how they are interrelated.

Minimization of waiting time may refer to reducing
(i) the number of days patients spend on the patient waiting list,
(ii) the time patients must wait on the day of surgery, or
(iii) the time surgeons must wait before and in between surgeries.

Maximization of utilization means to simultaneously minimize resource idle time and overtime.

Maximization of throughput means to treat as many patients as possible within a given time frame.

Maximization of leveling favors schedules without tendencies of capacity overload, i.e. schedules where the workload is well distributed among resources over time.

Minimization of makespan means to narrow the time frame in which surgery is performed on a given day or session.

Minimization of patient deferrals means to avoid cancelling surgeries as a consequence of delays earlier in the day.

Minimization of patient inconvenience[1] refers to prioritization (i.e. early scheduling) of special patients such as children, elderly and patients with a long travelling distance to the hospital.

Maximization of profits is a general form of measuring schedule quality in which several of the aforementioned measures are quantified into profits and costs and aggregated.

Whenever surgery of a patient is acknowledged, it is desirable to perform the surgery as soon as possible and minimize the patient waiting time. We will use the term "waiting time" and "waiting cost" interchangeably in this thesis. Even if the condition of an elective patient is not worsening as a consequence of postponement, the patient experiences lower quality of life before surgery than after. Hospitals usually operate with treatment deadlines that are set internally (e.g. by management) or externally (e.g. by government regulations). A common performance measure is to ensure that these deadlines are met. It is also possible to model the waiting cost as a function of the referral date and deadline that increases non-linearly with the decided scheduling date (Riise and Burke, 2011). Another notion of waiting time is the time patients must wait on the day of surgery. By requiring patients to show up early, the risk of patient delay decreases, and there may be time for rescheduling in the case of no-shows. However, especially for patients with generally poor health, waiting may be stressful and reduce the experienced treatment quality for patients. Waiting may also occur as a consequence of delays at the hospital, for instance if preceding surgeries in the same operating room lasts longer than expected. The third notation of waiting time, surgeon waiting time, is essentially the same as surgeon idle time, which we address in the context of resource utilization.

[1] the original term in (Cardoen et al., 2010) is "preference" and refers not only to patient preference, but also to preference of surgeons and other internal staff.

In surgery scheduling, the term utilization is often used interchangeably with resource efficiency, and the maximization of utilization means to simultaneously minimize resource idle time and overtime. For surgeons, this means that there should be as little waiting time as possible throughout the day and as little overtime as possible at the end of the day. This represents a trade-off because minimizing idle time leads to dense schedules in which the slightest delay can lead to overtime, while minimization of overtime leads to planned slack and waiting in between surgeries to hedge for potential delays. Furthermore, full utilization of a certain resource is likely to compromise the utilization of other resources. For instance, even if the operating room is fully utilized, the inpatient resources may be underutilized if only outpatients are scheduled within a certain time-frame.

The throughput objective measures the number of patients treated, and maximizing throughput is related to shortening the patient waiting list as much as possible. However, these measures may also be in partial conflict. This is because maximizing throughput favors shorter surgeries and disregards the long waiting time for patients waiting for comprehensive and lengthy surgeries. The throughput objective may also be in conflict with operating room utilization when an amount of cleaning time is needed in between surgeries (Marques et al., 2015) because increasing the number of scheduled surgeries also increases the time spent on cleaning.

The leveling performance objective is used to avoid capacity problems that occur when disruptions such as delays or emergency patient arrivals happen when resources are fully occupied. A schedule where the workload is well distributed among resources is less vulnerable to disruptions than a schedule allowing peaks. The makespan measure indicates the time between the start of the first surgery and end of the last surgery. Minimizing the makespan leads to dense schedules, and often levels the workload on resources. Dense schedules leave minimal slack in between surgeries, and there is little room for delays. Because of this, other measures taking duration uncertainty into account should be considered together with the makespan. When delay is substantial, surgeries at the end of the day may be cancelled to avoid unreasonable overtime and staff fatigue. Cancellations are expensive, inconvenient for the patient and the surgery must be rescheduled to another day. Therefore, the number of cancellations, or patient deferrals, is sometimes minimized to prioritize the quality of care given to all patients. Because cancelling surgery of prioritized patients such as children, elderly or patients with a long travelling distance to the hospital is more serious than cancelling other surgeries, some include the performance measure of patient preference. Typically, schedules where these patients are scheduled early in the morning are preferred because the realized timing of the first surgeries is unaffected by delays of previous surgeries.

The aforementioned performance measures may be converted to financial measures by assigning costs or profits to the different components of realized schedules. Treatment of a patient is associated with a profit while e.g. resource overtime is associated with costs. However, not all measures are easily quantified in terms of costs or profits. For instance, determining the idle time cost of an operating room or the cost of cancelling a surgery requires some analysis and approximation.

2.2.3 Decision Delineations

Because a hospital delivering surgical treatment to patients is a complex system, operational analysis of the entire surgery delivery process is not realistic at this point. Researchers concentrate on certain resources and decision granularities in order to conduct meaningful experiments and improvements to isolated levels and parts of the process. There are many suggested ways of categorizing decision levels in surgery scheduling. Traditionally, operational researchers have classified three decision levels:

(i) case mix planning which optimizes the composition and volume of patients treated in a hospital (Hof et al., 2017),

(ii) master surgery scheduling in which the cyclic timetable is optimized (Cardoen et al., 2010) and

(iii) patient scheduling which deals with the detailed scheduling of individual patients. However, as pointed out in (Cardoen et al., 2010), these definitions are not consistent among researches, and they are not detailed enough to define the decisions that actually take place. They suggest a two-dimensional classification of planning and scheduling decisions where four different types of decisions (date, time, room and allocated capacity) may be taken on three different levels:

(i) discipline level where decisions are taken for medical specialties and departments,

(ii) surgeon level where decisions are made for surgeons and

(iii) patient level where decisions are made for patient types such as elective/ non-elective patients, inpatients/ outpatients or for certain reoccurring surgical procedures.

A taxonomic review of planning decisions in health care presented in (Hulshof et al., 2012) includes a taxonomy for surgical care services that separates a set of decision-making activities into the well-known hierarchical levels of strategic (long-term), tactical (medium-term) and operational (offline and online short-term) decisions. At a strategic level, decisions must be made about regional coverage (i.e. geographical distribution of facilities), service mix (services provided by a facility), case mix (composition and volume of patients), facility layout and capacity dimensioning of resources such as operating rooms, operating time capacity, presurgical rooms, recovery wards, ambulatory surgical ward, equipment and staff. Tactical level decisions include patient routing, capacity allocation (e.g. design of a master surgery schedule), admission control (rules regarding selection of patients from the waiting list) and staff-shift scheduling. Offline operational decisions include staff-to-shift assignments and surgical case scheduling; i.e. decisions resulting in schedules for staff, facilities and patients. During schedule execution, a set of online operational decisions must be made as reactions to deviations from the plan. This includes emergency case scheduling, surgical case rescheduling and staff rescheduling.

When the same problem is studied by several researches over time, it is typically given a name and a clear definition. Such problems exist within surgery scheduling literature as well. These problems represent a set of decision delineations and include the advance surgery scheduling problem (Magerlein and Martin, 1978), the allocation scheduling problem (Magerlein and Martin, 1978) and the surgery admission planning problem (Riise and Burke, 2011). We explain these briefly in the following overview.

The advance scheduling problem is the problem of assigning a date to patients from

the patient waiting list subject to capacity constraints. Surgeries are not allocated to specific operating rooms, but the total available operating room time limits how surgeries are assigned to dates.

The allocation scheduling problem is the problem of deciding the sequence of surgeries on a given day, assuming a set of available patients ready for surgery on that day. This involves operating room allocation and start time assignment.

The surgery admission planning problem or elective surgery scheduling problem (Marques et al., 2015) combines the advance scheduling problem and allocation scheduling problem so that dates, operating room allocations, sequence and starting times are designed simultaneously.

2.2.4 Uncertainty

Two sources of uncertainty that are sometimes addressed in surgery scheduling literature are arrival uncertainty and duration uncertainty. Arrival uncertainty refers to the late arrival of staff or patients as well as the random flow of non-elective patients. Duration uncertainty refers to the fact that activities performed before, during and after surgery may take longer or shorter time than expected. Especially surgery durations are critically variable. It is possible to estimate the duration of surgeries by analyzing data, and many papers focuse solely on such estimation (e.g. Gillespie et al. (2011); Joustra et al. (2013); Kayış et al. (2015)). As it turns out, accurate prediction is very difficult, partly because the scope of surgical procedures vary based on small physical differences among patients that cannot be observed prior to surgery. The lognormal distribution is often used to model how the duration of surgical procedures vary, although other distributions are also applicable, such as the normal distribution and the gamma distribution (Choi and Wilhelm, 2012). The lognormal distribution fits data where the natural logarithm of the mean and variance constitutes a normal distribution. Formally, if the random variable, Y, follows the normal distribution with mean λ and standard deviation, δ, then the random variable, $X = e^Y$, is lognormally distributed with the following mean, μ, and variance, σ^2 (Choi and Wilhelm, 2012):

$$\begin{aligned} E[X] = \mu = e^{\lambda + \frac{1}{2}\delta^2}, \\ V[X] = \sigma^2 = \mu^2(e^{\delta^2} - 1) \end{aligned} \tag{2.1}$$

The probability density function, providing the probability of X taking any particular value x, is usually stated as follows (Zydney et al., 1994):

$$p(x) = \frac{1}{\sigma x \sqrt{2\pi}} e^{-\frac{[ln(x) - ln(\mu)]}{2\sigma^2}} \tag{2.2}$$

Surgery duration uncertainty is a prominent difficulty because schedules that perform excellent on expectation may be poor if surgeries are delayed or finish early. Delay early in

the morning propagates throughout the day and may lead to overtime and cancelled surgeries. On the other hand, surgeries that finish early leave expensive resources idle until the next patient is ready for surgery. Consequences of variable surgery durations are usually mitigated by rules of thumb such as added slack between surgeries.

2.2.5 Research Methodology

Once a surgery scheduling problem is properly defined with decisions to be made and performance to be maximized, the next step is to analyze the problem and typically to perform some kind of combinatorial optimization. As mentioned in (Cardoen et al., 2010), several exact methods (e.g. mathematical programming such as linear programming, dynamic programming, column generation and branch-and-price) and heuristic methods (e.g. constructive heuristics and metaheuristics such as simulated annealing, genetic algorithm (GA) and tabu search (TS)) have been applied to variations of combinatorial surgery scheduling problems. While it is common to ignore uncertainty with deterministic approaches, Monte-Carlo simulation and discrete-event simulation have been applied in combination with optimization techniques to incorporate uncertainty in search for good solutions. Several researchers have performed scenario analysis to evaluate the effect of certain changes to specific problem settings. Surgery scheduling problems have also been defined as decision problems, and a few papers conduct benchmark studies to derive insights from real-life best practice. We elaborate on relevant previous work in Chapter 3.

2.2.6 Applicability of Research

As reported in (Cardoen et al., 2010), although surgery scheduling is substantially studied, reported results rarely extend beyond illustration of how objectives may be accomplished and indications of computational efficiency. From analyzing papers on the matter, it is not clear to which extent scheduling algorithms have been put to actual routinely use in real hospitals. According to (Cardoen et al., 2010), neither successful implementation nor managements reluctance to implementation have been satisfactory reported in literature. However, some excellent examples on successful implementations do exist, such as (Rath et al., 2017) reporting an average utilization increase of 3.5% for anesthesiologists, 3.8% for operating rooms, and operational cost savings of 7%, or an estimated annual cost saving of $2.2 million.

2.3 Conceptual Description of the Problem Under Investigation

Encouraged by the conclusive remarks presented in (Cardoen et al., 2010) calling for clearer definitions of studied surgery scheduling problems, we use the reviewed terminology to provide a more precise description of the investigated problem. A detailed

mathematical formulation is given in Chapter 4. The investigated variation of the surgery admission planning problem considers elective patients only, without any arrival uncertainty associated with non-elective patients or patient no-shows. We do not differentiate between outpatients and inpatients explicitly, as resources outside the operating room are not part of our problem. However, a subset of the considered surgical procedures are short outpatient procedures, while other procedures are longer and more variable. The performance measures are minimization of patient waiting time on the patient waiting list and maximization of operating room and surgeon utilization. The investigated problem is best described as a variation of the surgery admission planning problem where both operating rooms and surgeons are constraining resources. The considered source of uncertainty is surgery durations, and our research methodology is within the category of heuristic combinatorial optimization.

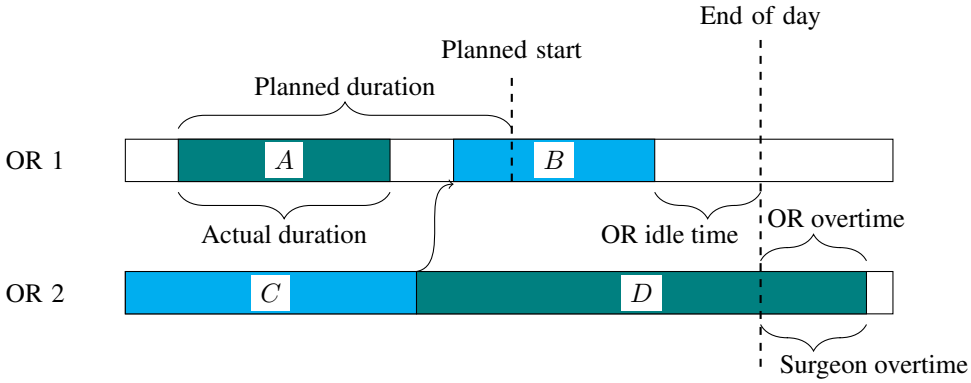


Figure 2.1: The daily problem.

Performance measures (except patient waiting time) and difference between planned duration and actual duration are illustrated for the daily problem in Figure 2.1. Surgery C and B are performed by the same surgeon, and the arrow drawn between the surgeries indicates that some surgeon idle time is incurred between surgery C and B. Surgery A lasts shorter than planned and incurs some operating room idle time between surgery A and B in operating room 1. Surgery B can start earlier than planned, and some unused operating room time is left at the end of the day. Surgery D incurs surgeon overtime and operating room overtime in operating room 2. The figure does not show that a set of surgeries are excluded from the schedule and that there is a waiting cost associated with postponing these surgeries.

2.4 Machine Scheduling Problems

In general, any situation in which a set of tasks must be assigned to a set of restricted resources that spend time accomplishing these tasks represents a scheduling problem (Abe-

dinnia et al., 2017). Such problems occur in many areas other than surgery scheduling, including project management, aerospace industry, computer science and personnel management. Arising from real problems in production industry, the tasks are often referred to as jobs and the restricted resources are called machines. A certain set of machine scheduling problems have been extensively studied within combinatorial optimization research. We review some of these in the following and include some theory on how genetic algorithms may successfully be applied to this category of optimization problems.

2.4.1 A Selection of Well-known Machine Scheduling Problems

We describe machine scheduling problems informally in the following based on a classification found in (Lenstra et al., 1977). In machine scheduling problems a set of jobs $j \in J$, each consisting of a set of sequenced operations $n \in O$, must be processed on a set of machines $m \in M$. Each operation must be processed without interruption (nonpreemption) on a given machine in M. The task is to find the optimal sequence of operations on each machine. For each job $j \in J$, the following data may be specified:

(i) the number of operations,
(ii) the machine order of the operations,
(iii) processing time for each operation,
(iv) operation weights,
(v) earliest possible starting times for the operations (release date or ready time),
(vi) due date or deadline and
(vii) a cost function describing the punishment for lateness.

The starting time and completion time for each operation may be derived from the decided sequence of operations in each machine, which in turn allows for computation of several objectives such as makespan, lateness or tardiness. Optimization of a single or multiple objectives may be subject to a variety of constraints, such as precedence constraints between jobs, no waiting constraints between operations within a job or hard deadline constraints for the jobs. We mention some of the most basic varieties of machine scheduling problems in the following.

The job shop scheduling problem (JSP) allows for any (and possibly different) number of operations per job and any machine sequence among the operations within a job.

The flexible job shop scheduling problem (fJSP) is a generalization of the job shop scheduling problem where the operations may be processed on any of the available machines (Pezzella et al., 2008).

The flow shop scheduling problem (FSP) assumes that all jobs include as many operations as there are machines, and that all operations in a job are executed on different machines following a machine sequence common to all jobs.

The permutation flow shop scheduling problem (pFSP) is a flow shop scheduling problem where every machine must process the jobs in the same order.

The open shop scheduling problem (OSP) takes no predefined sequence among the operations within jobs (Gonzalez and Sahni, 1976).

These problem variations may also be formulated with uncertainties in e.g. processing times or due-dates. The uncertainties are typically modelled as fuzzy sets resulting in problems such as parallel machine scheduling with fuzzy processing times (Peng and Liu, 2004). Another well-studied problem feature is sequence dependent setup times where each operation requires some setup activity that is dependent on the decided sequence (Ruiz and Maroto, 2006). Although specific machine scheduling problems may be solved efficiently, this genre of combinatorial optimization problems are generally hard to solve (Lenstra et al., 1977). Many problems are either proven to be NP-complete or very unlikely to be solvable with polynomial-bounded algorithms. Because of this, modern study on these problems typically focus on approximation algorithms or heuristics measured against benchmark solutions.

2.4.2 Genetic Algorithms for Solving Scheduling Problems

A popular metaheuristic for solving machine scheduling problems is the GA. To exemplify, the GA is the basis for solving
(i) JSP in (Gen et al., 1994),
(ii) fJSP in (Pezzella et al., 2008),
(iii) FSP in (Etiler et al., 2004),
(iv) pFSP in (Iyer and Saxena, 2004) and
(v) OSP in (Louis and Xu, 1996).
We provide a conceptual description of the GA metaheuristic in the following and highlight the necessary mechanisms for applying it to scheduling problems.

GAs work on an evolving population of solutions where selection, crossover and mutation is simulated over a set of generations until high-quality solutions emerge. Solutions are referred to as individuals and they possess genes that they pass on to their descendants. The genes are stored in the individuals chromosome. The chromosome must be decoded from coding space, in which mutation and crossover occurs, to solution space in order to perform evaluation and selection. In general, a representation may be characterized by how it enables a mapping from coding space to solution space (Cheng et al., 1996). Some representations guarantee feasibility, i.e. that the solutions produced by decoding are feasible with respect to constraints of the original problem. Other representations cannot guarantee feasibility, but they can guarantee legality, i.e. that the produced solution is a complete solution to the original problem. Furthermore, a mapping from coding space to solution space may be
(i) 1-to-1 so that a chromosome maps to a unique solution and vice versa,
(ii) 1-to-n so that a single chromosome maps to several different solutions or
(iii) n-to-1 so that several chromosomes maps to the same solution.
The representation of solutions in coding space, mutation and crossover operations are problem specific. We briefly discuss how a scheduling problem may be prepared for a GA in the following.

In (Cheng et al., 1996), nine different representations classified as either direct or indirect are described for the JSP. With a direct approach (e.g. operation-based, job-based, job pair relation-based, completion-based, random keys), chromosomes represent actual schedules

and explicitly state the schedule design. On the other hand, with an indirect approach (e.g. preference list-based, priority rule-based, disjunctive graph-based, machine-based), chromosomes represent a set of rules or prioritizations and some simple or complex heuristic is needed to decode the chromosome into an actual schedule. These decoders may be classified according to properties of the schedules they produce. A schedule may be categorized as semi-active or active (Sprecher et al., 1995). In a semi-active schedule, all tasks are scheduled as early as possible in a manner that adheres to the coded sequence. In an active schedule, the predefined sequence may be altered in order to fill gaps in the schedule. Once a representation is chosen, appropriate mutation operators and crossover operators bust be defined. Examples are found in (Cheng et al., 1996). The GA may then be applied to the scheduling problem by evolving a population of solutions until a stopping criterion, such as the maximum number of generations, is met.

2.5 Multiobjective Optimization

As already mentioned, multiple performance measures exist for both surgery scheduling problems and general scheduling problems. Sometimes, a single objective is appropriate and encompasses all desired qualities of the sought solution. However, decision makers often encounter situations where multiple qualities are desired simultaneously, and selecting only one performance measure will compromise the other measures. In the next subsections we explain ways of balancing different performance measures using multiobjective optimization. We start by looking closer at how multiobjective optimization is conceptually different from singleobjective optimization. A comprehensive review on multiobjective optimization may be found in (Deb, 2014).

2.5.1 Conflicting Objectives and Dominance

In singleobjective optimization there is one optimal solution with an associated optimal value. On the other hand, when there are multiple objectives there are generally three situations (Goh and Tan, 2009):

(i) total conflict, where the improvement of any objective inevitably leads to the deterioration of the others. In such a situation all solutions are optimal.

(ii) no conflict, where the objectives are correlated so that improvement of any objective leads to the improvement of the other objectives, in which case the problem may be treated as a singleobjective optimization problem with an arbitrarily selected objective.

(iii) partial conflict, where improvement can be pursued in all objectives simultaneously until improvement of one objective compromises at least one other objective.

Optimization with partially conflicting objectives requires the notion of dominance which we define in the following (Deb, 2014). From now on, when we use the term multiobjective optimization, we refer to optimization with partially conflicting objectives.

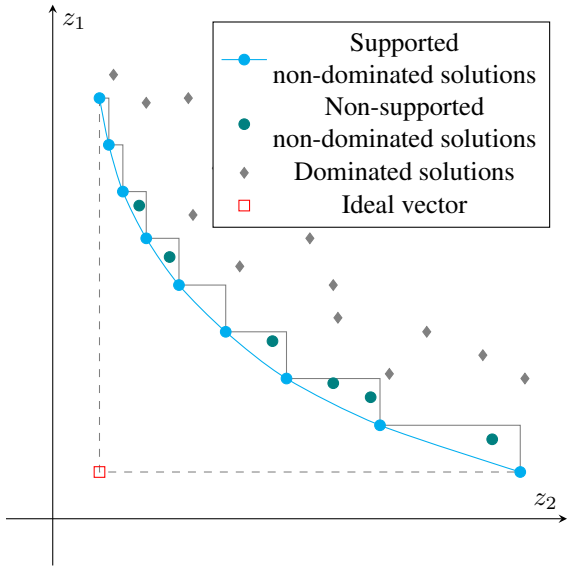


Figure 2.2: Solutions in multiobjective optimization.

Definition 2.5.1. Dominance. In an optimization problem with partially conflicting objective functions, a solution $\mathbf{x}^{(1)}$ dominates another solution $\mathbf{x}^{(2)}$ if both conditions 1 and 2 are met:

1. The solution $\mathbf{x}^{(1)}$ is no worse than $\mathbf{x}^{(2)}$ in all objectives.
2. The solution $\mathbf{x}^{(1)}$ is strictly better than $\mathbf{x}^{(2)}$ in at least one objective.

Equivalently, $\mathbf{x}^{(2)}$ is dominated by $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(1)}$ is non-dominated by $\mathbf{x}^{(2)}$. The following notation is standard for expressing that $\mathbf{x}^{(1)}$ dominates $\mathbf{x}^{(2)}$: $\mathbf{x}^{(1)} \preceq \mathbf{x}^{(2)}$.

2.5.2 The Goal of Multiobjective Optimization

Without knowledge of the users ideal composition of objective values, it is impossible to select a single optimal solution in a multiobjective problem. Therefore, the output of a multiobjective optimization problem must be a set of solutions representing different trade-offs between the objectives. Facing these trade-off solutions, the decision maker can make an informed choice among feasible solutions instead of deciding some general rule, prior to optimization, for how to favor the objectives in different situations. The optimization output should be a set of non-dominated solutions: the non-dominated set, defined in the following (Deb, 2014).

Definition 2.5.2. Non-dominated set. The non-dominated set P' are those solutions among all solutions in the search space P that are not dominated by any other solution in

P. The non-dominated set is also referred to as the Pareto-optimal set, the Pareto-optimal front or the non-dominated front.

The non-dominated set is illustrated in Figure 2.2. The goal of multiobjective optimization is to find the entire non-dominated set. As seen in the figure, non-dominated solutions can be either *supported* or *non-supported*. Supported non-dominated solutions lie on convex parts of the Pareto front, and the non-supported non-dominated solutions lie on non-convex parts of the Pareto front. We discuss different approaches for approximating the set of non-dominated solutions in the following subsection.

2.5.3 Traditional Approaches to Multiobjective Optimization

Two basic approaches to multiobjective optimization mentioned in (Deb, 2014) are the weighted sum method and the ϵ-constrained method. These methods utilize information from the user to convert multiobjective problems into singleobjective problems. The weighted sum method optimizes a linear combination of the objectives in the following manner:

$$min/max \quad F(x) = \sum_{m=1}^{M} w_m f_m(x) \tag{2.3}$$

where there are $m \in M$ objectives, $f_m(x)$ is the objective value of the m^{th} objective and w_m is the user defined weight given to objective m. Typically, the weights are normalized so that $w_m \in [0, 1]$ and $\sum_{m=1}^{M} w_m = 1$. The weighted sum method may be used to approximate the Pareto-front by strategically varying the weights over multiple optimization runs. However, there are two major problems with this approach:

(i) there is no straightforward strategy for varying the weight vectors to ensure a uniform spread of Pareto-optimal solutions and

(ii) solutions on the non-convex path of the Pareto-front (i.e. non-supported non-dominated solutions) will never be optimal with a weighted sum strategy.

The ϵ-constrained method can find solutions on the non-convex part of the Pareto-front (non-supported non-dominated solutions). The user defines lower bounds (i.e. ϵ-values) on the performance of all objectives but one. The result is a singleobjective optimization problem with one extra constraint for each non-included objective. Mathematically we have:

$$\begin{aligned} &\text{min/ max } F(x) = f_\mu(x) \\ &\text{subject to } f_m(x) \leqslant \epsilon_m, \quad \forall m \in M, m \neq \mu \end{aligned} \tag{2.4}$$

where there are $m \in M$ objectives, $f_m(x)$ is the objective value of the $\forall m \in M, m \neq \mu$ objectives, ϵ_m are the lower bounds on these objectives and μ is the selected objective with value of $f_\mu(x)$ to be maximized or minimized. The ϵ-constrained method may also be used repeatedly to approximate the Pareto-front, but the configuration of ϵ-values is

nontrivial. Another general drawback of the method is that it requires a lot of information from the user, and if the ϵ-values are set inappropriately, large parts of the solution space (or none of it) can be omitted.

Another classical approach that converts a multiobjective problem into a singleobjective problem is the weighted distance method (Marques et al., 2015). In this method, the objective is to minimize the weighted distance between a found solution, z, with a reference solution, z^*, that is at least as good as the ideal solution (illustrated in Figure 2.2). Varying the weights over consecutive optimization runs can yield all non-dominated solutions.

2.5.4 Multiobjective Evolutionary Algorithms (MOEAs)

Because evolutionary algorithms work on a population of solutions they have proven exceptionally well suited for finding multiple Pareto-optimal solutions within a single optimization run. As opposed to standard GAs that select individuals based on their fitness value, MOEAs use the concept of dominance to control the evolution of the population. Several variations of MOEAs have bee developed in the last decades, including SPEA (Zitzler and Thiele, 1999) and its successor SPEA2 (Zitzler et al., 2001), NSGA (Srinivas and Deb, 1994) and its successor NSGA-II (Deb et al., 2002), Pareto envelope-based selection algorithm (PESA) (Corne et al., 2000), ϵ-multiobjective-evolutionary-algorithm (ϵ-MOEA) (Deb et al., 2005) and multiobjective evolutionary algorithm based on decomposition (MOEA/D) (Chen et al., 2009). We outline the details of the two MOEAs we compare in this work, namely the SPEA2 and NSGA-II, in Chapter 5.

2.5.5 Measuring Performance of MOEAs

Assessing the performance of MOEAs is not straightforward because of two reasons:
(i) a multiobjective optimization problem is in itself a multiobjective optimization problem and
(ii) the true Pareto-front is not always known.
Several performance measures exist, and not all of them require a known Pareto front. We explain a selection of well-known performance measures mentioned in (Ripon et al., 2007) and (Yen and He, 2014) in the following. We denote the approximated front found by an algorithm as X, and let $\overline{X}$ denote the solutions in X that are non-dominated by all other solutions in X. The true Pareto front will be denoted as PF_{true}.

Convergence towards the true Pareto-front

For measuring convergence towards the true Pareto-front, PF_{true}, there are essentially two classes of measures:
(i) the number of discovered Pareto-optimal solutions and
(ii) the distance between found solutions and the true Pareto-front.

The *ratio of nondominated individuals* (RNI), *error ratio* (ER), *overall nondominated vector generation* (ONVG) and *Pareto dominance indicator* belong to the first class. Measures assessing distance to the true Pareto front include *generational distance* (GD), *inverted generational distance* (IGD) and *maximum Pareto-front error* (MPFE). We provide a description of these measures in the following and indicate whether or not the true Pareto-front must be provided in order to compute each measure.

The ratio of nondominated individuals (RNI) measures the ratio of nondominated individuals, $\overline{X}$, in the approximation front, X:

$$RNI = \frac{|\overline{X}|}{|X|} \tag{2.5}$$

A higher value of RNI is better, with $RNI \in [0, 1]$, and $RNI = 1$ indicating that all individuals in the approximation front are nondominated. The RNI measure does not require the true Pareto-front.

The error ratio (ER) is the ratio of nontrue Pareto points in the approximation front, X, where $e(x_i) = 0$ if $x_i \in PF_{true}$; and $e(x_i) = 1$ otherwise:

$$ER = \frac{\sum_{i=1}^{|X|} e(x_i)}{|X|} \tag{2.6}$$

Preferably, there are no nontrue Pareto points in the approximation front, and ER-values closer to 0 are better. The true Pareto front, PF_{true}, must be known during calculation of the ER measure.

The overall nondominated vector generation (ONVG) measures the total number of nondominated individuals in the approximation front, $\overline{X}$, where a large value is usually desired:

$$ONVG = |\overline{X}| \tag{2.7}$$

The ONVG measure does not require the true Pareto front.

The Pareto dominance indicator compares the approximation front of a given algorithm, e.g. the front X_1 found by algorithm A_1, to all approximation fronts, $X_1, X_2, ..., X_p$, found by p different algorithms. The assessment involves finding the ratio of solutions in X_1 that are not dominated by any solution in the combined solutions found by all algorithms:

$$NR(X_1, X_2, ..., X_p) = \frac{|X_1 \cap Y|}{|Y|} \tag{2.8}$$

where Y is the nondominated set of all approximation fronts combined: $Y = \{y_i | \forall y_i, \nexists x_j \in (X_1 \cup X_2 \cup \cdots \cup X_p) \prec y_i\}$. A value of $NR(X_1, X_2, ..., X_p) = 1$ indicates that X_1 includes all non dominated solutions found overall. As this measure compares the relative performance of two or more MOEAs, the true Pareto front is not needed.

The generational distance (GD) is a measure of how close the approximated Pareto front, X, is to the true Pareto front, PF_{true}:

$$GD = \frac{\sqrt{\sum_{i=1}^{|X|} d_i^2}}{|X|} \tag{2.9}$$

where d_i denotes the distance, in objective space, between an individual, $x_i \in X$, to the closest member, $x_j \in PF_{true}$, of the true Pareto front: $d_i = \min_j ||f(x_i) - PF_{true}(x_j)||$. A smaller value of GD implies a more accurate approximation of the true Pareto front, with $GD = 0$ indicating that all approximated solutions are actual Pareto optimal solutions. The GD measure cannot be computed without the true Pareto front.

The inverted generational distance (IGD) measures the distance, in objective space, from the points in the true Pareto front, PF_{true}[2], to the closest point in the approximation front, X:

$$IGD = \frac{\sum_{v \in PF_{true}} d(v, X)}{|PF_{true}|} \tag{2.10}$$

where $d(v, X)$ denotes the smallest distance between solution $v \in PF_{true}$ and a solution $x_i \in X$. While similar to the GD measure, the IGD measure includes information about the diversity of solutions in the approximation front, X. To illustrate, an approximation front consisting of a single solution that belongs on the true Pareto front will have $GD = 0$, but since this single point is distant from many other points on the true Pareto front, the IGD measure will be high. A value of $IGD = 0$ implies that all true Pareto optimal solutions are found, but not that all solutions in the approximation front are true Pareto optimal solutions. The IGD measure requires the true Pareto front.

The maximum Pareto-front error (MPFE) is the longest distance, in objective space, between a solution in the true Pareto front, PF_{true}, and its closest solution in the approximation front, X:

$$MPFE = \max_i d_i \tag{2.11}$$

with d_i as previously defined. A minimal value of MPFE is preferred. The measure requires the true Pareto front.

Distribution

A decision maker is generally not interested in an overflow of solutions that are almost identical. A desired quality of MOEAs is therefore to output a well-distributed set of solutions that are interestingly different from each other. Three measures attempt to quantify such a quality, namely *uniform distribution* (UD), *spacing* (S) and *number of distinct choices* (NDC). We describe these measures in the following.

[2] the points on PF_{true} must be uniformly distributed.

The uniform distribution (UD) measures the distribution of nondominated individuals, $\overline{X}$, on the approximation front, X. It uses the concept of *niche count*, $nc(\overline{x}_i)$ of an individual, x_i, which, informally, is the number of solutions among $\overline{X}$ with a distance lower than σ_{share} from x_i. Mathematically, we have:

$$nc(\overline{x}_i) = \sum_{j=1, j\neq i}^{|\overline{X}|} Sh(x_i, x_j),$$

$$Sh(x_i, x_j) = \begin{cases} 1 & \text{if } d(x_i, x_j) < \sigma_{share} \\ 0 & \text{otherwise} \end{cases} \quad (2.12)$$

The uniform distribution metric, UD, is expressed using the standard deviation of the niche counts of all individuals in $\overline{X}$, S_{nc}, with $\overline{nc}(\overline{X})$ denoting the average niche count:

$$UD = \frac{1}{1 + S_{nc}},$$

$$S_{nc} = \sqrt{\frac{\sum_{i=1}^{|\overline{X}|}(nc(\overline{x}_i) - nc(\overline{X}))^2}{|\overline{X}| - 1}} \quad (2.13)$$

The UD metric assesses whether the solutions in $\overline{X}$ are equally unique, i.e. that the solutions have equally many neighbouring solutions. In the utopian case, all solutions have the same niche count, leading to a standard deviation, $S_{nc} = 0$ and $UD = 1$ which is the desired upper bound of the UD metric. This metric does not require the true Pareto front.

The spacing (S) metric evaluates the distances from points in the approximated Pareto front to the closest points in the true Pareto front using d_i as already defined and the average of all distances, $\overline{d}$, as an assessment for how evenly solutions are spread along the known Pareto front:

$$S = \sqrt{\frac{1}{|\overline{X}|} \sum_{i=1}^{|\overline{X}|} (d_i - \overline{d})^2} \quad (2.14)$$

The true Pareto front is needed to compute the distances in the spacing metric.

The number of distinct choices (NDC) counts only solutions in the approximated Pareto front, X, that are sufficiently different from each other in the objective space. A user defined parameter $\mu \in < 0, 1 >$ is used to divide the M-dimensional solution space into $1/\mu^m$ grids, where M is the number of objectives. Two solutions within the same grid, $T_\mu(q)$, counts as one solution, as they represent practically equally valuable solutions to the decision maker. The measure is expressed mathematically (Wu

and Azarm, 2001):

$$NDC_{\mu}(X) = \sum_{l_m}^{(1/\mu)-1} \cdots \sum_{l_2}^{(1/\mu)-1} \sum_{l_1}^{(1/\mu)-1} NT_{\mu}(q, X),$$

$$NT_{\mu}(q, X) = \begin{cases} 1, & \exists x_i \in X, x_i \in T_{\mu}(q) \\ 0, & \forall x_i \in X, x_i \notin T_{\mu}(q) \end{cases},$$

$$q = (q_1, q_2, ..., q_m), \text{with } q = l_i/(1/\mu) \quad (2.15)$$

Generally, a high number of distinct choices is preferred. The NDC measure does not require the true Pareto front.

Maximum spread

Even if an outputted set of solutions are well distributed close to the true Pareto-front, essential parts of the front may not be covered. The *maximum spread* (MS) metric assesses how well the found solutions cover the true Pareto-front by measuring against the extreme points in the true Pareto Front. Let $PF_{true,m}^{max}$ and $PF_{true,m}^{min}$ denote the maximum and minimum value of objective $m \in M$ in the true Pareto front, respectively. Similarly, let X_m^{max} and X_m^{min}, denote the maximum and minimum values of objective $m \in M$ in the approximated Pareto front. The maximum spread (MS) can be computed as follows:

$$MS = \sqrt{\frac{1}{M} \sum_{m=1}^{M} \left[\frac{min\{X_m^{max}, PF_{true,m}^{max}\} - max\{X_m^{min}, PF_{true,m}^{min}\}}{PF_{true,m}^{max} - PF_{true,m}^{min}} \right]^2} \quad (2.16)$$

A MS-value closer to 1 is preferred, as this indicates a good spread in all objectives. The measure does not require the entire Pareto front, but the extreme points must be known.

Hypervolume

A measure that captures both the closeness to the true Pareto-front and the diversity among found solutions is the *hypervolume* metric (Ripon et al., 2007). From a reference point, W, in objective space, a hypervolume, v_i is constructed with every solution, x_i in the approximated Pareto-front, X, so that the reference point and solution are diagonal corners in the hypercubes. The hypervolume measure, is the volume of the union of all hypervolumes:

$$H = volume\left(\cup_{i=1}^{|X|} v_i \right) \quad (2.17)$$

This measure does not require the true Pareto front.

Set coverage

A useful metric for comparing two algorithms is the *set coverage* metric which essentially computes the ratio of solutions in one solution set that weakly dominates solutions in the other set. Consider two approximations, X_1 and X_2, to the same true Pareto front produced by two different algorithms. The set coverage metric (C) can be computed:

$$C(X_1, X_2) := \frac{|x_2 \in X_2; \exists x_1 \in X_1 : x_1 \preceq x_2|}{|X_2|} \tag{2.18}$$

If $C(X_1, X_2) = 1$, all solutions in X_2 are covered by X_1 as there are, for every solution $x_2 \in X_2$, a better or equally good solution $x_1 \in X_1$. The measure is not symmetric, so $C(X_2, X_1)$ will provide more information and should generally accompany the calculation of $C(X_1, X_2)$.

2.6 The Wilcoxon Signed-Rank Test

Because evolutionary algorithms rely on stochastic processes to guide random emergence of high quality solutions, a single computation run is not enough to express the applicability of a particular evolutionary algorithm on a given problem instance. The achieved solution quality may vary substantially from one optimization run to the next. Therefore, quality assessment of evolutionary algorithms must be accompanied by some statistical analysis based on several computation runs. A popular and simple nonparametric[3]statistical test that can be used to compare the performance of two different population based algorithms, is the Wilcoxon signed-rank test (Derrac et al., 2011). Consider two algorithms, A_1 and A_2, yielding N results, a_n, over $n \in N$ optimization runs on the same problem. The null hypothesis, H_0, is that the results from A_1: $\{a_1^1, a_2^1, \cdots, a_N^1\}$ and A_2: $\{a_1^2, a_2^2, \cdots, a_N^2\}$ are samples from the same distribution. The null hypothesis is investigated by pairwise comparison of the results, a_1^1 vs. a_1^2, a_2^1 vs. a_2^2 and so forth. The difference in performance, $diff_n$, is calculated for all pairs and ranked from largest to smallest according to absolute value in a sorted list. The largest difference is given the largest rank of N, the second largest is given the rank $N-1$ and so forth. Now, the sum of ranks for cases where one algorithm outperforms the other can be expressed with the following equations:

$$\begin{aligned} R^+ &= \sum_{diff_n>0}^{N} rank(n) + \sum_{diff_n=0}^{N} \frac{1}{2} rank(n) \\ R^- &= \sum_{diff_n<0}^{N} rank(n) + \sum_{diff_n=0}^{N} \frac{1}{2} rank(n) \end{aligned} \tag{2.19}$$

In these equations, R^+ represents the sum of ranks for all cases where one algorithm outperformes the other, and R^- denotes the opposite. For cases where performance is equal, the rank values are split (alternative strategies for ties exist). The smallest of the sums,

R^+ and R^-, is used in combination with the Wilcoxon distribution to either reject or not reject the null hypothesis. A rejection means that one algorithm significantly outperforms the other.

2.7 Optimization under Uncertainty

Uncertainty is omnipresent is the real world. Any otherwise accurately modelled real problem solved to optimality will often times be suboptimal in practice due to unforeseen changes in parameters, the environment or the problem structure. We begin this section by describing the different sources and categories of uncertainty in relation to general optimization problems. Then, we turn to how uncertainty may be incorporated in MOEAs.

2.7.1 Notions of Uncertainty in the Optimization Process

The first step when hedging for uncertainty during optimization is to identify the nature of the uncertainties and how they affect the solution developed in the optimization process. Consider, without loss of generality, a minimization problem (Shapiro, 2008):

$$\begin{aligned} &\text{minimize } f(x) \\ &\text{subject to } \mathbf{G}(\mathbf{x}) \leqslant 0 \\ &\forall x \in \mathbb{R}^n \end{aligned} \tag{2.20}$$

where $f(\mathbf{x})$ is a composite of costs representing the objective and $\mathbf{G}(\mathbf{x})$ are the constraints restricting the design variables $\mathbf{x}$. The objective function and constraints depend on parameters, $\xi \in \mathbb{R}^d$, which can be erroneous (e.g. rounding errors), noisy (e.g. sensory input) or *uncertain* (e.g. as in our case, uncertain processing times). When decisions are made under uncertainty, and then, after the resolution of the uncertain parameters, ξ, no mitigating or improving action is allowed, the optimization task is to design a solution, $\mathbf{x}$, that is promising with regards to solution quality, $f(\mathbf{x}, \xi)$, and solution feasibility, $\mathbf{G}(\mathbf{x}, \xi) \leqslant 0$, typically assuming some distribution for the uncertain parameters.

Another interesting situation occurs when the decision maker has the opportunity to react to the resolution of uncertainties after some permanent decisions have been made in the first stage. This kind of problem is often modelled mathematically as a two-stage stochastic program, and the goal is to find good solutions to the unchangeable first stage decision variables that puts the decision maker in a relatively good second-stage situation for all outcomes of the uncertain variables. In some problems, this structure is extended through multiple stages of uncertainty resolution and new decisions in which case the problem is often modelled as a multi-stage stochastic program.

[3] nonparameteric tests are necessary as conditions for parametric tests are often not met; e.g there is lack of independence and normality (Tan et al., 2002).

2.7.2 Classical Approaches to Optimization Under Uncertainty

Several techniques may be applied to hedge for uncertainties while searching for optimal solutions (Shapiro, 2008). The simplest approach is to assume expected values for all uncertain parameters and solve the problem as a deterministic problem. In a generally superior (but far more comprehensive) approach a finite number of scenarios are generated and the stochastic problem may be formulated as a (very large) deterministic problem in which all scenarios are tested. Such a problem may be solved using decomposition techniques such as the well known L-shaped method (Laporte and Louveaux, 1993)[4]. Another approach is called robust optimization, which in turn has several different notions in literature, but generally targets solutions that perform well even in the worst case, e.g. by assuming 95$^{\text{th}}$ percentiles of uncertain parameters during optimization. Chance programming is another technique in which the search targets solutions where the constraints hold with a given probability. Another comprehensive approach is to combine optimization and simulation in a common framework (Gosavi, 2003) e.g. using metaheuristic as we do in this thesis, or other techniques such as neural networks.

2.7.3 Evolutionary Optimization Under Uncertainty

Especially one mechanism in evolutionary algorithms is affected by the presence of uncertainty, namely the fitness evaluation. Under uncertainty, it is not straightforward to assess the quality or even the feasibility of a solution, and the winner among individuals is dependent on the scenario. Four categories of uncertainty in evolutionary optimization are discussed in (Jin and Branke, 2005), namely

(i) noise,
(ii) robustness,
(iv) time-varying fitness functions and
(iii) fitness approximation.

We describe these in the following overview.

Noise. If the fitness function is noisy, the same solution will be evaluated differently from one time to another. Mathematically we have fitness function (Jin and Branke, 2005): $F(x) = \int_{-\infty}^{\infty}(f(x) + \delta)p(\delta)d\delta$ where δ represents the noise added to the original objective function f. The classical approach to handling noisy fitness functions is to take the average of a finite number of fitness evaluations: $F^*(x) = \frac{1}{N}\sum_{i=1}^{N}(f(x) + \delta_i)$.

Robustness. Uncertainty is resolved after optimization and the found solution must be *robust*, i.e. well prepared for variability in design variables or environmental variables. The *effective fitness function* is mathematically stated in (Jin and Branke, 2005) for the case when the design variables are uncertain: $F(x) = \int_{-\infty}^{\infty} f(x+\delta)p(\delta)d\delta$. The

[4] when two-stage stochastic problem has complete recourse.

effective fitness function represents the expected fitness and is usually approximated by Monte Carlo integration: $F^*(x) = \frac{1}{N}\sum_{i=1}^{N} f(x + \delta_i)$.

Time varying fitness function. The fitness function is deterministic but varies over time because the environment is dynamic (Jin and Branke, 2005): $F(x) = f_t(x)$. The task is to develop solutions that may be adapted to changing environments without extensive computation.

Fitness approximation. In cases where each fitness evaluation is very computationally expensive, it may be advantageous to use an approximation during the search process (Jin and Branke, 2005). In this case, the approximation error cannot be averaged out as with a noisy fitness function, because the approximation error is deterministic. Most approaches therefore combine an approximation of the fitness with occasional computation of the actual fitness.

2.7.4 MOEAs in Uncertain Environments

Solving multiobjective problems under uncertainty is complicated because the dominance relations among individuals are uncertain. In (Goh and Tan, 2009), the performance of MOEAs are studied in three different situations:

(i) in the presence of noise,

(ii) in dynamic environments and

(iii) in situations where solutions must be robust against changes taking place after optimization.

MOEAs in the Precence of Noise

In the case of noisy fitness functions, the performance of classical MOEAs varies from one problem to another (Goh and Tan, 2009). For increasing levels of noise, the evolutionary optimization process degenerates to random search. This implies that significant levels of noise affects the MOEAs ability to adequately evaluate and select solutions in the evolutionary process. Several specializations of MOEAs have been developed to handle noise, including estimate strength Pareto evolutionary algorithm (ESPEA), multiobjective probabilistic selection evolutionary algorithm (MOPSEA), noise tolerant strength pareto evolutionary algorithm (NTSPEA) and modified non-dominated sorting genetic algorithm II (MNSGA-II). These algorithms improve the selection, elitism and diversity preservation schemes to handle the presence of noise. Examples include explicit averaging in the MNSGA-II, probabilistic selection in ESPEA and a finitie life time assigned to archive individuals in NTSPEA.

MOEAs in Dynamic Environments

Changes over time in dynamic environments can affect both previously found solutions and the true Pareto-front. Specialized MOEAs mentioned in (Goh and Tan, 2009) are directional-based dynamic evolutionary multi-objective optimization algorithm (DB-DEMOA) and dynamic non-dominated sorting genetic algorithm II (DNSGA-II). These algorithms detect changes and immediately react by fitness re-evaluations or some form of local search to track the time-varying Pareto-front. Another key part of such MOEAs is additional mechanisms for preserving or recreating diversity.

Robust Optimization with MOEAs

Robust optimization may be considered as a special case of dynamic optimization where re-optimization after shifts in solution space and decision space is not possible (Goh and Tan, 2009). Preparing for flexible re-optimization is therefore not a consideration. The goal in robust multiobjective optimization is typically to

(i) optimize the expected fitness in which case the desired output is referred to as the effective Pareto-front, $PF^A_{eff,\delta}$, or

(ii) optimize based on the worst case. A general mutiobjective minimization problem with M objectives and uncertainties δ_x associated with decision variables and δ_e associated with environmental variables may be formulated as follows (Goh and Tan, 2009):

$$\min \mathbf{f}(\mathbf{x}, \delta_\mathbf{x}, \delta_\mathbf{e}) = \left\{ f_1(\mathbf{x}, \delta_\mathbf{x}, \delta_\mathbf{e}), f_2(\mathbf{x}, \delta_\mathbf{x}, \delta_\mathbf{e}), ..., f_M(\mathbf{x}, \delta_\mathbf{x}, \delta_\mathbf{e}) \right\}$$
$$\text{subject to } \mathbf{g}(\mathbf{x}, \delta_\mathbf{x}, \delta_\mathbf{e}) \geqslant 0, \mathbf{h}(\mathbf{x}, \delta_\mathbf{x}, \delta_\mathbf{e}) = 0 \tag{2.21}$$

The effective Pareto front $PF^A_{eff,\delta}$ consists of all non-dominated solutions when the fitness value is calculated using the effective fitness function, which usually is approximated with Monte-Carlo integration. This straightforward approach is referred to as explicit averaging (Jin and Branke, 2005). Another simple approach mentioned in (Jin and Branke, 2005), called implicit averaging, exposes all individuals in the same generation to a single realization of the uncertain parameters and perturbs the design variables randomly to avoid favouring solutions that are fit by chance. If the population size is very large, this leads to correct convergence of the applied algorithm. A set of specialized MOEAs have been developed to produce robust solutions without performing computationally expensive Monte-Carlo integration or increasing the population size. Methods mentioned in (Goh and Tan, 2009) include robust multi-objective optimization evolutionary algorithm (RMOEA) and SPEA2 and NSGA-II with an incorporated fitness inheritance scheme.

Chapter 3

Literature Review

In this chapter we consider how surgery scheduling problems have been dealt with in computational research. We focus on papers that attempt to solve similar problems as the problem analyzed here. Essentially, this means that we highlight work dealing with one or several of the following aspects:

(i) scheduling elective patients,
(ii) multiobjective surgery scheduling,
(iii) detailed surgery scheduling over several days,
(iv) simultaneous scheduling of staff and operating rooms,
(v) heuristic approach to surgery scheduling problems and
(vi) surgery duration uncertainty.

3.1 Heuristic Approaches to Surgery Scheduling

As surgery scheduling problems are closely related to machine scheduling problems, which are generally very hard to solve, researchers often acknowledge that realistically sized surgery scheduling problems cannot be solved exactly in a computationally tractable manner. In (Batun et al., 2011), they show that their daily stochastic surgery scheduling problem with multiple operating rooms is NP-hard by reducing the bin-packing problem to a variation of their problem. The advance scheduling problem and the allocation scheduling problem are both argued to be NP-hard in (Hans et al., 2008) and (Cardoen et al., 2009), respectively. The combination of the two problems is an even harder problem, as argued by (Riise and Burke, 2011) and proved for a variation of the problem in (Aringhieri et al., 2015) and because of this, the problem is typically either decomposed or solved heuristically. We look closer at some heuristic approaches to surgery scheduling in the following.

3.1.1 Rule-based Heuristics

Simple rules for scheduling surgeries under duration uncertainty are compared against the NSGA-II in (Gul et al., 2011). After analyzing data to fit lognormal distributions to surgery durations, the following heuristics are used to determine the sequence of surgeries at a day in an outpatient clinic: longest processing time first (LPT), shortest processing time first (SPT), increasing variance and increasing coefficient of variation. As it turns out, the SPT sequencing rule yields solutions of comparable quality as the NSGA-II. The bicriteria heuristic used for solving the surgery admission planning problem in (Marques et al., 2015) is composed of a constructive phase and an improvement phase. Both phases are essentially a set of rules for designing (constructive) and improving (improvement) schedules to obtain good solutions. Because very little computation is necessary for such a rule-based approach, runtime is negligible and it is acceptable to run the algorithm several times in a bicriteria framework. Their algorithm performs quite well, however, under the assumption of deterministic surgery durations.

3.1.2 Metaheuristics

In (Mateus et al., 2017), two constructive heuristics developed for two variations of the surgical case assignment problem are combined with an improvement heuristic based on iterative neighbourhood search. A variable neighbourhood decent (VND) local search heuristic is developed in (Riise and Burke, 2011) for the surgery admission planning problem. Variable neighbourhood search is also the basis for solving the surgery tactical planning problem in (Dellaert and Jeunet, 2017). Adaptions of the TS, simulated annealing (SA) and multi-start method have also been implemented to solve variations of the advance surgery scheduling problem (Molina-Pariente et al., 2015b). However, none of the above algorithms are implemented to properly handle multiple objectives.

3.1.3 Multiobjective Heuristics

Several papers consider the surgery scheduling problem in a multiobjective context. While most papers use the simple weighted sum approach (e.g. Batun et al. (2011); Riise and Burke (2011); Zhou et al. (2016); Jebali and Diabat (2015); Mancilla and Storer (2012, 2013)), several recent examples of more advanced approaches exist such as multiobjective rule-based heuristics in (Marques et al., 2015; Gul et al., 2011) and population based multiobjective metaheuristics, e.g. the NSGA-II in (Guido and Conforti, 2017; Gul et al., 2011), a hybrid MOEA/D in (Zhang et al., 2017) and a modified ant colony optimization (ACO) algorithm in (Xiang, 2017). We elaborate on some interesting details in the following.

The bicriteria heuristic developed in (Marques et al., 2015) uses the weighted distance method minimizing a weighted Chebyshev distance to a reference point to simultaneously maximize surgical suite occupation and the number of surgeries scheduled. In (Gul et al., 2011), the NSGA-II is used to minimize expected overtime and expected waiting time. The

NSGA-II in (Guido and Conforti, 2017) uses an initial constructive phase that combines a set of feasible individuals with a set of semi-feasible individuals to improve convergence of the algorithm. Four objectives are optimized: minimization of operating room idle time, surgeon idle time, surgeon overtime and leveling of operating room time among surgery groups. In (Xiang, 2017) the load balance equilibrium objective is incorporated in an ACO implementation. Three different settings of ACO are compared for solving a daily multiobjective operating room scheduling problem:

(i) a singleobjective ACO minimizing makespan,
(ii) an ACO using the weighted sum method with equal weights on the objectives and
(iii) a hybrid Pareto-set ACO with multiple objectives.

They conclude that their hybrid Pareto-set ACO with multiple objectives is superior to the other algorithms and that it finds solutions with equal makespan as the singlobjective ACO focusing *only* on makespan but with better results on the other objectives. However, none of the mentioned papers on multiobjective surgery scheduling consider uncertain surgery durations.

3.2 Multi-stage and Multi-resource Problems

Compatible operating room schedules and surgeon schedules are created in (Riise and Burke, 2011). In (Molina-Pariente et al., 2015a) operating rooms and surgical teams are simultaneously considered, with surgery durations depending on the specific teams. In (Fei et al., 2010) the advance scheduling problem is solved with a column generation based heuristic and the allocation scheduling problem is solved with a hybrid GA. In addition to operating rooms, the recovery rooms are considered as a restricting resource. Another two-level approach for this problem is presented in (Aringhieri et al., 2015) with a TS procedure that iterates between the advance scheduling decision level and the allocation scheduling level. Post-surgery beds are also included in their model.

A daily surgery scheduling problem is modelled as a multi-resource constraint fJSP and solved with an ant ACO algorithm in (Xiang et al., 2015). The considered resources are operating rooms, surgeons, nurses, anesthesiologists and beds in the post anesthesia care unit. The same problem is considered in (Xiang, 2017) in a multiobjective setting. In (Zhou et al., 2016) Porter teams are considered in the perioperative stage together with operating rooms, surgeons, anesthesiologists and nurses in the perioperative stage as well as general care unit and intensive care unit in the postoperative stage. They formulate a multi-day scheduling problem as a complete three stages hybrid FSP with multiple resources and solve it with a new Lagrangian relaxation algorithm. The patient flow through the medium care unit, operating room and intensive care unit is considered together with the number of nursing hours in (Dellaert and Jeunet, 2017). In addition to operating rooms, particular focus is given to anesthesiologists in (Rath et al., 2017), and nurses in (Guo et al., 2016).

3.3 Surgery Scheduling under Duration Uncertainty

Surgery scheduling under duration uncertainty is researched with a variety of methodologies and we review some examples in the following. A majority of the reviewed papers assume that surgery durations follow the lognormal distribution (Addis et al., 2014).

3.3.1 Stochastic Programs

Detailed surgery scheduling problems are solved to optimality using stochastic programming by e.g. (Mancilla and Storer, 2012, 2013; Batun et al., 2011). In (Mancilla and Storer, 2013), an exact decomposition algorithm is used to solve small surgery scheduling instances where a single surgeon works in parallel operating rooms. In (Batun et al., 2011), the L-shaped method, modified with a set of valid inequalities, is used to solve daily surgery scheduling instances with up to 11 surgeries, 3 surgeons and 6 operating rooms. The L-shaped method is also the basis for appointment scheduling formulated as a stochastic linear model in (Denton and Gupta, 2003), where the method is used with sequential bounding to obtain optimal or close to optimal solutions.

A heuristic approach to a two-stage stochastic programming model where rescheduling may be done as a response to variances in surgery durations and arrival of emergency patients is developed in (Bruni et al., 2015). The heuristics are tailored to the advance scheduling problem to create feasible initial schedules, evaluate recourse strategies and improve initial schedules with a rolling horizon algorithm. For a similar problem, where the duration variability of both the operating room and intensive care unit is considered, sample average approximation is used to solve a two-stage stochastic program with recourse in (Jebali and Diabat, 2015). In (Denton et al., 2007), a two-stage stochastic recourse model is solved to optimality and it is shown that a simple heuristic where surgeries are sequenced according to increasing variance performs quite well. A two-stage, mixed-integer stochastic dynamic programming model with recourse is implemented for a detailed daily surgery scheduling problem in (Rath et al., 2017). In (Min and Yih, 2010), a sample average approximation approach to the advance scheduling problem under duration uncertainty is compared to a deterministic model using expected values for surgery durations. Simulation results show a substantial expected cost reduction of incorporating uncertainty. An exact and a heuristic approach to a detailed daily surgery scheduling problem under duration uncertainty are implemented in (Pulido et al., 2014). The deterministic problem can be solved to optimality, but for the stochastic case, a decomposition approach using a constructive and an improvement heuristic is superior in terms of speed and solution quality.

3.3.2 Chance-constrained Optimization

A chance-constrained optimization model is developed in (Shylo et al., 2012) for a surgery admission problem with uncertain surgery durations. The model is solved using normal approximation for the sum of surgery durations to simplify the chance constraints and

provide near optimal solutions within acceptable computation time. The high accuracy of their solution approach is assessed using a discrete simulation model.

3.3.3 Robust Optimization

A robust optimization approach to the advance scheduling problem is presented in (Addis et al., 2014). The notion of robustness is as follows: any solution is feasible if all constraints hold when at most Γ of the uncertain variables associated with each constraint take the maximum of a given interval (that represents the duration uncertainty) and the rest of them take the central value. Solutions to the robust model with varying values of Γ are compared to solutions to a deterministic model assuming expected values for surgery durations. Another robust optimization approach to a similar problem is presented in (Hans et al., 2008). The robustness is incorporated by adding slack to expected durations based on historical data from a real hospital. A set of constructive heuristics are compared to two local search methods (*random exchange method* and the SA metaheuristic) and the solutions are evaluated using Monte Carlo simulation.

3.3.4 Multiobjective Surgery Scheduling under Duration Uncertainty

In (Gul et al., 2011), they model surgery durations with lognormal distributions and experiment with different percentiles (i.e. different hedging levels) used during optimization with NSGA-II. They find, by simulating the effect of surgery duration variation on the produced schedules (*after* optimization), that using the 65^{th} percentile during optimization yielded the best results (56% of the efficient solutions found overall used this hedging level, with only 21% using the 70^{th} percentile and 8% using the 60^{th} percentile).

Chapter 4

Detailed Problem Formulation

This chapter details the investigated problem mathematically. We mention our assumptions in the first section before we present the multiobjective surgery admission planning model formulation in the second section.

4.1 Model Assumptions

In order to model the surgery admission planning problem in a comprehensible, implementable and computationally tractable manner, some assumptions must be made:

(i) the patient waiting list is static with every patient having a preassigned surgeon,

(ii) every patient is available at any day and time in the planning period,

(iii) all surgeons are available at any day and time in the planning period,

(iv) the considered resources are reserved for elective patients only (no emergency cases),

(v) there is no explicit difference between inpatients and outpatients,

(vi) all patients have been categorized according to surgical subspecialties,

(vii) all surgical subspecialties are covered by the master surgical schedule,

($viii$) necessary facilities and equipment are available in accordance with the master surgery schedule,

(ix) patients, equipment and staff always arrive on time (no arrival uncertainty),

(x) surgeons and operating rooms are the only considered resources (equipment and other staff are disregarded),

(xi) preoperative and postoperative capacity is not considered to limit the flow of patients through the surgery delivery system,

(xii) surgeons may switch between operating rooms without any turnover time,

($xiii$) all scheduled surgeries must be completed without interruption (no preemption),

(xiv) surgery durations follow procedure-dependent lognormal distributions and are in-

dependent of each other and all other decisions, such as scheduled sequence and allocated operating room,

(xv) there are no rescheduling activities during schedule execution.

We regard some of these assumptions as more critical than the others. For instance, rescheduling activities, such as cancellations or remobilization of staff, can take place quite frequently during a normal week of surgery. Still, we make this assumption because it significantly simplifies our model. Because no rescheduling activities are allowed, the execution of a schedule can be simulated as a single event where all durations are observed simultaneously. Surgeries are simply postponed or brought forward in time as realized surgery duration are observed. This assumption can be justified to some degree by noting that the objective in the modeled problem partly is to eliminate the need to reschedule. Also, by disregarding resources needed before and after surgery, solutions to our model may cause overload outside the operating room. However, it is broadly acknowledged in literature that the operating room is the bottleneck in most surgery delivering systems (Cardoen et al., 2010).

4.2 Mathematical Problem Formulation

Surgeries $i \in N$ from a patient waiting list are to be scheduled over a planning period of $d \in D$ days. Each day there are $r \in R$ available operating rooms that are reserved capacity according to a master surgery schedule. Every surgery belongs to a certain subspecialty $s \in S$ ($\sigma_{is} = 1$, if surgery i is of subspecialty s; $\sigma_{ik} = 0$, otherwise) and a qualified surgeon $k \in K$ is already assigned to each surgery ($\delta_{ik} = 1$, if surgery i is pre-assigned to surgeon k; $\delta_{ik} = 0$, otherwise). The goal is to select a set of surgeries from the patient waiting list and schedule them in rooms during the planning period so as to minimize patient waiting time on the waiting list, operating room overtime, surgeon overtime, operating room idle time and surgeon idle time. Formally, the decisions are as follows. At the advance scheduling level, the decision is whether a surgery $i \in N$ should be scheduled in a room $r \in R$ on a day $d \in D$, denoted as a binary decision variable:

$$x_{idr} = \begin{cases} 1 & \text{if surgery } i \text{ is scheduled on day } d \text{ in room } r \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

To simplify some expressions in our formulation, we also introduce a binary helping variable, z_{idk}, to denote whether a surgery i is performed by surgeon k on a given day, d:

$$z_{idk} = \delta_{ik} \sum_{r=1}^{R} x_{idr} = \begin{cases} 1 & \text{if surgery } i \text{ is performed by surgeon } k \text{ on day } d \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

These decisions must adhere to the master surgery schedule ($M_{drs} = 1$, if the master surgery schedule allows operating room r to be scheduled by specialty s on day d; $M_{drs} = 0$, otherwise). This is ensured by adding the following constraint:

$$x_{idr} \leq M_{drs} \tag{4.3}$$

At the allocation scheduling level, the decision is the sequence of surgeries on a given day (binary decision variables), and the starting time of each surgery (continuous decision variables):

$$y_{ij} = \begin{cases} 1 & \text{if surgery } i \text{ precedes surgery } j \text{ on a given day} \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

$$t_i \geq 0 : \text{scheduled starting time for surgery } i \tag{4.5}$$

The y_{ij} variables define precedence relations among surgeries. Note that immediate precedence of two surgeries is not necessary for there to be a precedence relation between them. It is only required that two surgeries are performed by the same surgeon or in the same operating room on the same day, as we specify in the following constraints:

$$y_{ij} + y_{ji} \leq 1 \tag{4.6}$$

$$y_{ij} + y_{ji} \geq x_{idr} + x_{jdr} - 1 \tag{4.7}$$

$$y_{ij} + y_{ji} \geq z_{idk} + z_{jdk} - 1 \tag{4.8}$$

The first constraint, Equation 4.13, is a basic precedence constraint stating that if surgery i precedes surgery j, then surgery j cannot precede surgery i. Equation 4.7 states that if two surgeries are scheduled in the same operating room on the same day, there must be a precedence relation between them. Equation 4.8 does the same for surgeons, i.e. there must be a precedence relation between surgeries preformed by the same surgeon on the same day.

A determination of the advance scheduling decision variables, x_{idr}, and the precedence relation variables, y_{ij}, allows for computation of scheduled staring times, t_i, using a duration estimate, $\mathcal{D}_i$, for each surgery $i \in N$. The scheduled starting times and resulting estimated ending times must adhere to normal opening hours of operating rooms (from starting time E^b_{dr} to ending time E^e_{dr}) and surgeons (from starting time E^b_{dk} to ending time E^e_{dk}). A certain amount of overtime, V, is allowed for both surgeons and operating rooms. Imposing the following constraints, expected starting times are computed:

$$t_i \geq t_j + \mathcal{D}_j - M(1 - y_{ji}) \tag{4.9}$$

$$t_i \geq max\Big\{ \sum_{r=1}^{R} x_{idr} E^b_{dr}, \sum_{k=1}^{K} z_{idk} E^b_{dk} \Big\} \tag{4.10}$$

$$t_i + \mathcal{D}_i \leq max\Big\{ \sum_{r=1}^{R} x_{idr} E^e_{dr}, \sum_{k=1}^{K} z_{idk} E^e_{dk} \Big\} + V \tag{4.11}$$

Equation 4.9 computes the starting times by adding up starting times and durations of preceding surgeries. A *big M method*[1] ensures that only preceding surgeries of a surgery i take part in the computation. Equation 4.10 and Equation 4.11 sets lower daily limits for starting times and upper daily limits for ending times, respectively. Once the starting times are computed, the starting times and ending times for surgeons may be set. In our model, we do not count surgeons as idle unless they are waiting in between surgeries within a daily interval of surgeries. The scheduled starting time, e^b_{dk}, and ending time, e^e_{dk}, for a surgeon coincide with the earliest and latest surgery assigned to that surgeon on a given day, respectively. These times are actually decisions in the model that may be derived directly from the timing of the surgeries:

$$e^b_{dk} = \min_{i \in N} z_{idk} t_i \tag{4.12}$$

$$e^e_{dk} = \max_{i \in N} z_{idk} (t_i + \mathcal{D}_i) \tag{4.13}$$

The waiting time objective, O_W, may be computed before schedule execution (because we do not allow cancellations of patients). The waiting time cost is based on the referral date, H_i, and the deadline, G_i, of surgery i in relation to the scheduled date, d, for the surgery:

$$O_W = \frac{1}{N} \sum_{i=1}^{N} \sum_{d=1}^{D} \sum_{r=1}^{R} x_{idr} \left(\frac{d - H_i}{G_i - H_i}\right)^2 \tag{4.14}$$

The waiting time cost expressed in Equation 4.14 is incurred on all scheduled surgeries. For surgeries that are not scheduled, we assume that they may be completed within the next scheduling period and assign the latest date in the next period, $d = 2|D|$, to compute their contribution to the waiting cost component.

The other objectives, surgeon idle time (O^K_I), surgeon overtime (O^K_O), operating room idle time (O^R_I) and operating room overtime (O^R_O), may be computed once the realized durations, $\mathcal{D}_i(\omega)$, are observed after schedule execution. Here, ω, denotes the collective outcome of all surgery durations and allows for computation of realized starting times, $t_i(\omega)$:

$$t_i(\omega) \geq t_j(\omega) + \mathcal{D}_j(\omega) - M(1 - y_{ji}) \tag{4.15}$$

$$t_i(\omega) \geq max\Big\{ \sum_{r=1}^{R} x_{idr} E^b_{dr}, \sum_{k=1}^{K} z_{idk} E^b_{dk} \Big\} \tag{4.16}$$

Note that there is no restriction on the ending time of surgeries corresponding to Equation 4.11. This is because *planned* overtime is not allowed beyond V added to ending times for surgeons and operating rooms, but *actual* overtime is unrestricted. In other

[1] The big M method (Winston et al., 2003) is often used to keep linearity in models with binary decision variables. For sufficiently large M, constraints may be turned on or off depending on the value of the decision variables.

words, we assume that all surgeries must be completed, no matter how much overtime is incurred. In addition to Equation 4.15 and Equation 4.16, we restrict how much earlier than planned surgeries may start. This is to address the assumption that the operating rooms and surgeons are the only critical resources associated with treatment of patients. Other resources, such as nurses, equipment and preoperative and postoperative resources, also follow potentially tight schedules, and these resources are not always ready to perform their task before the scheduled time. The following constraint imposes that no surgery can start T minutes earlier than scheduled:

$$t_i(\omega) \geq t_i - T \tag{4.17}$$

Once the realized starting times, $t_i(\omega)$, and (implicitly) completion times, $t_i(\omega) + \mathcal{D}_i(\omega)$, have been computed, the other objectives are calculated as follows.

$$O_O^R = \sum_{d=1}^{D} \sum_{r=1}^{R} \max_{i \in N} \Big(max\big\{ t_i(\omega) x_{idr} - E_{dr}^e, 0 \big\} \Big) \tag{4.18}$$

$$O_O^K = \sum_{d=1}^{D} \sum_{k=1}^{K} \max_{i \in N} \Big(max\big\{ t_i(\omega) z_{idk} - E_{dk}^e, 0 \big\} \Big) \tag{4.19}$$

$$O_I^R = \sum_{d=1}^{D} \sum_{r=1}^{R} \left[(E_{dr}^e - E_{dr}^b) - \sum_{i=1}^{N} \mathcal{D}_i(\omega) x_{idr} + \max_{i \in N} \Big(max\big\{ t_i(\omega) x_{idr} - E_{dr}^e, 0 \big\} \Big) \right] \tag{4.20}$$

$$O_I^K = \sum_{d=1}^{D} \sum_{k=1}^{K} \left[(e_{dk}^e - e_{dk}^b) - \sum_{i=1}^{N} \mathcal{D}_i(\omega) z_{idk} + \max_{i \in N} \Big(max\big\{ t_i(\omega) z_{idk} - E_{dr}^e, 0 \big\} \Big) \right] \tag{4.21}$$

The overall objective is thus to simultaneously minimize patient waiting time (O_W) and resource efficiency ($O_O^R, O_O^K, O_I^R, O_I^K$) subject to all surgery duration realizations, $\omega \in \Omega$. The objective can be formulated as follows where $\xi(\omega)$ denotes the random vector of surgery durations:

$$\begin{aligned} &\min \quad \theta(x, y, t) = O_W, \\ &E_\xi \big[O_O^R(x, y, t, \xi(\omega)), O_O^K(x, y, t, \xi(\omega)), O_I^R(x, y, t, \xi(\omega)), O_I^K(x, y, t, \xi(\omega)) \big] \end{aligned} \tag{4.22}$$

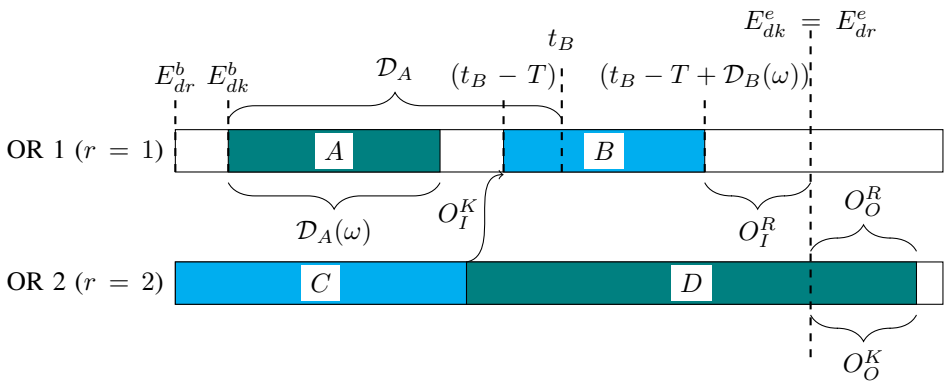


Figure 4.1: Model notation.

We illustrate some of the model notation in Figure 4.1. In this illustration, the colors represent two surgeons, so that surgery A and D are performed by the green surgeon, and surgery C and B are performed by the blue surgeon. In this example we have the following precedence relations: $y_{AB} = 1, y_{CD} = 1$ (operating room precedence) and $y_{AD} = 1, y_{CB} = 1$ (surgeon precedence). This illustration shows only one of possibly many days, $d \in D$, in the planning period. Note that operating room overtime, O^R_O, and surgeon overtime, O^K_O, often will coincide if normal surgeon ending times equal operating room ending times (as in the figure; $E^e_{dk} = E^e_{dr}$). The waiting time cost component, O_W, is not visible in the figure.

Chapter 5

Implemented Algorithms

In this chapter we give a detailed description of our implemented algorithms: SPEA2, NSGA-II and a singleobjective hybrid GA. All algorithms are based on the same genetic submechanisms that we highlight in the first section of this chapter. In order to efficiently solve the problem described in Chapter 4, we take inspiration from machine scheduling literature, seeing that the operating rooms are analogous to machines, surgeons are analogous to jobs and that surgeries are analogous to operations.

5.1 Genetic Submechanisms

As discussed in subsection 2.4.2, evolutionary algorithms mimic mechanisms that drive evolution in nature such as mutation, selection, competition and inheritance. In the following, we suggest new genetic mechanisms for applying MOEAs to the surgery admission planning problem under uncertainty.

5.1.1 Solution Representation

Using machine scheduling terminology, the surgery scheduling problem under investigation may be described as a multiobjective stochastic flexible open job shop scheduling problem with non-identical machines. The selected representation must be translated into a solution of the problem defined in Chapter 4, i.e. enable a determination of the room-day allocations (including which surgeries to include), x_{idr}, the precedence relations among them, y_{ij}, and the timing of surgeries, t_i. We let the chromosomes dictate the combinatorial part of the problem, i.e. the x_{idr} and y_{ij} binary variables, and use left shifting with the duration estimates, $\mathcal{D}_i$, to create active schedules and determine the continuous timing variables, t_i, during decoding.

Our selected representation is inspired by the two-vector representation used for fJSP in (Gao et al., 2007), where one vector represents machine assignments and the other represents the operation sequence. Two critical modifications must be made. First, since the problem under investigation considers fJSP over multiple days, the machine assignment vector must consist of tuples representing combinations of rooms and days. This is also necessary to control the feasibility of chromosomes (i.e. that the room-day combinations adhere to the master surgery schedule). Second, since there is no predetermined sequence among the surgeries, our scheduling problem is open, and the operation-based representation used in the sequence vector in (Gao et al., 2007) cannot consistently represent all possible sequences among operations. We therefore use a permutation-based representation in the sequence vector that dictates the sequence in which surgeries are fitted into the schedule during decoding. In our implementation, we use chromosome initialization, mutation and crossover operators that in combination with the decoding procedure guarantees feasibility. An example of a chromosome is shown in Figure 5.1.

$locus$	1	2	3	4	5	6
v_1	$(1,1)$	$(2,2)$	$(1,2)$	$(2,1)$	$(2,1)$	$(1,1)$
v_2	1	4	2	3	5	6

Figure 5.1: Two-vector chromosome representation. The room-day vector, v_1, indicates the room and day chosen for the surgery at that locus. The precedence vector, v_2, indicates the order in which the surgeries are put into the schedule.

5.1.2 Decoding

Chromosomes are translated into feasible schedules using a new variation of priority-based decoding and reordering. The decoding procedure must handle the continuous time model and ensure that operating room schedules and surgeon schedules are compatible (i.e. that a surgeon cannot be present in two operating rooms at the same time). We describe the decoding procedure informally in the following.

For every surgery, i, encountered in the chromosome from left to right in the precedence vector, v_2, the operating room, r, and day, d, for the surgery are found at the i^{th} position in the room day vector, v_1. The assigned surgeon, k, and an estimate for the surgery duration, $\mathcal{D}$, are found in provided surgery data. During decoding, both surgeon and operating room schedules are built from left to right while maintaining an overview of surgeon and operating room idle time intervals that are created as surgeons move in between operating rooms. If the surgery, i, cannot be scheduled within any idle time according to its duration, $\mathcal{D}_i$, it is scheduled at the end of the day. In this way, the schedule is always active, as scheduling a surgery in the earliest possible idle time interval represents a global left shift. Consider the schedule under construction in Figure 5.2. Surgeries assigned to surgeon B have been scheduled in operating room 3, 2 and 1, incurring operating room idle time at the start of the day in operating room 1 and 2. After this, a series of surgery inclusions require different kinds of modifications to the maintained set of surgeon and operating room idle time

intervals. We describe these modifications in the following where the superscripts indicate the sequence in which surgeries are included in the schedule:

(i) including surgery C^1 and A^2 reduces the operating room idle time interval at the start of the day in operating room 1 and 2, respectively,

(ii) including A^3 and C^4 creates two surgeon idle time intervals between surgery A^2 and A^3; and C^1 and C^4, respectively,

(iii) including A^5 splits the operating room idle time interval in operating room 1 between surgery C^1 and B in two. Additionally, the surgeon idle time interval between A^2 and A^5 is shortened to be from A^5 to A^3,

(iv) including C^6 splits the surgeon idle time interval between surgery C^1 and C^4 in two,

(v) including surgery C^7 creates a new surgeon idle time interval between surgery C^4 and surgery C^7,

(vi) including surgery A^8 creates a new operating room idle time interval between surgery C^4 and A^8.

All of the exemplified scenarios must be taken care of during decoding to ensure that the idle time intervals are updated.

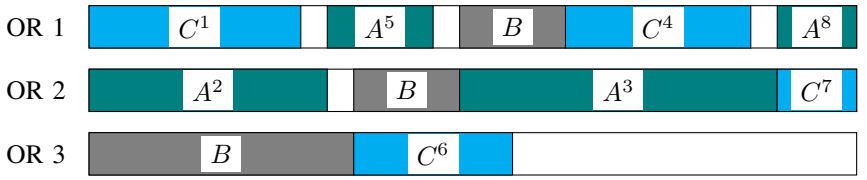


Figure 5.2: Idle time intervals.

In Figure 5.3 we illustrate the decoded chromosome from Figure 5.1. The surgeries are scheduled in the order specified by the precedence vector, v_2, in Figure 5.1 - $\{1, 4, 2, 3, 5, 6\}$:

(i) surgery 1 is scheduled in operating room 1 on day 1,

(ii) surgery 4 is scheduled in operating room 2 on day 1 (because it is performed by the same surgeon as surgery 1, it must start after surgery 1 is finished),

(iii) surgery 2 is scheduled in operating room 2 on day 2,

(iv) surgery 3 (performed by the same surgeon as surgery 2) is *not included* in the schedule, because it would incur overtime beyond the allowed limit of $E^e_{dk} + V$ or $E^e_{dr} + V$,

(v) surgery 5 is scheduled in the idle time interval in operating room 2 on day 1 (this is an example of a global left shift) and

(vi) surgery 6 is scheduled in operating room 1 on day 1, incurring an allowed amount of planned overtime

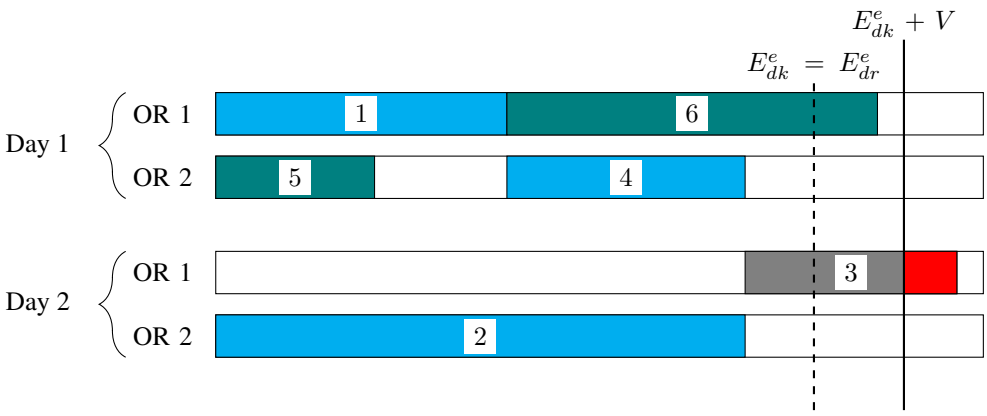


Figure 5.3: The chromosome in Figure 5.1 after decoding.

Three aspects of the decoding procedure are important to note. First, global left shifts may alter the sequence specified in the precedence vector. For instance, the precedence vector in Figure 5.1 specifies that surgery 4 has priority over surgery 5, but since surgery 5 fits into the idle time interval between the normal opening hours of operating room 2 and the beginning of surgery 4, the original sequence is altered. Second, which surgeries to include is not explicitly stated in the chromosome. It is discovered during the decoding procedure as the operating rooms are filled (see for instance surgery 3 in Figure 5.3). Third, because certain surgeries are omitted from the schedule, several chromosomes can lead to the same schedule (n : 1-mapping). For instance, any room-day allocation of surgery 3 ($\{1,1\}$,$\{1,2\}$,$\{2,1\}$,$\{2,2\}$) would lead to the exclusion of surgery 3 from the schedule, and thus, four (marginally different) chromosomes can lead to the same effective schedule.

5.1.3 Fitness Evaluation

Once a chromosome has been decoded into a feasible schedule, fitness evaluation is done according to

(i) Equation 4.14 (patient waiting time),
(ii) Equation 4.18 (opreating room overtime),
(iii) Equation 4.19 (surgeon overtime),
(iv) Equation 4.20 (operating room idle time), and
(v) Equation 4.21 (surgeon idle time).

The patient waiting time objective, O_W, is independent of the outcome of surgery durations, ω, and may be deterministically computed without any further computation. Taking the example in Figure 5.3 with the following referral dates, H_i (backward in time), and deadlines, G_i (forward in time), denoted (H_i, G_i), for surgeries $\{1, 2, ..., 6\}$:

$\{(3,1),(10,1),(4,2),(20,3),(6,1)\}$,

we get the following patient waiting time cost contribution from each surgery (according to the formula $(\frac{d-H_i}{G_i-H_i})^2$):

(i) surgery 1, scheduled at day 1: $O_W \mathrel{+}= (\frac{1-(-3)}{1-(-3)})^2 = 0 + 1 = 1$,

(ii) surgery 2, scheduled at day 2: $O_W \mathrel{+}= (\frac{2-(-10)}{1-(-10)})^2 = 1 + 1.190 = 2.190$,

(iii) surgery 3, *not scheduled*[1]: $O_W \mathrel{+}= (\frac{4-(-4)}{2-(-4)})^2 = 2.190 + 1.778 = 3.968$,

and so forth.

The computation of operating room overtime (O_O^R), surgeon overtime (O_O^K), operating room idle time (O_I^R) and surgeon idle time (O_I^K) requires a determination of the starting time decision variables, $t_i(\omega)$. For a single collective outcome of surgery durations (ω), the computation is straightforward. Because we do not allow any rescheduling activities, the surgery sequence and allocations remain constant over all possible realizations of ω. Simulation of a realized schedule involves only shifting starting times (and completion times) of surgeries to the right or left depending on the realized duration of each surgery. An example of a realized schedule is shown in Figure 5.4.

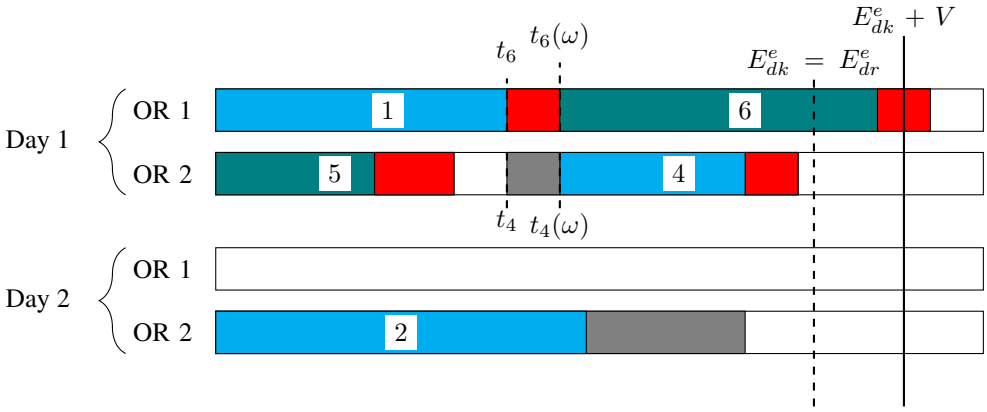


Figure 5.4: Realized schedule.

We defined the objective function in Equation 4.22 as:

$$\begin{aligned} &\min \quad \theta(x, y, t) = O_W, \\ &E_\xi \left[O_O^R(x, y, t, \xi(\omega)), O_O^K(x, y, t, \xi(\omega)), O_I^R(x, y, t, \xi(\omega)), O_I^K(x, y, t, \xi(\omega)) \right] \end{aligned} \tag{5.1}$$

The corresponding effective fitness function can thus be stated (Jin and Branke, 2005) for the case where the environmental variables are uncertain:

[1] assumes the last day in the next period, $d = 2|D| = 4$

$$
\begin{aligned}
F(x,y,t) = \big\{ & O_W(x,y,t), \\
& \int_{-\infty}^{\infty} O_O^R(x,y,t,\omega)p(\omega)d\omega, \\
& \int_{-\infty}^{\infty} O_O^K(x,y,t,\omega)p(\omega)d\omega, \\
& \int_{-\infty}^{\infty} O_I^R(x,y,t,\omega)p(\omega)d\omega, \\
& \int_{-\infty}^{\infty} O_I^K(x,y,t,\omega)p(\omega)d\omega \big\}
\end{aligned}
\tag{5.2}
$$

Since the effective fitness function does not have a closed form, it cannot be computed exactly. The obvious estimator for the effective fitness function is the Monte Carlo approximation:

$$
\begin{aligned}
F(x,y,t) = \big\{ & O_W(x,y,t), \\
& \frac{1}{N}\sum_{i=1}^{N} O_O^R(x,y,t,\omega_i), \\
& \frac{1}{N}\sum_{i=1}^{N} O_O^K(x,y,t,\omega_i), \\
& \frac{1}{N}\sum_{i=1}^{N} O_I^R(x,y,t,\omega_i), \\
& \frac{1}{N}\sum_{i=1}^{N} O_I^K(x,y,t,\omega_i) \big\}
\end{aligned}
\tag{5.3}
$$

As mentioned in section 2.7.4, using Monte Carlo integration to estimate the fitness of an individual in robust MOEAs is referred to as explicit averaging. The accuracy of the estimation increases with the sample size, N. However, increasing N also increases the number of fitness evaluations. In our case, decoding is needed only once, but the simulation and calculation of O_O^R, O_O^K, O_I^R and O_O^K must be done N times.

5.1.4 Crossover

We implement a version of the enhanced order crossover operator presented in (Gao et al., 2007) to produce a feasible child chromosome, c, of two selected parent chromosomes, p_1 and p_2. The crossover is executed in three steps. First, a subsection of the genes belonging to the first parent, p_1, is directly inherited by the child, c_1. Then, missing room-day allocations in the child, c_1, are filled with values from the room-day allocation vector, v_1, of the second parent, p_2. Finally, the integers missing in the child's permutation vector, v_2, are filled in from the second parent, p_2, starting from the left-most locus in v_2. The child

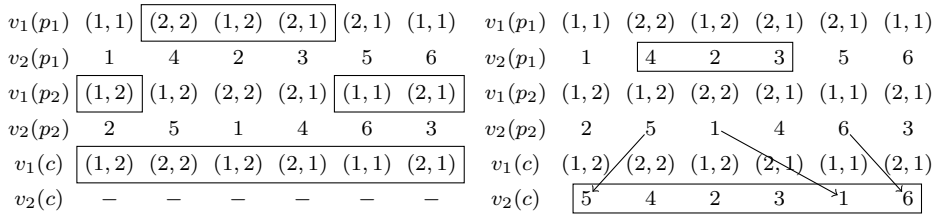


Figure 5.5: Crossover operation. Crossover of two feasible parent chromosomes, p_1 and p_2, resulting in a feasible child chromosome, c.

chromosome will always decode into a feasible schedule because the room-day allocations are inherited directly (the locus is unchanged) from the parents which have room-day allocations that adhere to the master surgery schedule. The crossover operation is illustrated for coding space in Figure 5.5 and for the decoded parents and child chromosome in Figure 5.6, Figure 5.7 and Figure 5.8, respectively.

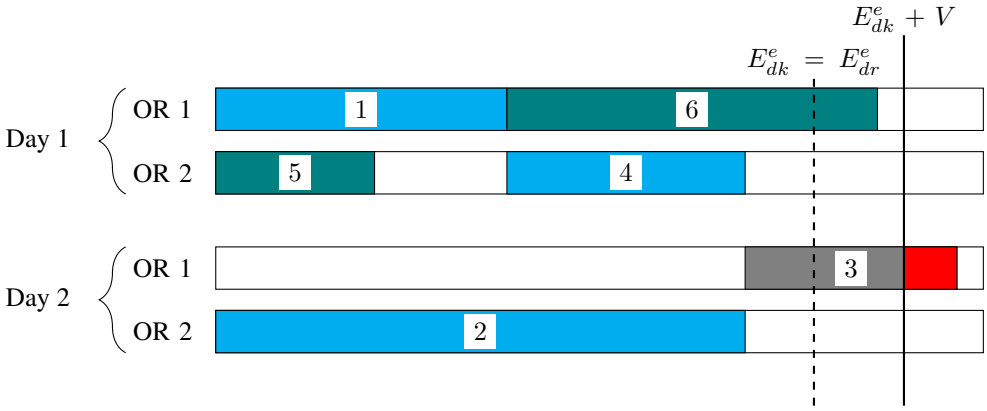


Figure 5.6: Parent 1.

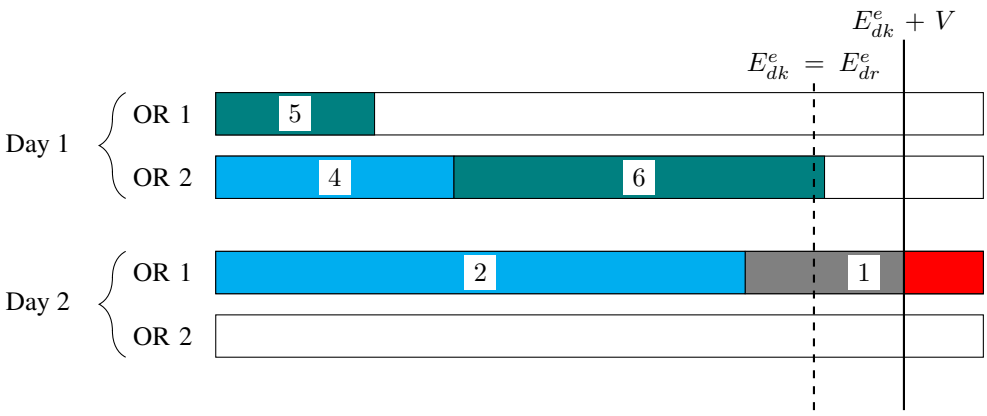


Figure 5.7: Parent 2. Surgery 3 is also omitted from the schedule (impossible to schedule after surgery 1.)

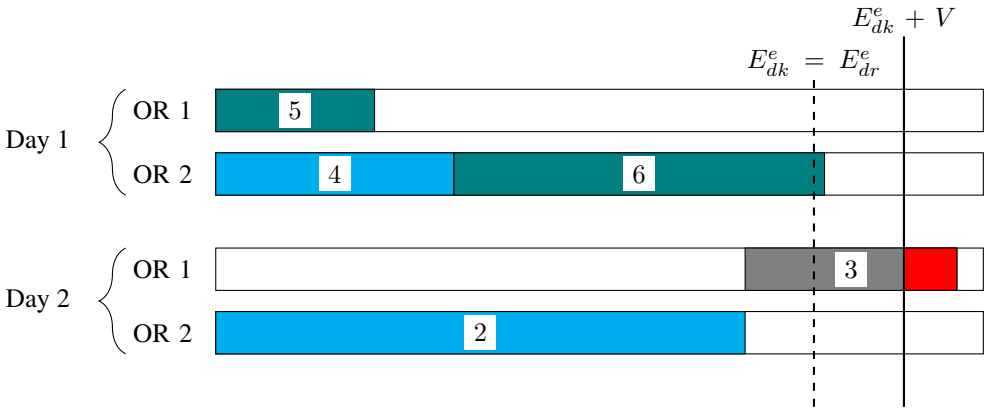


Figure 5.8: Child. Surgery 1 is also omitted from the schedule (impossible to schedule after surgery 3).

5.1.5 Mutation

We implement three mutation operators:
(i) room-day swap mutation,
(ii) precedence swap mutation and
(iii) renew room-day mutation.
All mutations maintain the feasibility of the mutated chromosome. The room-day swap mutation interchanges the room-day assignments (v_1) of two randomly chosen surgeries that are categorized by the same surgical subspecialty. This mutation operator is illustrated in Figure 5.10, where surgery 1 and surgery 4 (belonging to the same subspecialty) interchanges operating room in the same day. The precedence swap mutation, illustrated in

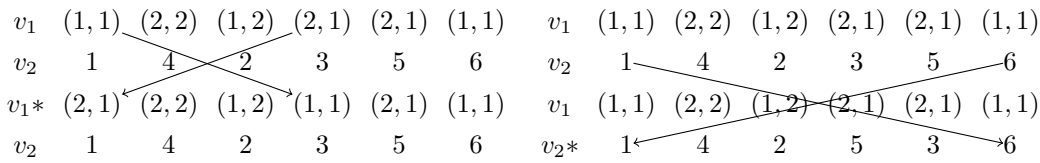


Figure 5.9: Swap mutation operators. Mutation operators with mutated vectors marked (*).

Figure 5.11, interchanges two random precedence values in v_2, thus altering the sequence in which surgery 4 and surgery 6 are scheduled. The renew room-day mutation, illustrated in Figure 5.12, assigns a new feasible room-day combination to a randomly selected surgery in v_1. Note that not only surgeries involved in the mutations are affected. A single mutation can have a large impact that is seen only after the chromosome is decoded. How the swap mutations work on the chromosome is illustrated in Figure 5.9. A room-day mutation is shown to the left. In this example, it is necessary that the surgery at locus 0 and locus 3 belong to the same surgical subspecialty - they can both be scheduled in OR 1 or OR 2 on day 1. A precedence mutation is shown to the right.

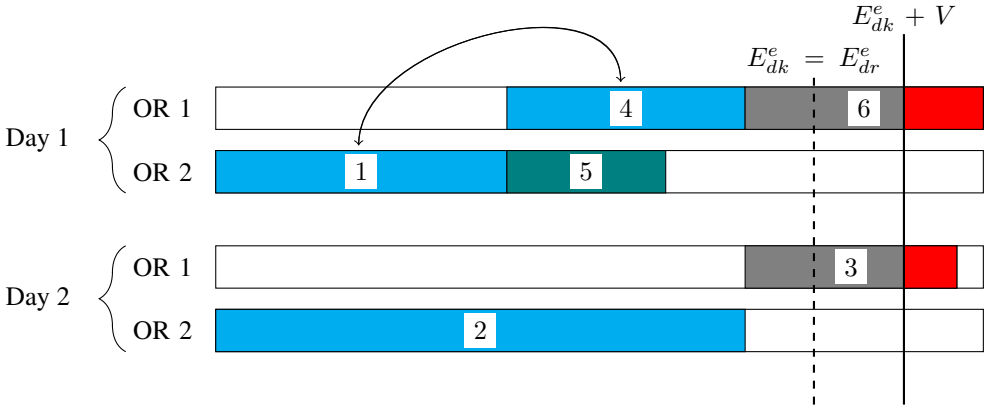


Figure 5.10: Room-day swap mutation.

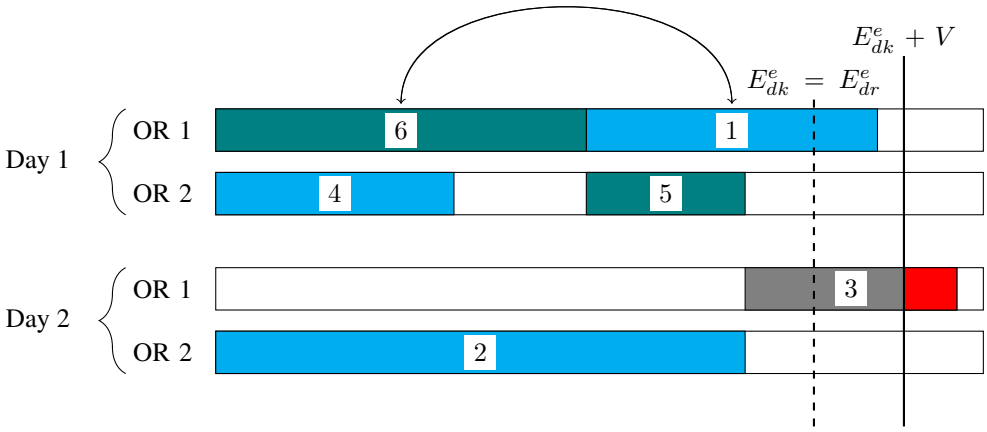


Figure 5.11: Precedence swap mutation.

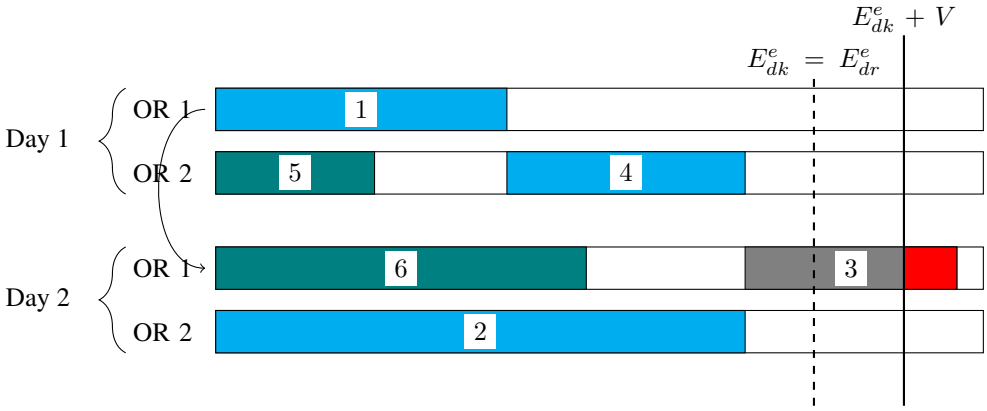


Figure 5.12: Renew room-day mutation.

5.1.6 Population Initialization

We initialize a random feasible individual in the following manner. Room-day assignments from the master surgery schedule are selected for each surgery with uniform probability to construct the room-day vector, v_1. That is, all room-days that are reserved for the surgical subspeciality that each surgery belongs to is equally likely to be selected for the surgery under consideration. The precedence vector, v_2, is initialized simply by randomly perturbing the integers $i \in N$. The population is initialized by repeating this procedure and adding the random feasible individuals to the population until the population is full.

5.2 SPEA2

From subsection 2.5.5 we known that a successful MOEA should provide solutions with minimal distance to the true Pareto-front and maximal diversity among the solutions. The SPEA2 maintains a fixed-sized archive of non-dominated solutions (close to the Pareto-front) that are protected from altering evolutionary mechanisms. Whenever the archive is overfull, solutions in dense areas of the objective space are removed in order to maximize diversity. A detailed description of the SPEA2 is given in (Zitzler et al., 2001). We render the most important details here.

The SPEA2 starts with an initial population P of size $|P|$ and an empty archive $\overline{P}$ of size $|\overline{P}|$. During evolution, non-dominated individuals are added to the improving archive in a way that preserves diversity. All individuals are assigned a fitness value that is based on the dominance relations among all individuals (in the population, P, and the archive, $\overline{P}$) and a density value based on the nearest neighbours of individuals. Three measures must be computed to determine the fitness of an individual $i \in P + \overline{P}$:

(i) the *strength value*, $S(i)$,
(ii) the *raw fitness*, $R(i)$, and
(iii) the *density*, $D(i)$.

The strength value, $S(i)$, is the number of solutions dominated by i:

$$S(i) = |\{j | j \in P + \overline{P} \wedge i \succ j\}| \tag{5.4}$$

The raw fitness, $R(i)$, is the combined strength values of all dominators of i:

$$R(i) = \sum_{j \in P+\overline{P}, j \succ i} \tag{5.5}$$

Low values of raw fitness indicate closeness to the Pareto-front. Non-dominated individuals have a raw fitness of 0, while individuals that are dominated by many or strong individuals have high raw fitness. As a measure for differentiating between individuals with identical raw fitness, $R(i)$, the density of a solution, $D(i)$, is computed as follows,

$$D(i) = \frac{1}{\sigma_i^k + 2} \tag{5.6}$$

where σ_i^k is the distance to the k$^{\text{th}}$ nearest neighbour in the objective space, usually with $k = \sqrt{(|P| + |\overline{P}|)}$. The fitness, $F(i)$, used during environmental selection in the algorithm, is the aggregate of an individuals raw fitness, $R(i)$, and density, $D(i)$:

$$F(i) = R(i) + D(i) \tag{5.7}$$

Because the distance, σ_i^k, is always positive, it follows that $D(i) < 1$ for all individuals, i. As a consequence, non-dominated individuals are easily recognizable as individuals with

a fitness value, $F(i) < 1$. In the first step of environmental selection, all non-dominated solutions (with fitness value $F(i) < 1$) are copied from the current population, P_t, and archive, $\overline{P}_t$, to the next generation archive, $\overline{P}_{t+1}$. The next step is to ensure that the the number of individuals in the archive equals the specified archive size, $|\overline{P}|$. There are three possible situations:

(i) $|\overline{P}_{t+1}| = |\overline{P}|$, in which case no further action is required at this point,

(ii) $|\overline{P}_{t+1}| < |\overline{P}|$, i.e. there are too few solutions in the next generation archive, $\overline{P}_{t+1}$, in which case the next generation archive is filled with the fittest dominated individuals remaining in $P_t + \overline{P}_{t+1}$ or

(iii) $|\overline{P}_{t+1}| > |\overline{P}|$, i.e. the next generation archive is overfull and non-dominated individuals must be removed until the archive size is right.

In the situation where non-dominated solutions must be removed from the archive, a diversity-preserving truncation procedure is invoked iteratively to remove, at each iteration, the solution with the closest distance to any other solution.

The implemented SPEA2 applies the developed genetic mechanisms in the standard flow of SPEA2 described in (Zitzler et al., 2001). Pseudocode is given in algorithm 1.

1: $t \leftarrow 0$
2: $P_t \leftarrow$ initialize population {population}
3: $\overline{P}_t \leftarrow \phi$ {archive}
4: **while** !$stop$ **do**
5: $fintessAssignment(P_t)$
6: $fintessAssignment(\overline{P}_t)$
7: $\overline{P}_{t+1} = \{i | i \in (P_t \cup (P)_t) \cap F(i) < 1\})$
8: **if** $|\overline{P}_{t+1}| < |\overline{P}|$ **then**
9: fill $\overline{P}_{t+1}$ with the fittest dominated individuals in P_t
10: **end if**
11: **if** $|\overline{P}_{t+1}| > |\overline{P}|$ **then**
12: $truncate(\overline{P}_{t+1})$
13: **end if**
14: **if** $t > maxGenerations$ **then**
15: output $\overline{P}_{t+1}$
16: $stop$
17: **end if**
18: $P_{t+1} \leftarrow \phi$
19: **while** $|P_{t+1}| < |P|$ **do**
20: $p_1 \leftarrow binaryTournamentSelection(\overline{P}_{t+1})$ {Parent 1}
21: $p_2 \leftarrow binaryTournamentSelection(\overline{P}_{t+1})$ {Parent 2}
22: $P_{t+1} \leftarrow P_{t+1} \cup crossover(p_1, p_2)$
23: **end while**
24: mutate(P_{t+1})
25: $t \leftarrow t + 1$
26: **end while**

Algorithm 1: SPEA2 procedure (Zitzler et al., 2001).

5.3 NSGA-II

The NSGA-II presented in (Deb et al., 2002) pursues convergence and diversity, not by manipulating an external archive, but by a two-level sorting and selection procedure of individuals in the merged parent population and child population. From a parent population, P_t, (initially a random population), individuals selected with binary tournaments are crossed over to create a a new population, Q_t with $|Q_t| = |P_t| = |P|$. Next, $|P|$ individuals must be selected from the combined population, $R_t = P_t + Q_t$, to create the next parent population, P_{t+1}.

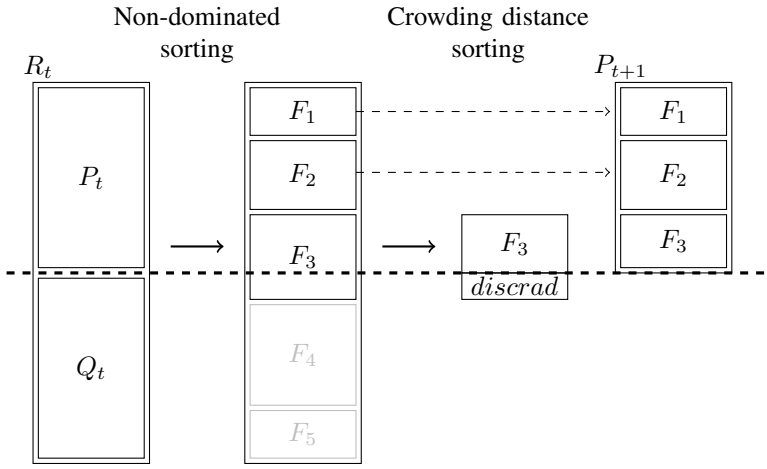


Figure 5.13: NSGA-II two-level sorting procedure.
Adapted from: Deb et al. (2002)

The merged population, R_t, is sorted according into Pareto fronts, $F_l \in \mathcal{F}$, of different levels, l, where all individuals on the same front are dominated by equally many $(l - 1)$ other individuals in R_t. Individuals in F_1 are not dominated by any solutions in R_t, individuals in F_2 are dominated only by individuals in F_1, individuals in F_3 are dominated only by individuals in F_1 and F_2 and so forth. This partitioning of solutions into fronts is illustrated in Figure 5.14. Individuals are sorted into fronts using the *fast-non-dominated sorting* procedure. Pseudocode for this procedure is given in algorithm 2.

When individuals have been sorted into fronts, $F_l \in \mathcal{F}$, with the fast-non-dominated-sorting procedure, individuals are added to the next generation, P_{t+1}, starting with individuals in the firsts fronts, F_1, F_2, F_3, etc., as long as all individuals of a front can be included in P_{t+1} without breaching the population size limit, $|P|$. At this point, it must be recalled that individuals on the same front represent equally valuable trade-off solutions in terms of solution quality and closeness to the Pareto-front. Another sorting procedure that preserves diversity, namely the *crowding distance sorting* procedure, is used to include only the most unique solutions from the front under investigation. All solutions on the front are assigned a *crowding distance* value that estimates the average side length of the

1: **for all** $i \in R_t$ **do**
2: $S_i = \phi$ {Set of solutions dominated by i}
3: $n_i = 0$ {Number of solutions that dominate i}
4: **for all** $j \in R_t$ **do**
5: **if** $(i \prec j)$ **then**
6: $S_i = S_i \cup \{j\}$
7: **else if** $(j \prec i)$ **then**
8: $n_i = n_i + 1$
9: **end if**
10: **end for**
11: **if** $(n_i = 0)$ **then**
12: $i_{rank} = 1$ {i belongs to F_1}
13: $F_1 = F_1 \cup \{j\}$
14: **end if**
15: **end for**
16: $l = 1$ {Front counter}
17: **while** $F_l \neq \phi$ **do**
18: $Q = \phi$ {Next front}
19: **for all** $i \in F_l$ **do**
20: **for all** $j \in S_i$ **do**
21: $n_j = n_j - 1$
22: **if** $(n_j = 0)$ **then**
23: $j_rank = l + 1$
24: $Q = Q \cup \{j\}$
25: **end if**
26: **end for**
27: **end for**
28: $l = l + 1$
29: $F_l = Q$
30: **end while**

Algorithm 2: Fast-non-dominated-sort (Deb et al., 2002).

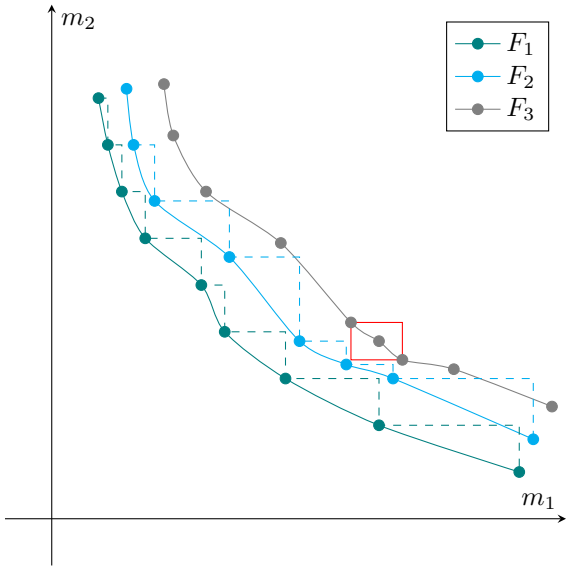


Figure 5.14: Partition of solutions into non-domination fronts.

hyperbox created by the neighbouring solutions in the normalized objective space (illustrated for the two-dimensional case by the red square in Figure 5.14). For a given objective $m \in M$, the crowding distance contribution to individual i is:

$$d_i^m = \frac{|m(i-1) - m(i+1)|}{f_m^{max} - f_m^{min}} \tag{5.8}$$

where $m(i-1)$ denotes the closest *lower* objective value of objective m and $m(i+1)$ denotes the closest *higher* objective value of objective m. The values f_m^{max} and f_m^{min} denotes the highest and lowest objective value of objective m present in the investigated solutions, respectively. The crowding distance for all individuals in a front is most efficiently found with the *crowding-distance-assignment* procedure shown in algorithm 3.

The implemented NSGA-II applies the developed genetic mechanisms in the standard flow of NSGA-II described in (Deb et al., 2002). Pseudocode is given in algorithm 4.

5.4 Hybrid GA for Singleobjective Optimization

From subsection 2.5.5 we have that the maximum spread metric measures the ability of a MOEA to cover the Pareto front with the following formula:

1: **for** $i \in F_l$ **do**
2: $i_{distance} \leftarrow 0$
3: **end for**
4: **for** $m \in M$ **do**
5: sort(F_l, m)
6: $F_l(1) = F_l(|F_l|) = \infty$
7: **for** $i \in (F_l(2), F_l(3), \ldots, F_l(|F_l| - 1))$ **do**
8: $i_{distance} = \frac{|m(i-1) - m(i+1)|}{f_m^{max} - f_m^{min}}$
9: **end for**
10: **end for**

Algorithm 3: Crowding distance assignment (Deb et al., 2002).

1: $t \leftarrow 0$
2: $Q_t \leftarrow \phi$
3: $P_t \leftarrow$ initialize population
4: **while** $Q_t < |P|$ **do**
5: $p_1 \leftarrow binaryTournamentSelection(P_t)$ {Parent 1}
6: $p_2 \leftarrow binaryTournamentSelection(P_t)$ {Parent 2}
7: $Q_t \leftarrow Q_t \cup crossover(p_1, p_2)$
8: **end while**
9: **while** $t < maxGenerations$ **do**
10: $R_t \leftarrow Q_t \cup P_t$
11: $\mathcal{F} \leftarrow fastNonDominationSort(R_t)$
12: $P_{t+1} \leftarrow \phi$
13: $i \leftarrow 1$
14: **while** $|P_t + 1| + |F_i| < |P|$ **do**
15: $crowdingDistanceAssignment(F_i)$
16: $P_{t+1} \leftarrow P_{t+1} \cup F_i$
17: $i \leftarrow i + 1$
18: **end while**
19: $sort(F_i, \prec_p)$
20: $P_{t+1} \leftarrow P_{t+1} \cup F_i[1 : (|P| - |P_{t+1}|]$
21: $Q_{t+1} \leftarrow \phi$
22: **while** $|Q_t| < |P|$ **do**
23: $p_1 \leftarrow binaryTournamentSelection(P_{t+1})$ {Parent 1}
24: $p_2 \leftarrow binaryTournamentSelection(P_{t+1})$ {Parent 2}
25: $Q_{t+1} \leftarrow Q_{t+1} \cup crossover(p_1, p_2)$
26: **end while**
27: mutate(Q_{t+1})
28: $t \leftarrow t + 1$
29: **end while**
30: output Q_{t+1}

Algorithm 4: NSGA-II procedure (Deb et al., 2002).

$$MS = \sqrt{\frac{1}{M}\sum_{m=1}^{M}\left[\frac{min\{X_m^{max}, PF_{true,m}^{max}\} - max\{X_m^{min}, PF_{true,m}^{min}\}}{PF_{true,m}^{max} - PF_{true,m}^{min}}\right]^2} \tag{5.9}$$

where $PF_{true,m}^{max}$ and $PF_{true,m}^{min}$ denotes the highest and lowest value of objective $m \in M$ on the true Pareto front, respectively. This motivates the implementation of a singleobjective algorithm that can locate the minimal possible values of each objective $m \in M$. Because these values are needed only to measure the performance of the SPEA2 and NSGA-II, computational speed is not a major concern. A well-known strategy for balancing exploration and exploitation of the solution space is to hybridize GAs by performing some kind of local search during the evolutionary process. Here, we hybridize a simple GA by performing VND local search on every individual whenever a new population is created from a parent population. We describe the local search algorithm in the following.

5.4.1 Variable Neighbourhood Decent

Our VND uses three neighbourhoods based on the mutation operators presented in subsection 5.1.5. The first neighbourhood, $\mathcal{N}_0(x)$, is defined as the set of solutions obtainable from a solution, x, by swapping the room-days of any two surgeries of the same subspecialty in the chromosome of x. The second neighbourhood, $\mathcal{N}_1(x)$, consists of all solutions like x where two integers in the precedence vector of x have been swapped. The third neighbourhood, $\mathcal{N}_2(x)$, include all solutions where a single surgery in x is assigned another feasible room-day. The procedure starts from an initial solution, x, and investigate all neighbourhoods, $\mathcal{N}_k \in \mathcal{N}$, as long as improvement is found in any of the neighbourhoods. The solution, x, is updated to the new best found solution in a neighbourhood, so that this newly found solution is the basis for generating and searching new neighbourhoods. The entire neighbourhood is searched, so that the best neighbouring solution is selected as the new starting point (steepest decent). The VND local search procedure, for a single objective, m, is outlined with pseudocode in algorithm 5.

5.4.2 Hybridizing a Simple GA

The hybrid GA follows the standard procedure of simple GAs, but invokes the VND local search procedure on every individual in a newly created generation. In order to balance exploration and exploitation of the solution space, we allow only K iterations in the VND procedure. We preserve the best known individual and copy it directly from one generation to the next (elitism). The hybrid GA is outlined in algorithm 6.

$x'' \leftarrow$ a feasible solution
$improvement \leftarrow$ True
$iteration \leftarrow 0$
while $improvement \ \& \ iteration < maxiterations$ **do**
  $x \leftarrow x''$
  $improvement \leftarrow$ False
  for all $\mathcal{N}_k \in \mathcal{N}$ **do**
    $x' \leftarrow \text{argmin}_{y \in \mathcal{N}_k(x)} f_m(y)$
    if $f_m(x') < f_m(x'')$ **then**
      $x'' \leftarrow x'$
      $improvement \leftarrow$ True
    end if
  end for
  $iteration \leftarrow iteration + 1$
end while

Algorithm 5: Variable neighbourhood decent.

$sampleDurations$
$P_0 \leftarrow$ initialize population
$t \leftarrow 0$
while $t < maxGenerations$ **do**
  $P_{t+1} \leftarrow \phi$
  $P_{t+1} \leftarrow P_{t+1} \cup eliteIndividual(P_t)$
  while $|P_t| < populationSize$ **do**
    $p_1 \leftarrow tournamentSelection(P_t)$ {Parent 1}
    $p_2 \leftarrow tournamentSelection(P_t)$ {Parent 2}
    $P_{t+1} \leftarrow P_{t+1} \cup crossover(p_1, p_2)$
  end while
  $mutate(P_{t+1})$
  for all $x \in P_{t+1}$ **do**
    $x \leftarrow VND(x, K)$ {Improves x with the VND procedure over K iterations}
  end for
  $sampleDurations$
  $fitnessAssignment$
  $t \leftarrow t + 1$
end while

Algorithm 6: Hybrid GA.

Chapter 6

Experimental Results

In this chapter we experiment with the algorithms presented in Chapter 5 applied to the surgery admission planning problem modelled in Chapter 4. We start by describing the instances we experiment with and the experimental setup. Next, we perform an initial analysis to argue that its appropriate to use the composite of objectives that we do, and that uncertainty is paramount during surgery scheduling optimization. Then, we compare the two MOEAs using the Wilcoxon signed-rank test on a selection of performance measures that do not require the true Pareto front. We turn to one of the most important contributions of this theses: incorporation of uncertainty, and investigate how the number of Monte-Carlo simulations used during fitness evaluation affects the multiobjective performance of the algorithms. We then compare solution sets provided by deterministic and stochastic versions of the SPEA2 and NSGA-II. Finally, we discuss our findings and give our recommendations for future work.

6.1 Test Data

In this section we describe how instances are created based on the surgery scheduling situation in a real life Norwegian hospital. In summary:

Surgery scheduling instances. *Our instances range in size from 2 rooms $\times$ 5 days with 34 patients on the waiting list to 8 rooms $\times$ 5 days with 255 patinets on the waiting list.*

Lognromal distributions. *Surgery durations are modelled with procedure dependent log-normal distributions where the mean is based on estimations used by practitioners, and the coefficient of variance is set to $c_v = 0.1$.*

Instances are modified from (Nyman, 2017) which considers the same hospital as we do here. Expected durations are assigned to surgeries according to a guideline in current use

at the hospital, which states, for every surgical procedure, how much time to allocate in the operating rooms. We ensure that the distribution of patients on the patient waiting list matches, to a reasonable extent, the capacity reserved in the master surgery schedule. Although the data used is not based on actual patient waiting lists and actual durations, we believe that the approximation is realistic enough for our analysis to be generalized to actual surgery scheduling settings.

6.1.1 Parameters

We use patient lists ranging from $N = 34$ to $N = 255$ patients to be scheduled in either 2 operating rooms over 5 days, 7 operating rooms over 3 days or 8 operating rooms over 5 days. For all instances, we set the same parameters for opening hours and allowed scheduled overtime so that $E^b_{dr} = E^b_{dk} = 0$, $E^e_{dr} = E^e_{dk} = 8$ hours and $V = 2$ hours. The considered hospital uses a weekly master surgery schedule to schedule surgeries of 7 different subspecialties (Local, Reconstructive, Plastic, Hand, Arthroscopy, Back and Prosthesis) into 8 operating rooms. The master surgery schedule is depicted in Figure 6.1. The instances we experiment with consider subsections of the master surgery schedule (indicated by colors and brackets), with the largest instance class considering the entire schedule. The number of surgeries and surgeons is set to match the number of operating rooms and days involved in each instance. We describe the instances thoroughly in the next section.

6.1.2 Instance Generation

We categorize nine instances into three different classes based on the number of room-day combinations considered. *Small* instances, I_1, consider schedules with 2 operating rooms in a planning period of 5 days ($2 \times 5 = 10$ room-day combinations). As can be seen in Figure 6.1, this category include 3 subspecialties: Hand, Arthroscopy and Plastic. *Medium* instances, I_2, include 7 operating rooms over 3 days ($7 \times 3 = 21$ room-day combinations) and involve five subspecialties: Reconstructive, Plastic, Arthroscopy, Back, Hand and Prosthesis. *Large* instances, I_3, include all 8 operating rooms over a weekly period ($8 \times 5 = 40$ room-day combinations) and distinguish between all 7 subspecialties.

For each instance category, I_1, I_2 and I_3, three different cases of patient lists are considered:

(i) the case where the load on the patient list matches the available resources well,

(ii) the case where the load on the patient list is approximately 50% larger than the available capacity, and

(iii) the case where the load on the patient waiting list matches the capacity available over two periods.

[1] in the original master surgery schedule, this operating room is either closed or open for various subspecialties. The aspect of closed or open operating rooms is omitted for simplicity. A subspecialty is selected for the room so as to complete the master surgery schedule.

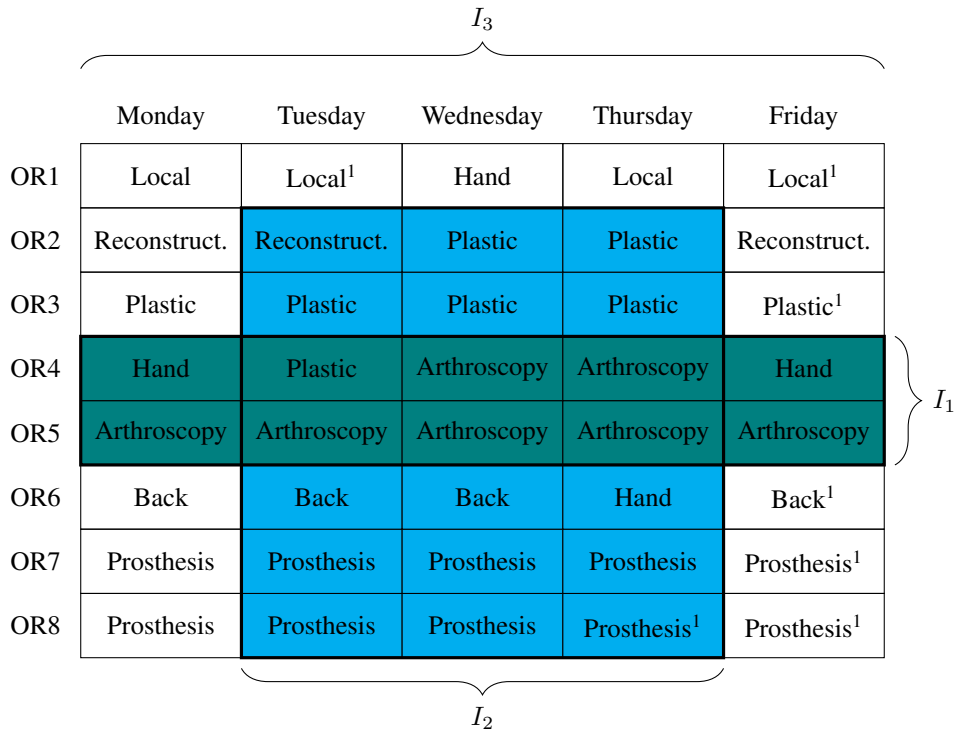


Figure 6.1: Instance classes created from a real-life master surgery schedule. The brackets and colors indicate the three different instance classes we experiment with. Note that the overlapping section in the middle is included in both instance class I_1 and I_2.

This distinction within each category results in 9 different instances, summarized in Table 6.1.

Instance	D	R	DxR	K	N
$I_1^{(1)}$	5	2	10	3	34
$I_1^{(2)}$	5	2	10	3	49
$I_1^{(3)}$	5	2	10	3	64
$I_2^{(1)}$	3	7	21	9	64
$I_2^{(2)}$	3	7	21	9	93
$I_2^{(3)}$	3	7	21	9	127
$I_3^{(1)}$	5	8	40	10	129
$I_3^{(2)}$	5	8	40	10	194
$I_3^{(3)}$	5	8	40	10	255

Table 6.1: Instances. Nine different instances experimented with in this thesis.

6.1.3 Surgery Properties

All surgeries are assigned a referral date, H_i, and a deadline, G_i. The first day in the planning period is defined to be day 0, the referral date, H_i, is the defined as a number of days *backward* in time and the deadline, G_i, is defined as a number of days *forward* in time. Referral dates and deadlines are set somewhat arbitrarily within a reasonable range of 0 to 20 days. The estimated duration of surgeries, $\mathcal{D}_i$, is based on a document schedulers in the hospital use when scheduling specific surgical procedures. From (Nyman, 2017), the instances are prepared for a discrete time model and duration is given in a number of periods. We convert these durations by multiplying the number of periods with the period length. Because we do not posses actual data, we assume that durations follow a lognormal distribution, with the estimated duration as expected value, μ, and two different cases for the standard deviation:

(i) $\sigma \sim 0$[2], i.e. the deterministic case, and

(ii) $\sigma = c_v \mu$ where c_v is the coefficient of variance that we set to 0.1.

6.2 Experimental Setup

In this section we justify our parameter settings. Our setup can be highlighted as follows:

Parameter settings. *The SPEA2 and NSGA-II use population size of 500, with 250 generations as the stopping criterion, binary tournament selection and 30 Monte Carlo simulation runs.*

[2]the lognormal distribution is not defined for $\sigma = 0$.

All test are run on an Asus laptop with a 0.80 GHz Intel multi core processor with 8 GB of RAM.

6.2.1 Calibration of Hyperparameters

The calibration of hyperparameters is partly based on experiences from (Nyman, 2017), partly on configurations found in literature and partly on preliminary experimentation with the developed algorithms. Comprehensive and systematic hyperparameter calibration is out of the scope of this thesis. The selected values for the SPEA2 and NSGA-II are justified in the following.

Inspired by the population and archive sizes chosen in (Zitzler et al., 2001) (250 for $m = 2$; 300 for $m = 3$ and 400 for $m = 4$), we set the SPEA2 population size and archive size to 500, as in our case, $m = 5$. The NSGA-II population is also set to 500. The stop criterion, i.e. the number of generations, is set to 250, as in (Yen and He, 2014), because we observed convergence at this point in our preliminary experimentation. The k-value in the SPEA2 is conventionally set to $\sqrt{populationsize + archivesize}$, which in our case is rounded to $k = 32$. We use binary tournament selection as in the original SPEA2 and NSGA-II, and select the tournament winner for reproduction with a probability of 90%. From previous experiences with the same problem in (Nyman, 2017) we chose a mutation rate of 20%. Note that we have an unconventional interpretation of the mutation rate, where the rate indicates the probability of an *individual*, and not a single gene in a chromosome, being selected for random mutation. Motivated by a common interpretation of the central limit theorem [3], we use 30 Monte-Carlo simulation runs to estimate the fitness of an individual. Inspired by (Gul et al., 2011), we use the 65$^{\text{th}}$ percentile to decide scheduled starting times for surgeries during chromosome decoding. In our baseline setting, we use $T = \infty$, to infer the assumption that all resources are available, even in cases where surgeries are moved forward in time.

6.3 Preliminary Analysis

In this section we elaborate the following statements.

Solution properties. *The solution to the presented problem is a finite set of 5-dimensional hypercubes whose side lengths are governed by random variables.*

The objectives are partially conflicting. *Each objective is in partial conflict with all other objectives.*

Performance measures that drive exclusion of surgeries. *The overtime objectives, O_O^R and O_O^K, are monotonically increasing with the collective increase of all surgery du-*

[3] in practical applications, it is common to assume normal distribution of the sample mean regardless of the sample distribution when the sample size is larger than 30 (ROC, 2006). By using this number of simulations, we have some control of the variability of the average fitness used during evolution.

rations. Together with the surgeon idle time measure, O_I^K, they drive the exclusion of surgeries from the schedule.

Performance measures that drive inclusion of surgeries. *The idle time objectives, O_I^R and O_I^K, are monotonically decreasing with the collective increase of all surgery durations. The waiting time objective, O_W, is independent of surgery durations. The waiting time objective, together with the operating room idle time objective, O_I^R, drive the inclusion of surgeries in the schedule.*

6.3.1 Solution Space and Pareto Front Properties

We include some comments here on the properties of the solution space and the Pareto front. As there are five objectives, the solution space is five dimensional, and in the deterministic case, the Pareto-front is a five dimensional hyperplane that separates dominated solutions on one side from infeasible solutions on the other. When duration uncertainty is taken into account, the Pareto front may be described in a various ways. One way is to use the efficient solutions (i.e. Monte-Carlo integration in our case, see Equation 5.3) to describe a hyperplane just as in the deterministic case. A more accurate description, is to describe the Pareto front as a finite set of five-dimensional hypercubes, where the length of each side can be described as a random variable. Each hypercube represents the possible values of the five objectives for all realizations of surgery durations. We discuss the nature of the hypercube lengths in the next paragraph.

Observing the equations describing the five objectives, O_W (Equation 4.14), O_O^R (Equation 4.18), O_O^K (Equation 4.19), O_I^R (Equation 4.20) and O_I^K (Equation 4.21) we can make the following statements about how the hypercube sides vary with the lognormally distributed surgery durations, $\mathcal{D}(\omega)$. The waiting time, O_W, is *independent* of $\mathcal{D}(\omega)$, and thus, the side length along the waiting time objective axis is always 0. The overtime objectives, O_O^R and O_O^K, are monotonically *increasing* for situations where *all* surgery durations increase. This can be seen, as the durations contribute positively to the objectives ($max\{t_i(\omega)x_{idr} - E_{dr}^e, 0\}$ for O_O^R; and $max\{t_i(\omega)z_{idk} - E_{dk}^e, 0\}$ for O_O^R), with $t_i(\omega)$ being the starting times deducted from the durations, $\mathcal{D}(\omega)$. The idle time objectives, O_I^R and O_I^K, are monotonically *decreasing* for situations where *all* surgery durations increase, as the durations contribute negatively to the idle time objectives ($-\sum_{i=1}^{N} \mathcal{D}_i(\omega)x_{idr}$ for O_I^R; and $-\sum_{i=1}^{N} \mathcal{D}_i(\omega)z_{idk}$ for O_I^K). We do not attempt to extend this basic understanding of how the objective values of found solutions vary with surgery durations, and in our further analysis, we will use nonparamteric methods to assess the quality of solutions.

6.3.2 Degree of Conflict Between Objectives

As discussed in subsection 2.5.1, mutliobjective optimization makes sense when desired properties of the sought solution are in partial conflict. In this subsection, we investigate how the considered performance measures relate to each other and argue that they are

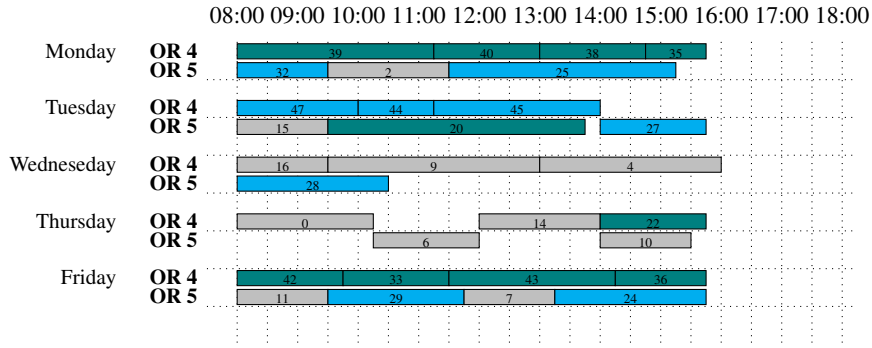


Figure 6.2: Optimized operating room overtime for $I_1^{(2)}$: $\{O_O^R = 0, O_I^R = 975, O_O^K = 0, O_I^K = 345, O_W = 59.29\}$

indeed partially conflicting. Our analysis is based on the hybrid GA applied to the deterministic problem, as this simplifies the analysis without loss of generality. As an example, we use instance $I_1^{(2)}$ and illustrate the schedules produced by performing variations of singleobjective optimization.

A solution with 0 operating room overtime is shown in Figure 6.2. All surgeries end before 16:00 which is the end of normal opening hours. There is a lot of idle time, both in operating rooms (e.g. after surgery 28 in OR 5 on Wednesday) and for surgeons (e.g. for the blue surgeon between surgery 32 and surgery 25 in OR 5 on Monday). The amount of idle time also leads to a high patient waiting cost, $O_W = 59.29$ (the best found waiting cost for this instance is $O_W = 24.66$). However, the surgeon overtime is also 0, as might be expected when the surgeon and operating room opening hours are identical. It might seem that operating room overtime, O_O^R, and surgeon overtime, O_O^K, are non-conflicting. But, from Figure 6.3 and Figure 6.4 showing schedules with 0 operating room idle time and 0 surgeon idle time respectively, we see examples of two cases where these two measures are not overlapping:
(i) $O_O^R > O_O^K$ and
(ii) $O_O^R < O_O^K$.
The first case happens when a surgeon finishes a surgery in an operating room after normal opening hours, and then starts a new surgery in *another operating room*. Overtime is then incurred in two operating rooms, but only for one surgeon. The opposite happens in the second case when a surgeon finishes a surgery in an operating room after normal opening hours before *another surgeon* starts a new surgery *in the same operating room*. Overtime is then incurred for two surgeons, but only for one operating room. Situations can occur where a choice must be made between scheduling two surgeons to work overtime in the same operating room or reserving overtime capacity for a single surgeon in two operating rooms. Thus, the objectives are partially conflicting. However, these situations are not common in desirable schedules because it requires a surgery to start after normal opening hours. Our main reason for separating the two objectives is two acknowledge that they represent two different concepts, and prepare the model and algorithms for real life situations

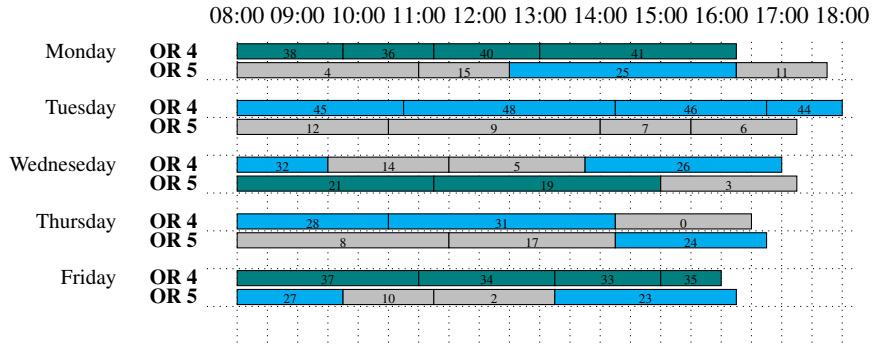


Figure 6.3: Optimized operating room idle time for $I_1^{(2)}$: $\{O_O^R = 540, O_I^R = 0, O_O^K = 555, O_I^K = 765, O_W = 38.32\}$

where surgeon and operating room opening hours often are different.

From Figure 6.4, showing a schedule with 0 surgeon idle time, it seems that optimizing surgeon idle time is uncorrelated with optimization of the other objectives. From an analytic perspective however, there is in fact some correlation between surgeon idle time, surgeon overtime and operating room overtime. To see this, it must be acknowledged that the *empty schedule* is a feasible solution to the analyzed problem. If there are no surgeries in the schedule, there is no surgeon waiting time in between surgeries, and no overtime can be incurred at the end of the day. Thus, these performance measures actually *drive the exclusion of surgeries*. These measures contrast the operating room idle time performance measure which, when optimized, overfills the schedule in Figure 6.3 and the waiting time measure which, when optimized, overfills the schedule in Figure 6.5. The operating room idle time measure and waiting cost measure *drive the inclusion of surgeries*. However, they are still partially conflicting, as minimizing operating room idle time does not guarantee prioritization of waiting patients, and minimizing the waiting time measure cannot guarantee the absence of idle time in between surgeries.

Based on the above analysis we state that all of the five objectives are partially conflicting. This does not mean that all of them must be included in a multiobjective surgery scheduling problem. Choosing appropriate performance measures is a task for hospital managers. But we can conclude that optimizing surgeon idle time, surgeon overtime or operating room overtime on one hand without optimizing operating room idle time or waiting cost on the other, inevitably would lead bad quality schedules[4].

6.3.3 The Effect of Uncertainty

As discussed in subsection 3.1.3, several papers balance multiple objective during surgery scheduling optimization by using the weighted average approach. We illustrate how a

[4] in our model, the empty schedule is feasible, and substantial overtime is allowed. Constraints can be added to the model so that it makes sense to optimize a single objective.

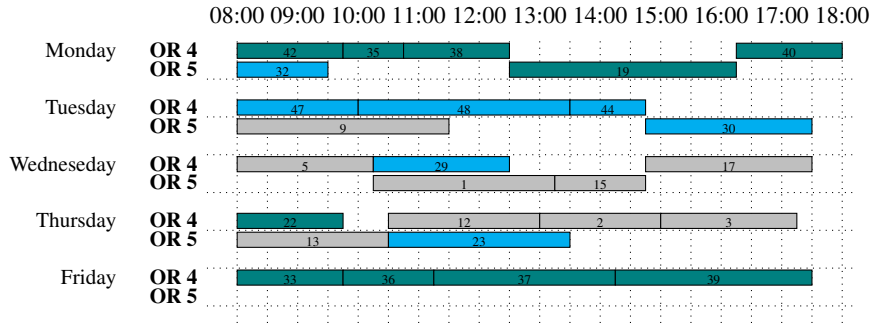


Figure 6.4: Optimized surgeon idle time for $I_1^{(2)}$: $\{O_O^R = 32, O_I^R = 113, O_O^K = 31, O_I^K = 0, O_W = 70.65\}$

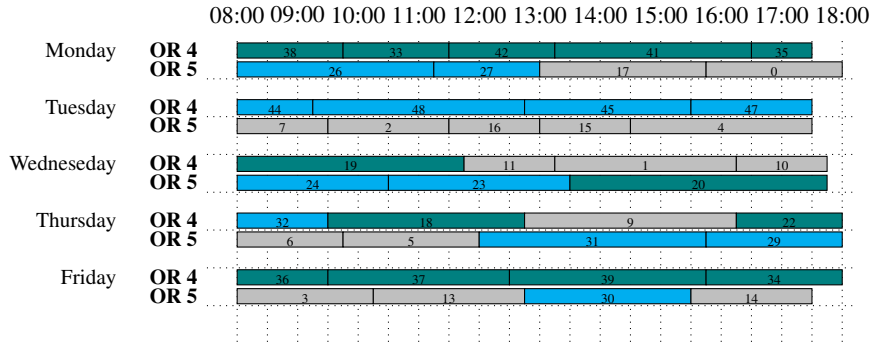


Figure 6.5: Optimized patient waiting time for $I_1^{(2)}$: $\{O_O^R = 1050, O_I^R = 0, O_O^K = 1065, O_I^K = 675, O_W = 24.66\}$

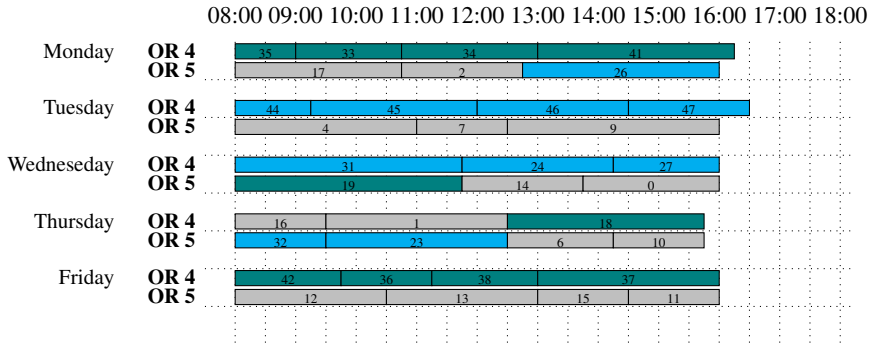


Figure 6.6: Optimized weighted sum for $I_1^{(2)}$: $\{O_O^R = 45, O_I^R = 30, O_O^K = 45, O_I^K = 0, O_W = 36.76\}$

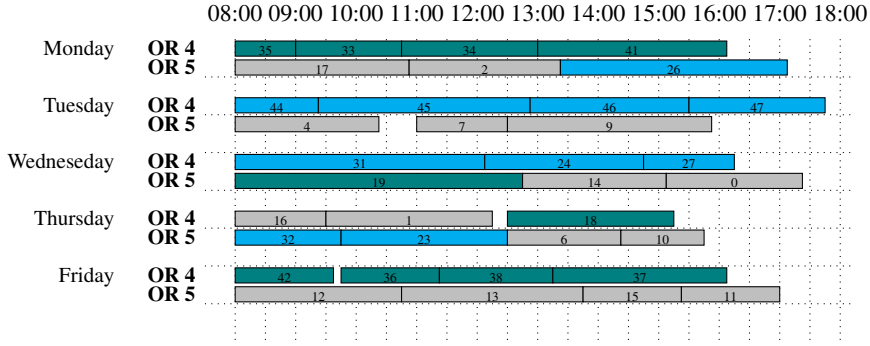


Figure 6.7: Realized schedule of the schedule in Figure 6.6 with a coefficient of variance of 0.1.

weighted average approach using expected durations leads to an apparently efficient schedule in Figure 6.6. Exposed to uncertainty, however, under the constriction that no surgery can start earlier than scheduled, the schedule quality deteriorates. The result is shown in Figure 6.7. Substantial overtime is incurred, e.g. in OR 4 on Tuesday and OR 5 on Wednesday. Comparing with the planned schedule, it is evident that some of the overtime is propagated from delay earlier in the day: surgery 45 and surgery 19 contribute to most of the delay in OR 4 on Tuesday and OR 5 on Wednesday, respectively. By simulating these effects during optimization, schedules where there is risk of such propagation is mitigated, e.g. schedules where durable surgeries are placed at the end of the day, are preferred. Thus, we are convinced that proper surgery scheduling algorithms should consider duration uncertainty during the search process.

6.4 Comparison of SPEA2 and NSGA-II

In this section, we compare the two implemented MOEAs by wide set of performance measures:

(i) Pareto-optimality is assessed by the ratio of nondominated individuals, overall nondominated vector generation and the pareto dominance indicator,

(ii) distribution of solutions is measured by uniform distribution and the number of distinct choices, and

(iii) maximum spread is approximated using the singlobjective Hybrid GA.

The main takeaway is as follows:

The SPEA2 does not perform better than the NSGA-II. *Contrary to what might be expected, the NSGA-II outperforms the SPEA2 on important performance measures such as set coverage and pareto dominance indicator on the larger instances.*

Due to the limited time available for testing, only three instances are considered in this analysis. We consider the hyperarea metric to be too comprehensive and out of the scope of this theses[5]. Because the true Pareto front is not known for the investigated instances, we cannot measure the error ratio, generational distance, inverted generational distance or maximum Pareto front error. In other words, we lack measures of convergence. We address the need of benchmark instances with known true Pareto fronts in the future research section at the end of this chapter.

6.4.1 Wilcoxon Signed-Rank Test Results

Here, we compare the overall nondominated vector generation (ONVG), ratio of nondominated individuals (RNI), Pareto dominance indicator (NR), uniform distribution (UD), number of distinct choices (NDC), maximum spread and set coverage metrics of SPEA2 and NSGA-II for three selected instances, $I_1^{(1)}$, $I_2^{(2)}$ and $I_3^{(3)}$. For the readers convenience, we summarize the applied measures (already detailed in subsection 2.5.5) in Table 6.2. Results for the smallest instance, $I_1^{(1)}$, are based on 30 computation runs, and results for the medium instance, $I_2^{(2)}$, and large instance, $I_3^{(3)}$, are based on 10 computation runs. We conduct the Wilcoxon signed-rank test, detailed in section 2.6, using critical values shown in table (Company, 1964). Both 10 and 30 runs are sufficient for achieving results of statistical significance. However, the number of problem instances included in this analysis are too few to separate random applicability relations between algorithm and instance from general performance.

In Table 6.4 the comparison results are shown with the following notation:

(i) if H_0 cannot be rejected with significance level, $\alpha < 0.05$, we write SPEA2 $\sim$ NSGA-II, which indicates similar performance of the two algorithms,

(ii) if H_0 can be rejected so that SPEA2 outperforms NSGA-II, we write SPEA2 $\succ$

[5]for the considered problem, the union of five-dimensional hypervolumes must be computed in order to measure hyperarea. A fast algorithm for computing the hyperarea metric for up to 10 objectives is presented in (While et al., 2006).

Table 6.2: Summary of selected performance measures. Below the formula for each performance measure, we indicate the possible values that the measures can take. The best value is marked with a star (*).

RNI	ONVG	NR	UD
$\frac{\lvert\overline{X}\rvert}{n}$	$\lvert PF_{known}\rvert$	$\frac{\lvert A_1 \cap B\rvert}{\lvert B\rvert}$	$\frac{1}{1+S_{nc}}$
$\in [0, 1^*]$	$\in \mathcal{Z}^{\geq}, large^*$	$\in [0, 1^*]$	$\in [0, 1^*]$

Set Coverage	NDC
$\frac{\lvert x_2 \in X_2; \exists x_1 \in X_1 : x_1 \preceq x_2\rvert}{\lvert X_2\rvert}$	$\sum_{l_m}^{(1/\mu)-1} \cdots \sum_{l_2}^{(1/\mu)-1} \sum_{l_1}^{(1/\mu)-1} NT_\mu(q, X)$
$\in [0, 1^*]$	$\in \mathcal{Z}^{\geq}, large^*$

Maximum Spread

$$\sqrt{\frac{1}{M}\sum_{m=1}^{M}\left[\frac{min\left\{X_m^{max}, PF_{true,m}^{max}\right\} - max\left\{X_m^{min}, PF_{true,m}^{min}\right\}}{PF_{true,m}^{max} - PF_{true,m}^{min}}\right]^2}$$

$\in [0, 1^*]$

Table 6.3: Critical values for the Wilcoxon signed rank test for sample sizes $n = 10$ and $n = 30$.

	α			
n	0.01	0.02	0.05	0.10
10	3	5	8	11
30	109	120	137	152

Table 6.4: Comparison of SPEA2 and NSGA-II for a selection of instances. The results of the Wilcoxon signed rank test over 30 computation runs.

		$I_1^{(1)}$	$I_2^{(2)}$	$I_3^{(3)}$
Pareto-optimality	ONVG	SPEA2 $\sim$ NSGA-II	SPEA2 $\succ$ NSGA-II	SPEA2 $\succ$ NSGA-II
		–	$p < 0.01$	$p < 0.01$
	RNI	SPEA2 $\sim$ NSGA-II	SPEA2 $\succ$ NSGA-II	SPEA2 $\succ$ NSGA-II
		–	$p < 0.01$	$p < 0.01$
	NR	SPEA2 $\succ$ NSGA-II	SPEA2 $\prec$ NSGA-II	SPEA2 $\prec$ NSGA-II
		$p < 0.01$	$p < 0.01$	$p < 0.01$
Distribution	UD	SPEA2 $\prec$ NSGA-II	SPEA2 $\prec$ NSGA-II	SPEA2 $\prec$ NSGA-II
		$p < 0.01$	$p < 0.01$	$p < 0.01$
	$\text{NDC}_{0.2}$	SPEA2 $\prec$ NSGA-II	SPEA2 $\sim$ NSGA-II	SPEA2 $\sim$ NSGA-II
		$p < 0.01$	–	–
Maximum Spread		SPEA2 $\sim$ NSGA-II	SPEA2 $\sim$ NSGA-II	SPEA2 $\succ$ NSGA-II
		–	–	$p < 0.02$
Set Coverage		SPEA2 $\sim$ NSGA-II	SPEA2 $\prec$ NSGA-II	SPEA2 $\prec$ NSGA-II
		–	$p < 0.01$	$p < 0.01$

NSGA-II,

(iii) if H_0 can be rejected so that NSGA-II outperforms SPEA2, we write SPEA2 $\prec$ NSGA-II.

We also indicate the p-value whenever H_0 is rejected. The only conclusion that is consistent for all instances is the uniform distribution measure, which indicates that the NSGA-II yields approximation fronts with less clustered solutions. From looking at actual solutions provided by the two algorithms, it seems that replicates of solutions are kept in the archive of SPEA2, which can explain why NSGA-II outperforms SPEA2 with regards to this metric. The most interesting result from Table 6.4, however, is the fact that NSGA-II outperforms SPEA2 on the larger instances, $I_2^{(2)}$ and $I_3^{(3)}$, as measured by the set coverage metric and the Pareto dominance indicator. This is contrary to what is generally reported in literature, namely that the SPEA2 usually outperforms the NSGA-II when more than two objectives are optimized (Yen and He, 2014). This result indicates that this might not be the case when the fitness value of individuals are uncertain.

6.5 Dealing With Uncertainty

This section investigates, by examples, the effect of incorporating uncertainty during evolutionary search for high quality schedules. Our main findings are:

The computation time increases linearly with the number of Monte-Carlo simulations. *The simulation procedure is fast, and moderate increase in computation time is incurred by increasing the number of simulations.*

The multiobjective spread increases with the number of Monte-Carlo simulations. *In several examples, we see an increase in the objective spread as the simulation number increases. We hypothesize that solutions with high expected value are risky.*

Overtime is significantly reduced by incorporating uncertainty. *We detect an average overtime decrease of 38.10% when uncertainty is incorporated together with hedging.*

6.5.1 Explicit Averaging

As discussed in section 2.7.4, an intuitive way of handling uncertainty during multiobjective evolutionary optimization is explicit averaging. With this approach, the fitness of an individual is estimated using Monte-Carlo simulation. An important decision when using this approach is the number of simulation runs used to estimate the fitness. There is a trade off between the accuracy of estimation and the computation cost associated with each simulation run. In Figure 6.8 we show the result of running the NSGA-II and SPEA2 on $I_1^{(1)}$, $I_1^{(2)}$ and $I_1^{(3)}$ with three different choices for the number of simulation runs involved in each fitness evaluation:

(i) a single simulation run,
(ii) 5 simulation runs,
(iii) 30 simulation runs, and
(iii) 100 simulation runs.

In each case, the algorithms are run with otherwise standard settings for the hyperparameters. Each individual in the resulting population is evaluated with 100 simulation runs and the objectives are expressed in terms of boxplots showing the 2$^{\text{nd}}$ percentile, the 1$^{\text{st}}$ quantile, the median, the 3$^{\text{rd}}$ quantile and the 98$^{\text{th}}$ percentile of objective values present in the population. We find, by brief experimentation with the Shapiro-Wilk normality test (Shapiro and Wilk, 1965)[6], that the objectives represented in the population or archive are generally not normally distributed. One explanation for this is frequently encounterd lower bounds, e.g. 0 operating room overtime, 0 surgeon overtime or 0 surgeon idle time. We therefore base our percentiles and quantiles on the following nonparamteric approach. All objectives, $m \in M$, are sorted in increasing order in M separate lists, A_m. Next, any percentile, ρ, of objective $m \in M$ can be estimated by taking the $||(\rho * |A_m|)||^{\text{th}}$ element from A_m where $||x||$ denotes the nearest integer of x, and $|A_m|$ is the size of the sorted list. The computation time needed to solve each instance is indicated by the orange line referring to the values on the left y-axis in Figure 6.8.

First, an interesting takeaway from the results in Figure 6.8 is that there seems to be a linear relationship between the number of simulations and computation time. A simple

[6] conducted online (Dittami, 2009)

Table 6.5: Estimate of computation time needed for implicit averaging.

Problem	50 individuals	200 individuals	500 individuals	1000 individuals
NSGA2 on $I_1^{(1)}$ (a)	0.57	2.6	8.98	27.33
SPEA2 on $I_1^{(1)}$ (b)	0.28	1.53	7.76	28.41
NSGA2 on $I_1^{(2)}$ (c)	0.78	3.53	11.25	31.26
SPEA2 on $I_1^{(2)}$ (d)	0.33	1.83	7.9	29.28
NSGA2 on $I_1^{(3)}$ (e)	1.03	4.45	13.63	35.33
SPEA2 on $I_1^{(3)}$ (f)	0.43	2.23	8.98	31.23

linear regression analysis conducted in MS Excel gave the following R^2-values and associated gradients, Δ, in the fitted linear function for the computation time in subfigures (a)-(f): $\{R^2 = 1, \Delta = 0.1136\}, \{R^2 = 0.9755, \Delta = 0.0634\}, \{R^2 = 0.9999, \Delta = 0.126\}, \{R^2 = 0.9965, \Delta = 0.1027\}, \{R^2 = 0.9999, \Delta = 0.11369\}, \{R^2 = 0.997, \Delta = 0.1082\}$. Thus, a linear relationship seems likely (although we make no claim about statistical significance in this brief analysis), and using the average gradient, $\overline{\Delta} = 0.1046$ may serve as an estimate for the marginal computation cost of a simulation (for this instance category and these hyperparameters). To exemplify, we would approximate the added computational cost of using 50 simulation runs to estimate fitness to be 50 runs $\times$ 0.1046 minutes/run $=$ 5 minutes and 14 seconds. The described linear relationship is not surprising, as increasing the number of simulation runs only increases the thoroughness of each fitness evaluation, and not the number of fitness evaluations. The computational cost is relatively low because the chromosome under evaluation is only decoded once before simulation is performed to determine starting times, ending times and objectives for different outcomes of surgery durations. The decoding procedure is quite comprehensive, but the simulation procedure is not.

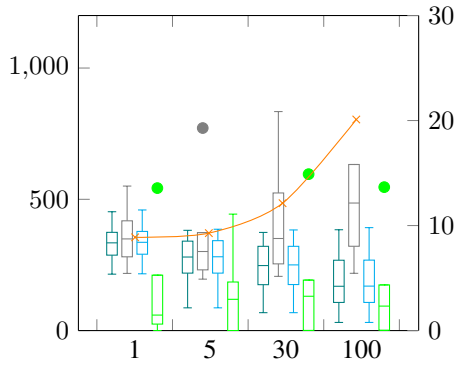
As mentioned in section 2.7.4, an alternative to explicit averaging is implicit averaging. With implicit averaging, random perturbations are done on the design variables of individuals in very large populations to average out peaks and ensure convergence over time. We have not implemented implicit averaging, but we can asses the computational cost of increasing the population size while using only one simulation run to estimate the fitness of an individual. The computation time results for the same instances that we investigated in Figure 6.8 are shown in table Table 6.5. Apparently, the computation time needed increases non-linearly with the number of individuals in the population. Indeed, simple analysis in MS Excel indicates a better polynomial or power fit than linear fit on the computation time as a function of population size. This nonlinear relationship is in line with theory as there are sorting procedures in both the SPEA2 and NSGA2 with nonlinear complexity in relation to population or archive size. Although we cannot compare explicit and implicit averaging without comparing the solution quality of each approach, our assessment is that explicit averaging is more comprehensible, straightforward and less computationally expensive than implicit averaging.

There are two important tendencies to notice regarding how the population improves as the number of simulations increases. First, as might be expected, the operating room and surgeon overtime measures are significantly improved. By improved, we mean that the three lower box plot values, 2nd percentile, the 1st quantile and the median, are lower. Occasionally, this comes at a cost, namely an increase in surgeon and operating room idle time. However, the overtime improvement seems to outweigh the idle time deterioration. Also, in most of the cases, solutions with idle times as low as the 2nd percentile in the 1-scenario population exist. Especially the surgeon idle time performance measure seems well maintained as scenarios increase. This leads us to the second important tendency. The spread in solution space seems to increase with the number of simulations conducted during fitness evaluation. We propose an explanation for this in the following paragraph.

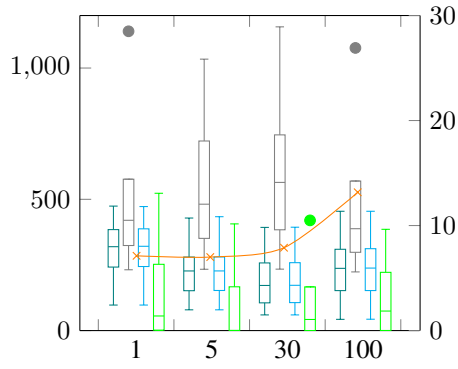
Inaccurate fitness estimation leads to fluctuations in the nondomination relationships among individuals. An individual can be nondominated in one generation and dominated in the next, depending on the particular outcome of surgery durations. In such environments, solutions that are robust against duration variability will be favoured, while risky solutions will not survive occasional unfortunate variations. The underlying assumption for explaining a larger spread as the number of simulation runs increases is as follows. *Increasing expected performance also increases performance variability*. This is a well-known and much applied assumption in e.g. finance where the optimal trade-off between expected return and risk of different stocks outlines the efficient Pareto-front. If only a few scenarios are used for fitness evaluation, solutions that are nondominated on expectation, e.g. solutions with close to 0 overtime without too much idle time, might occasionally be dominated and left out during selection. This line of thought is interesting because it illustrates a major difficulty with multiobjective optimization under uncertainty: that there are actually three notions of multiobjectivity in such problems:

(i) the trade-off among partially conflicting performance measures,
(ii) the trade-off between closeness to, and coverage of, the Pareto front, and
(iii) the trade-off between expected performance and performance variability.

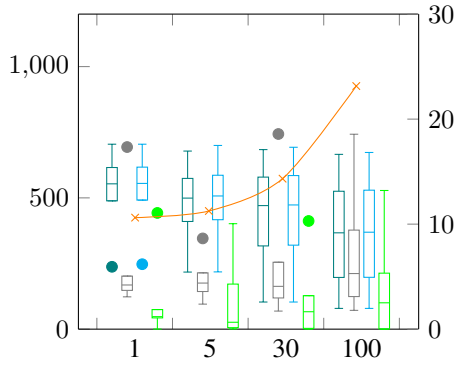
We may hypothesize that the overtime improvement seen in Figure 6.8 as the number of scenarios increase comes at the cost of increased performance variability: an issue that we have not covered in this thesis. The danger is, paradoxically, that increasing the number of scenarios in pursuit of robust solutions might actually cause risky solutions to dominate interestingly robust solutions. We do not elaborate on this interesting question here in order to manage the thesis scope.



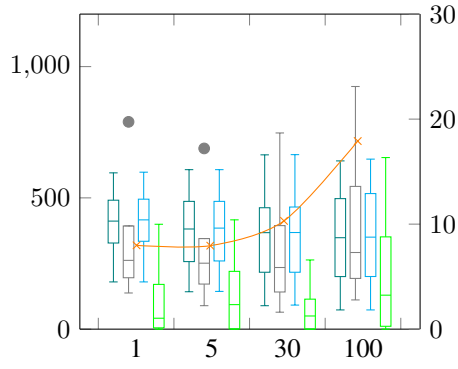
(a) NSGA2 on $I_1^{(1)}$.



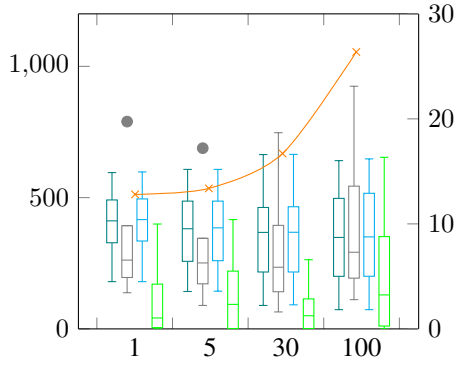
(b) SPEA2 on $I_1^{(1)}$..



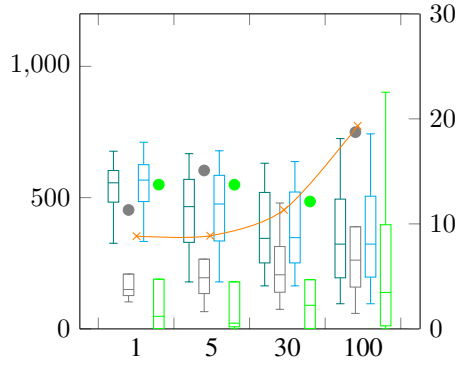
(c) NSGA2 on $I_1^{(2)}$.



(d) SPEA2 on $I_1^{(2)}$.



(e) NSGA2 on $I_1^{(3)}$.

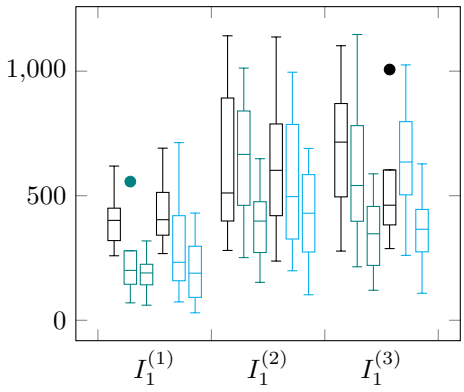


(f) SPEA2 on $I_1^{(3)}$.

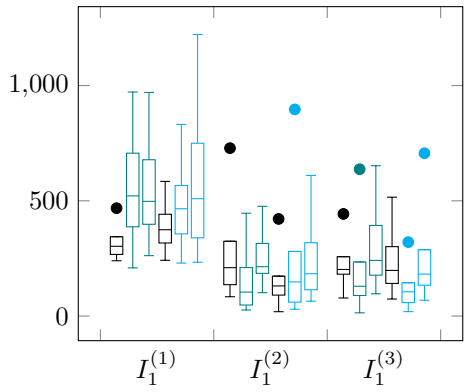
Figure 6.8: Explicit averaging for instance $I_1^{(1)}$, $I_1^{(2)}$ and $I_1^{(3)}$. Box plots indicate, from left to right, operating room overtime, operating room idle time, surgeon overtime and surgeon idle time. The left y-axis indicates performance in minutes. The right y-axis indicates the computation time in minutes.

6.5.2 Comparison with a Deterministic Approach

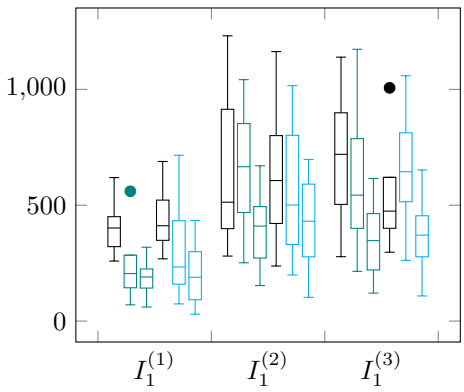
In order to get an impression of how incorporating uncertainty can benefit the decision maker, we compare solutions found by the developed SPEA2 and NSGA-II with their deterministic counterparts. The deterministic versions base the search process on expected values for the surgery durations. Note that this is not the same as using a single simulation run for fitness evaluation as the expected value is used both for schedule construction and fitness evaluation. The resulting solution set is then evaluated assuming uncertain surgery durations and compared against the algorithms that incorporate uncertainty during evolution. In order to differntiate between improvements gained by incorporating uncertainty and imporvements gained from hedging, we use two different values for the scheduling confidence in the SPEA2 and NSGA 2: the expected value and the 65$^{\text{th}}$ percentile. Results for $I_1^{(1)}$, $I_1^{(2)}$ and $I_1^{(3)}$ are shown in Figure 6.9.



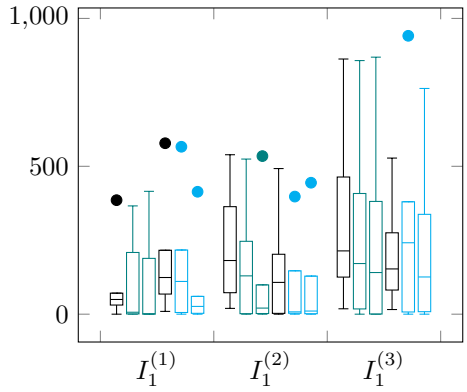
(a) Operating room overtime.



(b) Operating room idle time.



(c) Surgeon overtime.



(d) Surgeon idle time

Figure 6.9: Deterministic vs. stochastic solutions. Box plots indicate, from left to right, SPEA2 deterministic, SPEA2 without hedging, SPEA2 with hedging, NSGA-II deterministic, NSGA-II without hedging and NSGA-II with hedging. The y-axis indicates performance in minutes.

In general, the overtime performance measures are substantially improved, especially when combining uncertainty incorporation and hedging. To illustrate, the mean operating room overtime is reduced from 401 minutes to 190 minutes for $I_1^{(1)}$ with SPEA2; 403 to 189 minutes for $I_1^{(1)}$ with NSGA2; 510 minutes to 398 minutes for $I_1^{(2)}$ with SPEA2; 601 to 429 minutes for $I_1^{(2)}$ with NSGA2; 715 minutes to 347 minutes for $I_1^{(3)}$ with SPEA2 and 461 to 365 minutes for $I_1^{(3)}$. This is an average overtime decrease of 38.10 %. Usually, there is improvement without hedging as well, but this is not consistent. We expect this to be due to random performance variations of the algorithms; a reminder that this is only anecdotal evidence and that thorough statistical analysis should be conducted in the future.

The waiting cost for deterministic, uncertainty incorporating and hedging versions of the

algorithms is shown in **??**. It is not a surprise that uncertainty incorporation and hedging induces more waiting cost for patients, as the produced schedules should be less dense to avoid overtime. The increase in waiting cost from $I_1^{(1)}$ to $I_1^{(2)}$ to $I_1^{(3)}$ is due to the larger number of patients on the patient waiting lists associated with these instances.

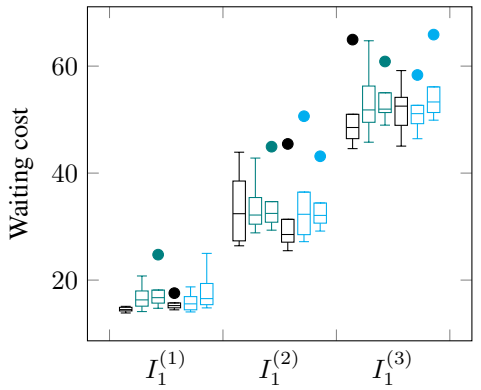


Figure 6.10: Comparison of waiting time objective using deterministic and stochastic approach. Box plots indicate, from left to right, SPEA2 deterministic, SPEA2 without hedging, SPEA2 with hedging, NSGA-II deterministic, NSGA-II without hedging and NSGA-II with hedging.

6.6 Discussion

The presented analysis provides some interesting insights that we discuss further here. In the preliminary analysis (section 6.3) we argue, by showing examples of schedules produced by the deterministic singleobjective hybrid GA, that multiobjectivity and uncertainty incorporation is necessary when optimizing elective patient surgery schedules. This thesis suggests two relatively fast algorithms for achieving both of these goals simultaneously, namely the SPEA2 and NSGA2 using Monte-Carlo simulation for fitness approximation. We see improvement over deterministic approaches, especially regarding overtime objectives. However, a major concern that we do not address properly in this work considers the trade off between the expected performance and performance variability of individuals in the evolving population. Realizing this new dimension of multiobjectivity and how it may affect the evolutionary search process, we question if properly solving multiobjective surgery scheduling problems under uncertainty without any user input is realistic at this point. By ”proper” we refer to the goal of multiobjective optimization of finding all user-preference-independent trade-off solutions, which in this context would refer to finding all optimal trade-offs between expected value and risk for all optimal trade-offs among the five performance measures.

An advantage of the proposed algorithms is their flexibility. By changing only a few lines of code in the fitness function, selection may be based on weighted average or a selection of the presented performance measures. To accommodate different levels of user risk

aversion, the expected value, mean or any percentile of the simulation results may be used as fitness value during evolution. Although we acknowledge that the developed algorithms cannot be put to immediate use by any hospital, it is not hard to imagine actual use after risk adjustments and performance prioritizations have been done in tandem with hospital management at a particular department treating elective patients.

We are now prepared to give concise answers to our research questions from the introductory chapter:

What performance measures are appropriate to include in a realistic surgery scheduling problem?

Overtime objectives and surgeon idle time on one hand and operating room idle time and patient waiting time on the other pulls in different directions when it comes to exclusion and inclusion of surgeries, respectively. Thus, it is appropriate to include at least one driver of including surgeries and one driver for excluding surgeries during optimization. These performance measures must reflect the uncertainty present in realistic surgery scheduling problems. We have assumed that it is appropriate to optimize based on the expected value of the performance measures, but there is little doubt that if ever applied to a real life hospital, the risk aversion of practitioners must be reflected in the optimization process.

Are MOEAs appropriate for solving multiobjective surgery scheduling problems?

The novel genetic mechanisms developed here exemplifies how MOEAs may be applied to a variation of the surgery scheduling problem. Scheduling problems are among the hardest combinatorial optimization problems, and the problem instances investigated here are both large in terms of the number of jobs, machines and operations, and complex in terms of multiple resources, open sequence among jobs, partial machine allocation flexibility and uncertain processing times. Our assessment is that the presented results serve as an argument for continued development and experimentation with MOEAs on surgery scheduling problems, as we find several promising solutions within 20 minutes of computation time for relatively comprehensive problem instances. The most important advantage with the chosen representation and associated genetic mechanisms is that only feasible solutions are considered. While the shortest path from an initial solution to the optimal solution may often traject through infeasible solutions, allowing infeasability in our problem would enlarge the decision space substantially. Furthermore, another feature that decreases the decision space is that which surgeries to included is not explicitly stated in the chromosome. We therefore conclude that MOEAs are appropriate for solving surgery scheduling problems. However, we call out for benchmark problems with known Pareto optimal fronts that can assist accurate convergence assessment and comparison with alternative algorithms.

How should sources of uncertainty be modelled and accounted for in surgery scheduling algorithms?

Our results indicate that using explicit averaging with Monte-Carlo simulation is a predictably fast way of estimating fitness during evolutionary optimization. Based on a very simple computational analysis, we do not recommend implicit averaging for this problem. It is important to note, however, that this conclusion cannot be generalized to problems where decisions are made after uncertainty resolution. Our approach does not accurately asses the dynamics in real hospitals where decisions, such as rescheduling activities and cancellations, actually take place as uncertainty resolution is observed throughout the day. In such situations, the simulation cost is probably much higher, and implicit averaging might be a better option then explicit averaging. We have also realized through reflection and discussion, that uncertainty incorporation comes with a new set of multiobjective considerations regarding expected value and variability of performance. It is appropriate to involve hospital management in such considerations as the task of locating the true Pareto front for all possible trade-offs between expected performance and performance variability seems unrealistic.

Our final conclusion is that there is great potential when it comes to modeling and solving realistic surgery scheduling problems. In the next section, we give our thoughts on how to proceed in order to see actual use of surgery scheduling algorithms.

6.7 Further Work

In this section we give our recommendations for future research on the considered subject, summarized here:

Gain confidence about convergence. *We welcome benchmark problems with a known Pareto-front that can enable accurate assessment of convergence during development and comparison of multiobjective heuristic applied to the surgery admission planning problem.*

Increase the scope of analysis and testing. *Further testing should be conducted in order to compare the SPEA2 and NSGA-II with statistical significance.*

Towards computational tractability. *The presented MOEAs can be configured in various ways, and other population based methods can be applied to the presented problem.*

Dynamic surgery scheduling. *The surgery scheduling problem is, in reality, a dynamic problem, and MOEAs for dynamic environments can assist practitioners before and during schedule execution.*

Involve hospital management. *As the motivation for this work is to see actual use of surgery scheduling algorithms, we suggest that future operational research work closer with hospital management, particularly when it comes to deciding appropriate performance measures and risk preferences.*

The motivation of the presented work is to narrow the gap between theoretical surgery analysis and actual use of surgery scheduling algorithms. The three major challenges we address are computational tractability, multiobjectivety and uncertainty. We suggest several extensions on how to direct future research at these challenges in order to increase surgery scheduling algorithm quality and accommodate the requirements of real life hospitals.

As a means to overcome the computational intractability of the modeled surgery admission planning problem, we apply evolutionary algorithms. The presented SPEA2 and NSGA-II are implemented with only a sample configuration of the underlying genetic mechanisms. Other configurations may certainly be superior, and we welcome studies suggesting e.g. other representations and decoding procedures. Also, the presented hyperparameters are based on a rather adhoc trial and error procedure, and a systematic adjustment of important parameters such as population size and mutation rate could lead to better computational performance of the two MOEAs. Furthermore, we are not able to confidently measure the convergence of our MOEAs, as convergence metrics such as error ratio, generational distance, inverted generational distance and maximum Pareto front error all require the true Pareto front which is not known for the investigated problem instances. We therefore call for benchmark instance generation of realistic multiobjective surgery scheduling problems that can assist precise quality assessment of multiobjective surgery scheduling algorithms. This would also facilitate a more comprehensive and accurate comparison between SPEA 2 and NSGA-II for the presented problem. Other population based algorithms, such as particle swarm optimization, ant colony optimization and bees algorithm, could also be successfully applied to the presented problem.

We model a two-stage stochastic program in which no action is allowed after realization of the uncertain lognormally distributed surgery durations. In reality, surgery schedulers often skillfully adapt the schedule as deviations are observed throughout the day. A more accurate model considers the surgery scheduling problem as a dynamic problem, and MOEAs created for dynamic environments, such as DB-DEMOA and DNSGA-II, would be more appropriate than the developed SPEA2 and NSGA-II that use simple Monte-Carlo simulation for robust optimization. Such methods could also consider situations where the same resources are allocated to both elective and non-elective patients. In order approach actual use of surgery scheduling algorithms, hospital managers should cooperate with developers. This could be mutually beneficial as user preference can simplify the algorithmic search *and* ensure that produced solutions are relevant to the user.

Chapter 7

Conclusion

Surgery scheduling problems are spacial instances of large, multiobjective machine scheduling problems with uncertain processing times and multiple resource constraints. These problems are continuously encountered in hospitals delivering surgical treatment to patients, and they must be solved with great effort and care in order to utilize scarce resources, balance conflicting interests and hedge for uncertainty. The motivation of this thesis is to narrow the gap between evolutionary approaches to machine scheduling problems from literature and actual use of surgery scheduling algorithms in hospitals. We formulate a new variation of the surgery admission planning problem and develop genetic mechanisms to solve it with state of the art multiobjective evolutionary algorithms. By using Monte-Carlo simulation to estimate fitness of individuals during optimization, we attempt to find solutions that are robust against variations in surgery durations.

Our most important contribution is uncertainty incorporation in multiobjective search for robust surgery schedules. The proposed way of estimating fitness with Monte-Carlo simulation (explicit averaging) performed *after* decoding shows significant improvements on the overtime performance measures without incurring too much computational cost. However, this approach assumes a simplified problem structure where no decisions are made after the resolution of uncertainty. Future research should consider dynamic surgery scheduling where rescheduling activities are allowed as realized durations are observed throughout the day. Implicit averaging should be considered as an option if simulation becomes too time consuming in this endeavour.

An underlying assumption of this work is that scheduling algorithms are not widely used in hospitals partly because algorithms to date cannot accurately handle the dynamics present in surgery scheduling environments. There are certainly other reasons, such as lack of knowledge and resources needed to implement successful algorithms in hospital information systems. Our assessment is that evolutionary algorithms have great potential when it comes to solving realistic surgery scheduling problems. In order to see actual use of surgery scheduling algorithms, we encourage the involvement of hospital managers. It is

particularly important to align user preference when it comes to performance trade-offs and risk with algorithmic search for diversity and robustness.

Bibliography

Abedinnia, H., Glock, C. H., Schneider, M. D., 2017. Machine scheduling in production: a content analysis. Applied Mathematical Modelling 50, 279–299.

Addis, B., Carello, G., Tànfani, E., 2014. A robust optimization approach for the advanced scheduling problem with uncertain surgery duration in operating room planning-an extended analysis.

Aringhieri, R., Landa, P., Soriano, P., Tànfani, E., Testi, A., 2015. A two level metaheuristic for the operating room scheduling and assignment problem. Computers & Operations Research 54, 21–34.

Batun, S., Denton, B. T., Huschka, T. R., Schaefer, A. J., 2011. Operating room pooling and parallel surgery processing under uncertainty. INFORMS journal on Computing 23 (2), 220–237.

Bruni, M., Beraldi, P., Conforti, D., 2015. A stochastic programming approach for operating theatre scheduling under uncertainty. IMA Journal of Management Mathematics 26 (1), 99–119.

Cardoen, B., Demeulemeester, E., Beliën, J., 2009. Optimizing a multiple objective surgical case sequencing problem. International Journal of Production Economics 119 (2), 354–366.

Cardoen, B., Demeulemeester, E., Beliën, J., 2010. Operating room planning and scheduling: A literature review. European journal of operational research 201 (3), 921–932.

Chen, C.-M., Chen, Y.-p., Zhang, Q., 2009. Enhancing moea/d with guided mutation and priority update for multi-objective optimization. In: Evolutionary Computation, 2009. CEC'09. IEEE Congress on. IEEE, pp. 209–216.

Cheng, R., Gen, M., Tsujimura, Y., 1996. A tutorial survey of job-shop scheduling problems using genetic algorithmsi. representation. Computers & industrial engineering 30 (4), 983–997.

Choi, S., Wilhelm, W. E., 2012. An analysis of sequencing surgeries with durations that

follow the lognormal, gamma, or normal distribution. IIE Transactions on Healthcare Systems Engineering 2 (2), 156–171.

Company, A. C., 1964. Critical Values for the Wilcoxon Signed-Rank Test. https://math.ucalgary.ca/files/math/wilcoxon_signed_rank_table.pdf, [Online; accessed 04-June-2018].

Corne, D. W., Knowles, J. D., Oates, M. J., 2000. The pareto envelope-based selection algorithm for multiobjective optimization. In: International Conference on Parallel Problem Solving from Nature. Springer, pp. 839–848.

Deb, K., 2014. Multi-objective optimization. In: Search methodologies. Springer, pp. 403–449.

Deb, K., Mohan, M., Mishra, S., 2005. Evaluating the ϵ-domination based multi-objective evolutionary algorithm for a quick computation of pareto-optimal solutions. Evolutionary computation 13 (4), 501–525.

Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., 2002. A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE transactions on evolutionary computation 6 (2), 182–197.

Dellaert, N., Jeunet, J., 2017. A variable neighborhood search algorithm for the surgery tactical planning problem. Computers & Operations Research 84, 216–225.

Denton, B., Gupta, D., 2003. A sequential bounding approach for optimal appointment scheduling. IIE transactions 35 (11), 1003–1016.

Denton, B., Viapiano, J., Vogl, A., 2007. Optimization of surgery sequencing and scheduling decisions under uncertainty. Health care management science 10 (1), 13–24.

Derrac, J., García, S., Molina, D., Herrera, F., 2011. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm and Evolutionary Computation 1 (1), 3–18.

Dittami, S., 2009. Shapiro-Wilk Normality Test. http://sdittami.altervista.org/shapirotest/ShapiroTest.html, [Online; accessed 04-June-2018].

Etiler, O., Toklu, B., Atak, M., Wilson, J., 2004. A genetic algorithm for flow shop scheduling problems. Journal of the Operational Research Society 55 (8), 830–835.

Fei, H., Meskens, N., Chu, C., 2010. A planning and scheduling problem for an operating theatre using an open scheduling strategy. Computers & Industrial Engineering 58 (2), 221–230.

Gao, J., Gen, M., Sun, L., Zhao, X., 2007. A hybrid of genetic algorithm and bottleneck shifting for multiobjective flexible job shop scheduling problems. Computers & Industrial Engineering 53 (1), 149–162.

Gen, M., Tsujimura, Y., Kubota, E., 1994. Solving job-shop scheduling problems by genetic algorithm. In: Systems, Man, and Cybernetics, 1994. Humans, Information and Technology., 1994 IEEE International Conference on. Vol. 2. IEEE, pp. 1577–1582.

Gillespie, B. M., Chaboyer, W., Fairweather, N., 2011. Factors that influence the expected length of operation: results of a prospective study. BMJ Qual Saf, qhc–2011.

Goh, C.-K., Tan, K. C., 2009. Evolutionary multi-objective optimization in uncertain environments. Studies in Computational Intelligence 186.

Gonzalez, T., Sahni, S., 1976. Open shop scheduling to minimize finish time. Journal of the ACM (JACM) 23 (4), 665–679.

Gosavi, A., 2003. Simulation-based optimization. parametric optimization techniques and reinforcement learning.

Guido, R., Conforti, D., 2017. A hybrid genetic approach for solving an integrated multi-objective operating room planning and scheduling problem. Computers & Operations Research 87, 270–282.

Gul, S., Denton, B. T., Fowler, J. W., Huschka, T., 2011. Bi-criteria scheduling of surgical services for an outpatient procedure center. Production and Operations management 20 (3), 406–417.

Guo, M., Wu, S., Li, B., Song, J., Rong, Y., 2016. Integrated scheduling of elective surgeries and surgical nurses for operating room suites. Flexible Services and Manufacturing Journal 28 (1-2), 166–181.

Hans, E., Wullink, G., Van Houdenhoven, M., Kazemier, G., 2008. Robust surgery loading. European Journal of Operational Research 185 (3), 1038–1050.

Hof, S., Fügener, A., Schoenfelder, J., Brunner, J. O., 2017. Case mix planning in hospitals: a review and future agenda. Health care management science 20 (2), 207–220.

Hulshof, P. J., Kortbeek, N., Boucherie, R. J., Hans, E. W., Bakker, P. J., 2012. Taxonomic classification of planning decisions in health care: a structured review of the state of the art in or/ms. Health systems 1 (2), 129–175.

Iyer, S. K., Saxena, B., 2004. Improved genetic algorithm for the permutation flowshop scheduling problem. Computers & Operations Research 31 (4), 593–606.

Jebali, A., Diabat, A., 2015. A stochastic model for operating room planning under capacity constraints. International Journal of Production Research 53 (24), 7252–7270.

Jin, Y., Branke, J., 2005. Evolutionary optimization in uncertain environments-a survey. IEEE Transactions on evolutionary computation 9 (3), 303–317.

Joustra, P., Meester, R., van Ophem, H., 2013. Can statisticians beat surgeons at the planning of operations? Empirical Economics 44 (3), 1697–1718.

Kayış, E., Khaniyev, T. T., Suermondt, J., Sylvester, K., 2015. A robust estimation model for surgery durations with temporal, operational, and surgery team effects. Health care management science 18 (3), 222–233.

Laporte, G., Louveaux, F. V., 1993. The integer l-shaped method for stochastic integer programs with complete recourse. Operations research letters 13 (3), 133–142.

Lenstra, J. K., Kan, A. R., Brucker, P., 1977. Complexity of machine scheduling problems. In: Annals of discrete mathematics. Vol. 1. Elsevier, pp. 343–362.

Louis, S. J., Xu, Z., 1996. Genetic algorithms for open shop scheduling and re-scheduling. In: Proc. of the 11th ISCA Int. Conf. on Computers and their Applications. Vol. 28. pp. 99–102.

Magerlein, J. M., Martin, J. B., 1978. Surgical demand scheduling: a review. Health services research 13 (4), 418.

Mancilla, C., Storer, R., 2012. A sample average approximation approach to stochastic appointment sequencing and scheduling. IIE Transactions 44 (8), 655–670.

Mancilla, C., Storer, R. H., 2013. Stochastic sequencing of surgeries for a single surgeon operating in parallel operating rooms. IIE Transactions on Healthcare Systems Engineering 3 (2), 127–138.

Marques, I., Captivo, M. E., Pato, M. V., 2015. A bicriteria heuristic for an elective surgery scheduling problem. Health care management science 18 (3), 251–266.

Mateus, C., Marques, I., Captivo, M. E., 2017. Local search heuristics for a surgical case assignment problem. Operations Research for Health Care.

Min, D., Yih, Y., 2010. Scheduling elective surgery under uncertainty and downstream capacity constraints. European Journal of Operational Research 206 (3), 642–652.

Molina-Pariente, J. M., Fernandez-Viagas, V., Framinan, J. M., 2015a. Integrated operating room planning and scheduling problem with assistant surgeon dependent surgery durations. Computers & Industrial Engineering 82, 8–20.

Molina-Pariente, J. M., Hans, E. W., Framinan, J. M., Gomez-Cia, T., 2015b. New heuristics for planning operating rooms. Computers & Industrial Engineering 90, 429–443.

Nyman, J., 2016. A stochastic optimization approach for scheduling surgeries in parallel operating rooms. Master's thesis, Norwegian University of Science and Technology, Norway.

Nyman, J., 2017. Metaheuristics for the Multiobjective Surgery Admission Planning Problem, unpublished project thesis, Norwegian University of Science and Technology.

Nyman, J., Ripon, K. S. N., 2018. Metaheuristics for the Multiobjective Surgery Admission Planning Problem, unpublished conference paper, IEEE Congress on Evolutionary Computation.

Peng, J., Liu, B., 2004. Parallel machine scheduling models with fuzzy processing times. Information Sciences 166 (1-4), 49–66.

Pezzella, F., Morganti, G., Ciaschetti, G., 2008. A genetic algorithm for the flexible job-shop scheduling problem. Computers & Operations Research 35 (10), 3202–3212.

Pulido, R., Aguirre, A. M., Ibáñez-Herrero, N., Ortega-Mier, M., García-Sánchez, Á., Méndez, C. A., 2014. Optimization methods for the operating room management under

uncertainty: Stochastic programming vs. decomposition approach. J Appl Oper Res 201 (6), 3.

Rath, S., Rajaram, K., Mahajan, A., 2017. Integrated anesthesiologist and room scheduling for surgeries: Methodology and application. Operations Research 65 (6), 1460–1478.

Riise, A., Burke, E. K., 2011. Local search for the surgery admission planning problem. Journal of Heuristics 17 (4), 389–414.

Ripon, K. S. N., Kwong, S., Man, K.-F., 2007. A real-coding jumping gene genetic algorithm (rjgga) for multiobjective optimization. Information Sciences 177 (2), 632–654.

ROC, R., 2006. Determination of sample size in using central limit theorem for weibull distribution. Information and Management 17 (3), 31–46.

Ruiz, R., Maroto, C., 2006. A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. European Journal of Operational Research 169 (3), 781–800.

Shapiro, A., 2008. Stochastic programming approach to optimization under uncertainty. Mathematical Programming 112 (1), 183–220.

Shapiro, S. S., Wilk, M. B., 1965. An analysis of variance test for normality (complete samples). Biometrika 52 (3/4), 591–611.

Shylo, O. V., Prokopyev, O. A., Schaefer, A. J., 2012. Stochastic operating room scheduling for high-volume specialties under block booking. INFORMS Journal on Computing 25 (4), 682–692.

Sprecher, A., Kolisch, R., Drexl, A., 1995. Semi-active, active, and non-delay schedules for the resource-constrained project scheduling problem. European Journal of Operational Research 80 (1), 94–102.

Srinivas, N., Deb, K., 1994. Muiltiobjective optimization using nondominated sorting in genetic algorithms. Evolutionary computation 2 (3), 221–248.

Tan, K. C., Lee, T. H., Khor, E. F., 2002. Evolutionary algorithms for multi-objective optimization: Performance assessments and comparisons. Artificial intelligence review 17 (4), 251–290.

While, L., Hingston, P., Barone, L., Huband, S., 2006. A faster algorithm for calculating hypervolume. IEEE transactions on evolutionary computation 10 (1), 29–38.

Winston, W. L., Venkataramanan, M., Goldberg, J. B., 2003. Introduction to mathematical programming. Vol. 1. Thomson/Brooks/Cole Duxbury; Pacific Grove, CA.

Wu, J., Azarm, S., 2001. Metrics for quality assessment of a multiobjective design optimization solution set. Journal of Mechanical Design 123 (1), 18–25.

Xiang, W., 2017. A multi-objective aco for operating room scheduling optimization. Natural Computing 16 (4), 607–617.

Xiang, W., Yin, J., Lim, G., 2015. An ant colony optimization approach for solving an operating room surgery scheduling problem. Computers & Industrial Engineering 85, 335–345.

Yen, G. G., He, Z., 2014. Performance metric ensemble for multiobjective evolutionary algorithms. IEEE Transactions on Evolutionary Computation 18 (1), 131–144.

Zhang, Z., Li, C., Wang, M., Wu, Q., 2017. A hybrid multi-objective evolutionary algorithm for operating room assignment problem. Journal of Medical Imaging and Health Informatics 7 (1), 47–54.

Zhou, B.-h., Yin, M., Lu, Z.-q., 2016. An improved lagrangian relaxation heuristic for the scheduling problem of operating theatres. Computers & Industrial Engineering 101, 490–503.

Zitzler, E., Laumanns, M., Thiele, L., 2001. Spea2: Improving the strength pareto evolutionary algorithm. TIK-report 103.

Zitzler, E., Thiele, L., 1999. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. IEEE transactions on Evolutionary Computation 3 (4), 257–271.

Zydney, A. L., Aimar, P., Meireles, M., Pimbley, J. M., Belfort, G., 1994. Use of the log-normal probability density function to analyze membrane pore size distributions: functional forms and discrepancies. Journal of Membrane Science 91 (3), 293–298.