Mathematical Formulations for Complex Resource Scheduling Problems

T R Lalita

Indian Statistical Institute

January 18, 2021

Indian Statistical Institute

Doctoral Thesis

Mathematical Formulations for Complex Resource Scheduling Problems

Author:
T R Lalita

Supervisors:
Dr G S R Murthy

A thesis submitted to the Indian Statistical Institute
in partial fulfilment of the requirements for
the degree of
Doctor of Philosophy (in Quality, Reliability and Operations Research)

Statistical Quality Control & Operations Research Unit

Indian Statistical Institute, Hyderabad

January 18, 2021

Dedicated to My Parents and Teachers

Acknowledgements

I would like to express my heartfelt thanks and gratitude to my supervisor, Dr. G S R Murthy for his guidance, patience and stimulating discussions throughout the period of my research and for his ideas and time without which this thesis wouldn't have materialized. I would also like to thank Dr Amitava Bandyopadhyaya, ISI-Kolkata, for bringing to us a unique opportunity for research in scheduling. He played a very important part in the development of this thesis. I am also thankful to Dr D K Manna for his ideas and discussions during the initial work on this thesis and for permitting me to include our joint work in this thesis. I would also like to thank Dr Amit Biswas, ISI-Chennai Center, and Dr G Ravindran, ISI-Chennai Center, for various discussions and ideas.

I am also grateful to all my teachers at masters and graduate level, especially, Late Dr Archana Bansal, for teaching me how to code, Dr Debasree Goswami for the most amazing classes in computer science, and Late Dr Nanda for teaching mathematics creatively.

I am grateful to the staff and teachers at the SQC and OR Unit, ISI- Hyderabad for providing me facilities for research at the institute. I am grateful to Mr K V Ramana, Mr Sirivardhan, Mr O.Shiva Kumar, Mrs Lalita and other staff for all the administrative support.

My family has supported me throughout the duration of my research. Most of all I am grateful to my grandfather Mr T S Ramachandra Rao who has inspired me from childhood to appreciate excellence and to achieve it. I am grateful to my parents, for their wise counsel and unwavering support through the years and my sisters, for all their love. I thank my in-laws for all their support. Lastly, I am grateful to my partner and husband, V V S Ravindra who continues to be my constant source of encouragement.

T R Lalita
29-08-2020

Contents

List of Figures

List of Tables

Chapter 1

Thesis Overview

1.1 Introduction

This thesis deals with development of effective models for large scale real-world resource scheduling problems. Efficient utilization of resources is crucial for any organization or industry as resources are often scarce. Scheduling them in an optimal way can not only take care of the scarcity but has potential economic benefits. Optimal utilization of resources reduces costs and thereby provides a competitive edge in the business world. Resources can be of different types such as human (personnel-skilled and unskilled), financial(budgets), materials, infrastructures(airports and seaports with designed facilities, windmills, warehouses' area, hotel rooms etc) and equipment (microprocessors, cranes, machinery, aircraft simulators for training), etc. Typically, scheduling of resources is done over a period of time (planning horizon), but in many cases there are other dimensions to the problem induced by limited resources.

In one class of resource scheduling where resources are facilities, there is a special requirement that the limited resources are assigned to tasks in an adjacent manner. This class is known as adjacent resource scheduling (ARS) (see Duin and Sluis (2006)). There are numerous applications to this model but in this thesis we are concerned with two important applications of this class of problems. The first one is assigning check-in counters to flight departures at airports. Here the counters assigned to each departure must be adjacent (physically). Because the number of counters is limited in any given

airport, counter number (or its id) becomes a second dimension in this assignment and scheduling problem. The second application is in the cargo ship management. In cargo ship terminals, ships are moored to the quay for loading and unloading operations. For modelling purposes, the quay is discretised into berth sections of unit length. A ship of length k units requires a space of k contiguous (adjacent) berth sections in the quay. The limited quay space has to be reused to accommodate multiple ships over the planning horizon. Thus, quay space becomes a second dimension in this problem. In addition, the loading and unloading operations are performed by cranes which are fixed on rails. This restricts the free movement of cranes and they cannot bypass their adjacent ones. This adds a third dimension to the berth and crane allocation and scheduling problem. The cranes serving a ship in this set-up, therefore, must be adjacent. Thus, a third dimension to the problem is induced by limited number of movement-restricted cranes. There are a number of other applications of ARS. Some other applications include: (i) private warehouses let out adjacent storage spaces to customers requiring temporary storage spaces, (ii) hotel room reservations for different customers requiring multiple adjacent rooms, (iii) gate allocation to flight arrivals requiring adjacent or nearby gates for connecting flights, etc.

Another class of resource scheduling problems to which this thesis makes an important contribution is the integrated staff and task scheduling problem (ISTSP). In this problem, the resources are staff or workers who work in shifts. Given a set of tasks to be performed over a given period of time, the problem seeks minimum number of workers to complete the tasks. Each task comes with the specification of (i) a time interval during which the task must commence and (ii) duration of the task and the number of workers required for it during each time period. Further, the tasks have precedence relationships like in a project management problem. On the other hand the workers work in shifts according to conditions stipulated and governed by organizational rules and labour laws. This problem is generic to a variety of industries and organizations. The current burning problem of hospital staff management for treating corona patients will be a classic application of ISTSP. In the same context, carrying out lockdown operations round the clock with limited police staff badly requires the application of ISTSP. Call centers, software industries, medical emergency services, airports, etc., are other areas which require the application of ISTSP.

Finally, we look at a scheduling problem that interlinks two scheduling problems of a windmill and a paper mill, both owned by the same organization. The organization trades the green power generated by its windmill with power consumed from the grid at its paper mill. This trading of power at the two mills is governed by a certain scheme offered by the government to promote green power generation. According to this scheme, the organization has to declare, on a daily basis, a power supply schedule at the windmill and draw up a plan (a schedule) to consume power from grid at its paper mill. The two schedules are declared on the planning horizon of 96 time periods of a day, each of 15 minutes duration. The decisions for the schedule at paper mill require dividing the planning horizon into at most four spells, a spell being a group of contiguous time periods of the planning horizon, and determining the amount of power to be drawn from the grid during each spell. This one is a challenging combinatorial optimization problem with a quadratic objective function and intricate relationships among decision variables. It is different from the usual decision making problems that are encountered in the literature. The determination of spells requires dividing the time periods into groups of contiguous time periods. This is a form of adjacency requirement. We have come up with a novel formulation and solution approach for solving this problem. The novelty of our solution approach is in adapting a technique for solving a seemingly unrelated staff scheduling problem ISTSP mentioned above.

This thesis has evolved from our attempt to provide effective solutions to real-life problems from industry. We shall briefly outline the genesis of our contributions in this paragraph. More details are given in the next section. In case of check-in counters problem (Chapter 3), the request for a solution had come from a large international airport which was facing the problem of managing a huge number of flight departures with a limited number of check-in counters. While solving the problem, we found that the solutions offered in the existing literature could not handle the size the of the problem at hand. Further, we could not find a solution in the literature to address a particular aspect of the problem, namely, implementing the first-in-first-out queue discipline in modelling. The next contribution to berth and crane allocation and scheduling problem (Chapter 4) is an off-shoot of extending our ideas used in solving check-in counter problem. In this contribution, we have come up with formulations that are amenable for commercial solvers to find solutions in reasonable times. The main difficulty with the formulations

in the existing literature is that the problem sizes (number of variables and constraints) are too large for the commercial solvers to handle. The origin of our third contribution to ISTSP (Chapter 2) is a request from a software company that needs to prepare weekly schedule for their staff. Starting with this and exploring the literature, we extended the scope to address the more general problem, the ISTSP. The result is that we have come up with a methodology that reduces the solution times drastically compared to the solutions offered in the literature. Our contribution to the windmill problem (Chapter 5) was initiated by a request from a large paper manufacturing company in India.

1.1.1 Types of Resource Scheduling Problems

There are different types of resource scheduling problems. For a detailed discussion of various types of resource scheduling problems see Blazewicz et al. (2013), Pinedo (2002), Brucker and Knust (2000) etc. In this section we will present some important ones and other problems that have close relation with them.

1.1.1.1 Resource Constrained Project Scheduling Problem

In project management problems, various activities of the project are to be scheduled over a planning horizon. The activities require different types of resources. In the resource constrained project scheduling problem (RCPSP), one has to schedule activities with the renewable resources taking into account various restrictions on the activities and the limited resources. Since most real-world problems have limited resources, there is a need for managing them optimally. In the literature, project scheduling problems are known as "time/cost tradeoff problems." Cost reduction often results in increased activity durations and time reduction in increased cost of resources (for a survey on project scheduling problems see Icmeli et al. (1993)). The activities may have precedence relationships and time line specifications. RCPSP is applicable to a wide variety of areas such as construction activities, timetabling, production and manufacturing processes, service industries such as airways, railways, shipping, hospitals, etc (Brucker and Knust (2000)). To address variants of RCPSP, different models have been developed (see the survey by Hartmann and Briskorn (2010)). Thus, RCPSP is the problem of scheduling

activities or jobs over a planning horizon with the available resources and satisfying in-built constraints.

Mathematically, RCPSP may be stated as follows. A project consists of a set $\mathcal{A} = \{0, 1, ..., n\}$ of $n + 1$ activities which are to be processed without interruption, and R_k renewable resources of type k, $k = 1, 2, \ldots, K$. Activity $i \in \mathcal{A}$, $i \neq 0$, n, requires r_{ik} resources of type k and p_i units of processing time. The activities 0 and n are dummy activities denoting the start and end of the project. They require 0 resources and 0 processing time. There is a (strict) partial order on $\mathcal{A}$, i.e., an irreflexive and transitive relation, which represents precedence constraints among the activities. A solution for the RCPSP is a schedule that specifies the start times of the activities and can be represented by a vector $S = (s_0, s_1, \ldots, s_n)$ where s_i is the starting time of activity $i \in \mathcal{A}$. The starting times s_is must satisfy all constraints with respect to both time and resources. Constraints with respect to time are formulated as $s_i + p_i \leq s_j$ for all $i, j \in \mathcal{A}$ with $i \prec j$ (that is, i precedes j). Constraints with respect to resources can be formulated as follows. For each k and each time period t in the planning horizon, $\sum_{i \in A(t)} r_{ik} \leq R_k$, where $A(t)$ is the set of all activities which are under process during time period t. The most common goal of an RCPSP is to minimize the duration of the project. RCPSP is NP-hard in the strong sense (see Blazewicz et al. (1983)). A host of exact and heuristic solution procedures are available for RCPSP (see Vanhoucke et al. (2002), Kolisch and Padman (2001), Neumann et al. (2012)).

1.1.2 Adjacent Resource Scheduling Problem

Adjacent resource scheduling (ARS) problem deals with determination of minimum number of resources required to complete a set of jobs or activities over a planning horizon. The check-in counter problem is a typical application of ARS. For a given set of departures over a day, the question is: What is the minimum number of counters required to serve the departures? The 'adjacency' in this problem refers to and arises from the requirement that the counters assigned to each departure must be adjacent. Another important application of ARS is the determination of the minimum number of quay cranes required to serve cargo ships whose arrival and departure times are fixed over a planning horizon of, say, one week. Two important variants of ARS are: (i) constant

resource allocation (ARS-R) and (ii) variable resource allocation (ARS-V). In ARS-R, each activity requires the same number of resources during its processing time window, whereas in ARS-V the number of resources required varies from one time period to another within the processing time window.

ARS was formally introduced in Duin and Sluis (2006) in the context of check-in counter allocation problem. The initialisms ARS-R and ARS-V were introduced by them.

Mathematically, the problems can be described as follows. Consider a planning horizon of T time periods denoted in the chronological order as $1, 2, \ldots, T$. Let the resources, C in number, be denoted by $1, 2, \ldots, C$. Let $\mathcal{A} = \{1, ..., N\}$ be the set of N activities to be scheduled during the planning horizon. We shall assume that activity $i \in \mathcal{A}$ starts in time period s_i and ends in time period f_i, $1 \leq s_i \leq f_i \leq T$, and requires $r_i(t)$ resources in time period t for $s_i \leq t \leq f_i$. Further, we assume that $s_i,\ f_i$ and $r_i(t)$ are given inputs for each $i \in \mathcal{A}$ and each $1 \leq t \leq T$. With these definitions and inputs, the ARS problem is to determine, for each $i \in \mathcal{A}$, a set of feasible vectors $\mathcal{S}^i = (S^i_{s_i},\ S^i_{s_i+1},\ \ldots,\ S^i_{f_i})$, where each S^i_j is a *contiguous subset* of the resource set $\mathcal{C} = \{1, 2, \ldots, C\}$. Say that B is a contiguous subset of $\mathcal{C}$ if the elements of B are successive numbers when ordered in the ascending order (it is assumed that resources j and j' are adjacent if $|j - j'| = 1$). The vectors $\mathcal{S}^i,\ i \in \mathcal{A}$ are defined to be feasible if, and only if, $S^i_t \cap S^{i'}_t = \emptyset$ for any time period t and any pair of activities $(i,\ i')$, $i \neq i'$.

ARS-R: ARS is said to be ARS-R (R for rectangular) if $r_i(t)$ is invariant of t, that is, $r_i(t) = r_i$ for all $t \in [s_i, f_i]$. This means, the number of resources required for any activity is constant throughout its processing time window.

ARS-V: ARS is said to be ARS-V (V for variable) if $r_i(t)$ varies with t. In other words, we must have an $i \in \mathcal{A}$ and $s_i \leq t \neq t' \leq f_i$ such that $r_i(t) \neq r_i(t')$. This means that there is at least one activity which requires different number of resources during its processing time window.

ARS assignment can be presented in a time-counter space diagram. Figure 1.1 presents examples ARS-R and ARS-V in the context of check-in counter assignment. Duin and Sluis (2006) explore the nature of ARS and observe that they are strongly

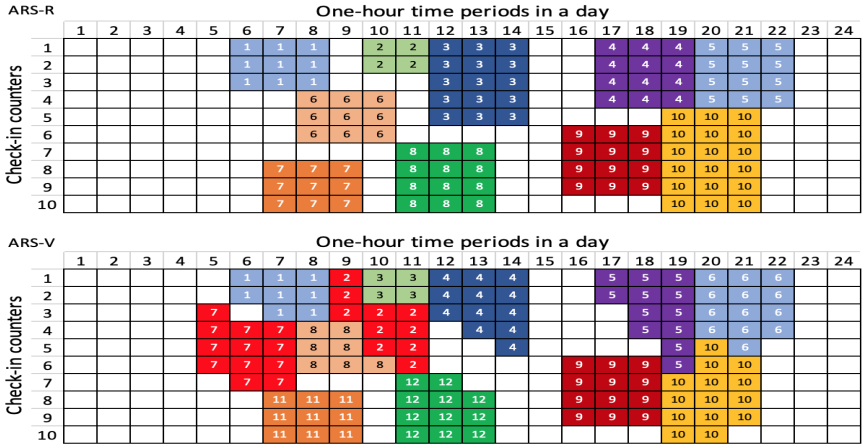


FIGURE 1.1: Assignment of check-in counters under ARS-R and ARS-V. Numbers in the coloured boxes are the departure ids. Number of boxes for each departure in each time period is the number of counters required for that departure in that time period. Number k in cell (i, j) means counter i is assigned to departure k during time period j.

NP-complete. Further, they determine a number of special cases of ARS-R that are solvable in polynomial time. Dijk and Sluis (2006) proposed an integer linear programming formulation for check-in counter problem under ARS-V assumptions.

1.1.3 Strip Packing Problem

The strip packing problem, also known as 2-dimensional cut ting stock problem, is the problem of cutting rectangular pieces of given widths and lengths from a sheet of a given width and infinite length. The sheet of infinite length from which the rectangular pieces are cut is called the master sheet. The objective in this problem is to cut the rectangular pieces using minimum length of the master sheet. This problem is an NP-hard problem. An important application of the strip packing problem occurs in paper industry where paper sheets are cut from a jumbo reel of the paper (Murthy (2016)). Strip packing problem has numerous other applications. In wood or glass industries, rectangular components have to be cut from large sheets of material. In warehousing contexts, goods have to be placed on shelves. In newspapers paging, articles and advertisements have to be arranged in pages. See Lodi et al. (2002) for a survey article on the subject.

1.1.4 Interval Scheduling Problem

The basic interval scheduling problem consists of scheduling a given set of time intervals over a planning horizon. The problem is to determine maximum number of non-overlapping intervals. Typically, each interval stands for an uninterruptible job. This problem has numerous applications. Kolen et al. (2007) provides a brief description of some selected applications in crew scheduling, satellite photography, cottage renting, a channel assignment problem in VLSI-layout, maintenance problem in the aviation industry, a problem in computational biology that determines a maximal subset of a given set of segments of amino-acids that *match* a given sequence of amino-acids, etc.

A mathematical description of the problem is as follows. Given n time intervals $I_j = [s_j,\ f_j),\ j = 1, 2, \ldots, n$, and m machines indexed by $i = 1, 2, \ldots, m$, the basic interval scheduling problem is to find an assignment matrix $X = (x_{ij})$, where x_{ij} is the indicator variable which is 1 if machine i is assigned to interval I_j, so that $\sum_{ij} x_{ij}$ is maximum subject to the constraints: (i) $\sum_i x_{ij} \leq 1$ for all j, and (ii) $I_j \cap I_{j'} = \emptyset$ for all j and j' if $\sum_i x_{ij} = \sum_i x_{ij'}$. Here, each interval stands for a job whose start time is s_j and finish time is f_j. In the discretised version of the problem, the time intervals are replaced by the contiguous subsets of the planning horizon, that is, each interval is of the form $I_j = \{k, k+1, k+2, \ldots, k+p_j\}$ for some integers k and p_j with $k \geq 1,\ p_j \geq 0$ and $k + p_j \leq T$, where T is the number of time periods in the planning horizon. Here, p_j is the processing time of job j.

There are a number of variants of the basic interval scheduling problem. Some are listed below.

- All jobs must be completed (that is, $\sum_i x_{ij} = 1$), job j must start at s_j, and finishing time of job j is $s_j + c_{ij}$ if machine i processes job j, for all $i,\ j$. In this case, one may consider different objective functions. For example, $\sum_{ij} c_{ij} x_{ij}$ stands for the cost of completing all jobs where c_{ij} is the cost of processing job j on machine i. If c_{ij} stands for processing time of job j with machine i, then one might be interested in minimizing $\max\{s_j + c_{ij}x_{ij} :$ `over all` $i, j\}$. This objective function is known as makespan. In the context of berth allocation problems at cargo terminals, this refers to completion time.

- Limited resources: In this case, the number of machines is fixed. Here, the objective is to maximize the number of jobs to be completed. In the check-in counter allocation problem, this means accommodating as many departures as possible with the available number of counters in the airport.
- Flexible start times: In this case, the start times s_js are allowed to belong to an interval. This case arises in the problem of task scheduling.

1.1.5 Multiprocessor Scheduling

In multiprocessor scheduling, the problem is to assign unrelated jobs to multiple processors so that the jobs are processed simultaneously. It is widely applied in parallel computing and manufacturing processes. This is an NP-hard problem and has a number of variants. Two basic versions, $Pm|size_j|C_{\max}$ and $Pm|fix_j|C_{\max}$, are described below. The first parameter Pm states that there are m processors in the system; the second parameter describes the nature of the processors - $size_j$ meaning identical processors and fix_j meaning dedicated processors; and the third parameter $C_{\max}$ stands for the makespan, the completion time of the last job to finish. In $Pm|size_j|C_{\max}$, job j requires $m_j(\leq m)$ processors during its processing time of duration $p_j,\ j = 1, 2, \ldots, n$. In $Pm|fix_j|C_{\max}$, job j requires a specified subset S_j of the m processors during its processing time of duration $p_j,\ j = 1, 2, \ldots, n$. Both versions assume that jobs cannot be preempted. This means no processor can process two jobs if the jobs' processing intervals are overlapping.

The problems described above have relations to the problems considered in this thesis. Clearly the two dimensional strip packing problem (denoted by **2SP**) and the ARS are closely related to the check-in counter and berth and crane allocation problems. Adjacent resource scheduling is a sequencing problem within the class of fixed-interval scheduling problems. Duin and Sluis (2006) observe that ARS-R is a special case of $Pm|size_j|C_{\max}$. Similarly, the RCPSP has close parallels with ISTSP (see Volland et al. (2017b)).

All the problems studied in chapters 2 to 5 are known to be NP-hard problems. Different authors have used a variety of solution approaches to address these problems. These approaches include mathematical programming formulation, branch and bound,

branch and price (column generation), set partitioning, Lagrangian relaxation, simulation, adoptive search methods, heuristics, meta-heuristics, genetic algorithms and so on. Our focus has been on formulation strategies that help obtain practical solutions. In case of check-in counters, the focus was on solving large scale problems; in case of staff scheduling and berth and crane allocation problems, the focus was on reducing problem sizes so that they can be solved using commercial solvers; and in case of windmill problem, the focus is on developing an innovative formulation to convert the nonlinear constraints into linear ones and providing a solution approach.

1.2 Overview of Chapters

This thesis has seven chapters including this introductory chapter and the concluding chapter. This section provides an over view of the next five chapters. Based on the extensive literature study on check-in counters, a survey article is prepared and this is included as chapter A. Chapters 2 to 5 relate to our contributions to the resource scheduling problems. Of these, we consider our solution to ISTSP as the most significant contribution as it demonstrates substantial reduction in solution times compared to the existing solutions from the literature. Therefore, we start the presentation with this as the second chapter. This is followed by the chapter on the check-in counters problem (Chapter 3) which in turn is followed by the chapter on berth and crane allocation and scheduling problem. The windmill problem and its solution are presented in Chapter 5. The thesis is concluded in Chapter 7 with concluding remarks and scope for future research.

1.2.1 Integrated Staff and Task Scheduling Problem

Staff scheduling is a part and parcel of every organization. In many organizations, there is a requirement of staff scheduling on a periodic basis. This is due to various factors such as varying work loads, availability of workers, nature of workers (regular and temporary), labour laws, payment structures and so on. In organizations which work round the clock (such as hospitals, police, medical emergency services, paper industry, call centers, cab services in a city, etc.), workers may work according to flexible shifts

which vary in duration as well as work timings within a day. Scheduling staff in such cases is a complex problem as there are many constraints imposed by various factors. Staff scheduling in which the planning horizon extends beyond a day with days-off constraints is known as tour scheduling problem. Scheduling problems in which shifts do not extend across two consecutive days are referred to as discontinuous staff scheduling problems (see Stolletz (2010), Brunner and Stolletz (2014)). Staff scheduling is often driven by tasks with time varying workforce requirements. Tasks usually come with time lines and precedence relationships. Once tasks are scheduled, they fix the workforce requirements over different time periods of the planning horizon, and one has to choose a suitable staff schedule that is optimal in some way. However, it is possible to derive better solutions by making a combined decision of scheduling tasks and staff together. This problem is referred to as the integrated staff and task scheduling problem, the ISTSP. In Chapter 2, we deal with ISTSP and a special case of it, namely, the staff scheduling problem. In staff scheduling problem, the task schedule is fixed and its staff requirements are taken as given inputs.

The work of Chapter 2 started with a problem addressed to us by a software company. A division of the company works for a client. The company receives workforce plan from the client every week. The plan specifies the number of staff required in each of the 336 thirty-minute periods of the week. The company's problem is to determine the staff schedule to meet the specified plan. According to the company's rules, workers can be assigned work in shifts subject to the following conditions: (i) shift has two tea breaks each of 15 minutes duration and one lunch break of 60 minutes, (ii) no break in the first 90 minutes, (iii) at least 90 minutes gap between any two successive breaks, and (iv) the duration of the shift including breaks is 9 hours. Further, the work demand exists only during 8 am to 10 pm. The problem is to determine a feasible staff schedule for a given plan with minimum number of workers. This problem is similar to the one addressed by Stolletz (2010) and Brunner and Stolletz (2014) using column generation approaches for workforce planning at check-in systems at airports. In 2017, observing that there is limited literature on ISTSP, Volland et al. (2017b) propose a solution approach using an iterative solution procedure using column generation approach. Following this study, we extended our solution approach for the software company's problem to the ISTSP. This has lead to interesting contributions of the chapter.

Providing relief breaks in shifts is an important aspect of scheduling (see Jacobs and Bechtold (1993), Aykin (1996), Su et al. (2014)). The size of staff scheduling problem increases dramatically with the introduction of breaks in the shifts (see Stolletz (2010) and Brunner and Stolletz (2014)). Combining this with flexibility in task scheduling, the size of ISTSP will blow up exponentially. Furthermore, with respect to problem size there is a huge difference between the problems with continuous work demand case and the discontinuous one. Among the three papers cited in the previous paragraph, only Brunner and Stolletz (2014) allows breaks, that too just one break. In contrast, our models consider both continuous and discontinuous staff scheduling problems with multiple flexible lunch and rest breaks. Comprehensively, the following features are incorporated in our models.

- With regard to staff scheduling restrictions, allow: flexible shift lengths, flexible start times, maintaining the time gap between consecutive shifts, limiting the number of shifts per worker in the planning horizon and flexible days-off.
- With regard to task scheduling restrictions, allow: flexible start times within specified time windows, flexible task durations, flexible manpower requirements and precedence relationships.

The objective function is either cost of workers or the number of workers. We use a two-stage approach to solve the problems. Using integer linear programming formulations, we obtain near optimal solutions. When the cost depends only on the shift characteristics (duration and the timings), our method yields optimal solutions. We test the efficacy of the solutions of method with a large number of problem instances and compare them with relevant results from the literature. To handle large scale problems, we introduced *the split technique*. This has resulted in heavy reductions in processing times.

1.2.2 The Check-in Counter Allocation Problem

Check-in counters are a critical resource for airport operators. Passengers of each flight first go to the check-in counters assigned for their flight to drop their luggage and obtain boarding passes. Check-in counters are intertwined with an automated system that takes

care of printing and issuing boarding passes and luggage handling (printing luggage tags, ensuring that the luggage goes to the correct flight, etc). Check-in counters are built according to a well planned layout (depending on design and size of the airport) and hence are physically static resources. In large airports, they are typically organized in rows within *islands* - each island containing two rows of contiguous counters in opposing directions (see Figure 1.2).

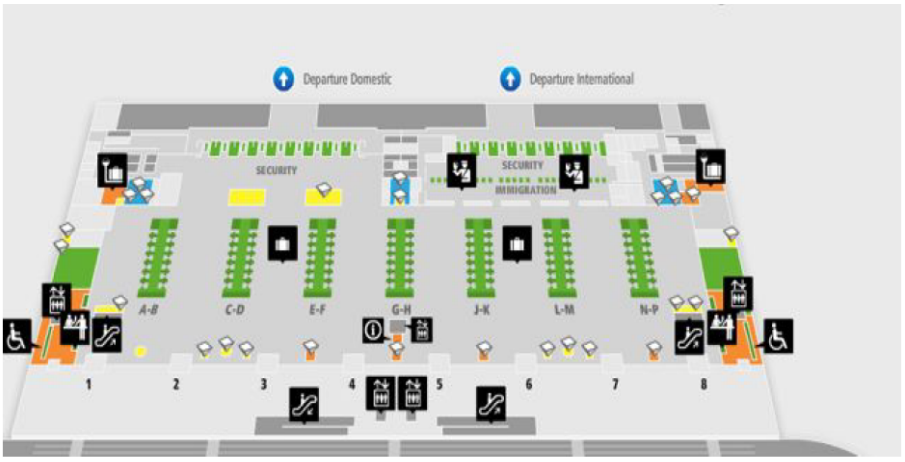


FIGURE 1.2: Check-in counters

Check-in counters are the property of the airport operator and it is the job of the operator to assign counters to different airlines for the departing flights of the airlines. The number of counters assigned to a departure (the departing flight) depends upon factors such as the aircraft features (number and classes of seats) and category (domestic or international flight), airline category (national or private carrier), etc. Counters assigned to a particular departure typically operate for the departure in a time window of 3 to 4 hours prior to the flight's departure time excluding the last 45 minutes to one hour. If the number of counters is constant throughout this time window, then it is known as fixed counter allocation. Otherwise it is known as variable counter allocation. As the number of passengers arriving for check-in during the time window varies, variable counter allocation leads to better utilization of the counters. While the fixed counter allocation is convenient from passenger perspective, the variable counter allocation may become inevitable if the airport has to accommodate a large number of departures. In the check-in counter allocation problem, we are concerned with the variable counter allocation. A large international airport in India approached us for a solution to this problem. The airport inaugurated in 2010 and built for handling 34 million passengers

per annum had to handle 57.7 million passengers in the financial year 2016-17. The airport was handling about 2500 departures per week when they approached us for a solution. The solution sought is meant for tactical planning. The airport's requirement is to prepare seasonal plans for two seasons in a year. The seasonal plans are based on weekly roster. That is, the same departures of a week are repeated every week during the season. Taking requests for departures from different airlines, the airport operator has to decide which departures have to be accepted. While making this decision, the operator must ensure that it will be feasible to assign the limited number of check-in counters available in the airport to the accepted departures.

There are two concerns in this problem: (i) to determine the assignment of counters to the requested departures and (ii) the solution must meet the standard norms which mainly refer to passenger waiting times assessed under rational assumptions. A number of authors have addressed (i) in the literature. However, the solutions offered by them are not suitable for large scale problems (details are presented in the literature section of Chapter 3). Regarding (ii), one of the rational assumptions is serving the passengers according to first-in-first-out queue discipline. This aspect has not been addressed in the literature and we are the first to include this constraint in modelling the problem. We addressed the two concerns using a two-stage optimization approach. In the first stage, we determine the variable number of counters required for each departure through an integer linear programming formulation that ensures first-in-first-out principle. The problem of the second stage requires assignment of adjacent counters for departures and thus comes under ARS. Using a special strategy to handle large scale problems, we propose an integer linear programming formulation. The passenger waiting time aspects are related to the solution of the first stage problem. Our results are compared with the corresponding solutions from the literature and the benefits are discussed. Through the application of our formulation for the second stage problem to large scale problem instances, we demonstrate the effectiveness of our solution.

Our solution to check-in counter allocation problem is applicable to all airports and will be essential for airports with large number of departures. World air traffic is experiencing phenomenal growth. Several countries are increasing number of airports extending the air travel even to small cities and towns. In 2017, India topped the air traffic growth chart with a 20.3% rise in passenger traffic in one year and announced the

UDAN (Ude Desh Ka Aam Nagrik) scheme to double the number of airports. China has announced to double its number of airports by 2037. Under this scenario, solution methodologies like ours will be very essential for planning infrastructural development as well as operational requirements.

1.2.3 Berth Allocation and Crane Assignment and Scheduling Problem

Container shipping is a major mode of transportation of goods and commodities across the world. Around 80 per cent of the volume of international trade in goods is carried by sea, and the percentage is even higher for most developing countries (Review of Maritime Transport, UNCTAD, 2019). Ships are technically sophisticated, high value assets (larger hi-tech vessels can cost over US $200 million to build), and the operation of merchant ships generates an estimated annual income of over half a trillion US Dollars in freight rates (International Chamber of Shipping (2020)). In terms of value, global seaborne container trade is believed to account for approximately 60 percent of all world seaborne trade, which was valued at around 12 trillion U.S. dollars in 2017. While the quantity of goods carried by containers has risen from around 102 million metric tons in 1980 to about 1.83 billion metric tons in 2017, vessels have likewise increased their capacity. Between 1980 and 2019, the deadweight tonnage of container ships has grown from about 11 million metric tons to around 266 million metric tons (see Statista (2020)). This huge amount of container/ quantity of goods carried by ships around the world has necessitated operational efficiency at ports.

In Chapter 4, we are concerned with an important operational problem at cargo terminals of seaports, namely, the berth and crane allocation and scheduling problem. There are different types of cargo terminals at seaports, and we are concerned with seaports with quay equipped with Gantry cranes fixed on rails along the quay (see Figure 1.3).

Ships are moored along the quay for loading and unloading operations. The cranes on the rails, also known as quay cranes, perform the unloading and loading operations from the ships by transferring the cargo from ships (sea side) to trucks (land side) and

FIGURE 1.3: Gantry Cranes at a seaport

vice versa. Depending on the length of the quay, multiple ships are moored to the quay and are served parallelly. Due to limited quay length and limited number of cranes, ships are served in a particular sequence. Ships arrive at different times with different loads. The port operator is responsible for serving the ships (performing the loading and unloading operations) under various contractual agreements, penalty structures for delays, and so on. The decision making problem we are concerned with is scheduling the service times of the ships, their positions on the quay and the cranes that serve them. Depending upon the modelling assumptions, this problem has several variants.

We shall present the variant that is relevant to our study. If there are no restrictions on positioning the ships along the quay, the problem comes under continuous berth allocation. For the purpose of modelling, the quay is discretised into berth sections of unit lengths so that ships are moored at contiguous berth sections adequate in number to accommodate the lengths of the ships. Thus, in the scheduling problem, ships are assigned contiguous berth sections for a specified period of time during the planning horizon. This problem is known as the berth allocation problem (BAP). On the other hand the available cranes are distributed to ships moored at the quay and are scheduled to serve the ships. The specification of which crane(s) will serve which ships and during what periods is known as the quay crane assignment problem (QCAP) (see Bierwirth and Meisel (2010) for details). When the scheduling is planned taking into account all possible options of the two problems, BAP and QCAP, the problem is known as berth and crane assignment (specific) problem (BACASP) (Türkoğulları et al. (2014) and Agra and Oliveira (2018)).

BAP, QCAP and BACASP have been studied by several authors (see the survey articles Bierwirth and Meisel (2010) and Bierwirth and Meisel (2015)). Most of the solution methods using integer linear programming formulations have a very large number of variables and constraints. A number of nonlinear constraints arise in the process of ensuring assignment of adjacent berth sections to ships and non-crossing constraints of cranes. To convert the nonlinear constraints to linear constraints, big-M restrictions are introduced in the formulations. The huge spike in the size of the problem is triggered by the requirement of adjacency constraints. As a result of this, the problems using these formulations cannot be solved directly using commercial solvers. One way to handle the size of the problem is to use column generation approach (see Wang et al. (2018)). Presenting a mathematical model based on the relative position formulation for the problem, Agra and Oliveira (2018) propose enhancements to the formulation and bounds on the objective function derived from a rolling horizon heuristic.

The formulations and solution approaches discussed in the above paragraph are aimed at finding global optimal solutions. Further, the solutions obtained in this fashion may be difficult to implement. For example, if the cranes are to be shifted from one ship to another back and forth, it will not only result in operational inconvenience but also in time losses due to frequent changes - an aspect that is usually not considered in the formulations. To avoid such undesirable solutions, one would often compromise on the optimality by confining to a class of near optimal solutions that are operationally convenient and easy to find. For instance some authors have considered time-invariant crane assignment models in which cranes assigned to a ship cannot serve other ships until the service of the ship is completed (see Park and Ahn (2003), Bierwirth and Meisel (2010), Türkoğulları et al. (2014)). In Chapter 4, we introduce a new class of solutions in which both berths and cranes remain invariant during the entire planning horizon. We call these solutions as berth and crane invariant (BCI) solutions. Operationally BCI solutions are very convenient. We present an integer linear programming formulation to find BCI solutions. This formulation has a great advantage over the existing formulations in terms of problem size and its suitability to commercial solvers. To cite an example, for one small instance with only 10 ships and 7 cranes over one week planning horizon (with 168 one-hour time periods), the number of variables, n and constraints, c are as follows: (i) for DRPF formulation $n = 4968$ and $c = 2,37,286$, (ii) for DRPF++ formulation

$n = 7130$ and $c = 3,52,751$, and (iii) for BCI formulation $n = 464$ and $c = 1,350$. The DRPF and DRPF++ are the formulations presented in Agra and Oliveira (2018). We establish the performance of our formulation for BCI solutions through a number of numerical experiments using real-life data. The results are compared with existing solutions with respect to objective values and processing times. One dimension that adds to the complexity of the problem is with respect to crane capacities. If all cranes have the same capacity of loading and unloading speeds, it is referred to as the homogeneous case. We consider both homogeneous and heterogeneous cases as in Agra and Oliveira (2018) and propose two different formulations for the two cases. Finally, we propose an extension to our formulation to expand the class of solutions to reduce the optimality gap.

1.2.4 The Windmill Problem

Globally, the power sector is shifting from the traditional use of fossil fuels to renewable sources of energy. Renewable energy can supply two-thirds of the total global energy demand, and can contribute to the bulk of the greenhouse gas emissions reduction that is required by 2050 (see Gielen et al. (2019)). Wind power is totally pollution free and does not call for maintenance systems such as air or water cooling (https://www.eia.gov/energyexplained/wind/wind-energy- and-the-environment.php).

Most countries in the world are now transitioning to renewable sources of energy. With increasing industrial energy consumption growth in 2010-17 (3.9% annual growth) (International Energy Agency, Paris (2019)), India has also made changes to its Electricity Policy to allow growth of wind power generation by offering private organizations attractive schemes to produce wind power. The government's Inter State Transmission System(ISTS) is one such scheme which allows companies to generate wind power at any place and in lieu of that utilize power drawn from grid (government source) at another place. This exchange happens under stipulated conditions of the ISTS scheme, and is beneficial for the organizations that come forward to generate wind power.

In Chapter 5, the problem deals with scheduling activities of a paper mill that owns a windmill and exchanges power under ISTS scheme. The reason for opting for the scheme

is that the paper mill and the windmill are far apart (situated in two different states and are 800 km apart). The scheduling decisions to be made are as follows. The company has to declare, on a daily basis, a power supply schedule at the windmill and draw up a plan (a schedule) to consume power from grid at its paper mill. The two schedules are made on a planning horizon of 96 time periods of a day, each of 15 minutes duration. The declared power supply schedule at the wind mill is known as the commitment at the windmill. The deviations from this commitment are subject to penalties/rewards according to a pre-defined piece-wise linear cost function. Shortages are penalized more severely than the rewards paid for power supplied in excess of what is committed in each time period of the planning horizon. According to the ISTS scheme, the company can draw power from the grid at the paper mill but the amount of power drawn in any of the 96 time periods of the planning horizon cannot exceed what has been declared at the windmill for that time period.

The paper mill has its own power generating unit, a boiler plant, generating power from the unused wood portion (lignin) as a byproduct of its paper production process. This power is much more expensive than what it gets from the grid in the exchange scheme. The paper mill has a number of power consuming units with known power requirements whose source of power can be switched between the boiler plant power and the grid power during the planning horizon. Due to operational restrictions, the number of switches in a day cannot exceed 3. The company has to decide which units, during what time periods should be connected to the grid. This requires dividing the planning horizon into at most four spells, a spell being a group of contiguous time periods of the planning horizon, and determining the amount of power to be drawn from the grid during each spell. Since the power draw in each time period is limited to what is declared at the windmill, the decisions at the two mills are interlinked. This is a challenging combinatorial optimization problem. The objective function is the overall cost and includes the penalties and rewards at the windmill and the cost of power at the paper mill. It is a quadratic function of the decision variables. This problem is different from the usual decision making problems that are encountered in the literature in the context of windmill power generation problems. The determination of spells requires dividing the time periods into groups of contiguous time periods. This is a form of adjacency requirement. We have come up with a novel formulation and solution

approach for solving this problem. An important part of this solution approach is in adapting an idea from the seemingly unrelated staff scheduling problem, the ISTSP.

The possible power generation at the windmill depends upon the wind speeds which are subject to natural variations. The company has a statistical expert system for forecasting the possible wind power that can be generated during each of the 96 time periods of the planning horizon. Our model takes these forecasts as deterministic inputs in our formulation for computing deviations and penalties/rewards. Due to complexities involved in the model, our model produces ϵ-optimal solutions. That is, given any positive number ϵ, our formulation yields a solution with an optimality gap of at most ϵ. The performance of our solution method is tested with a number of numerical instances and the results are presented.

1.2.5 Contributing Papers

1. *Mathematical formulations for large scale check-in counter allocation problem*, T R Lalita, D K Manna, G S R Murthy, Journal of Air Transport Management, 85 (2020): 101796.

2. *An Efficient Algorithm to the Integrated Shift and Task Scheduling Problem*, T R Lalita, G S R Murthy, Journal of Scheduling (under review)

3. *The Check-in Counter Allocation Problem: A Literature Review*, T R Lalita, G S R Murthy, International Journal of Aviation Management (under review).

4. *The Wind Power Scheduling Problem*, T R Lalita, G S R Murthy, OPSEARCH (accepted for publication).

5. *Compact ILP Formulations For a Class Of Solutions To Berth Allocation and Quay Crane Scheduling Problems*, T R Lalita, G S R Murthy, EJOR (submitted to journal).

For data sets used in different chapters and code associated with this thesis, refer to Appendix B.

Chapter 2

The Integrated Staff And Task Scheduling Problem

2.1 Introduction

Personnel scheduling problems arise in a variety of applications and deal with assignment of shifts to workforce over a planning horizon. A large number of applications involve flexible work schedules. Workforce requirements over the planning horizon are induced by task characteristics such as duration, number of workers required to perform the task, deadlines, etc. Demand, the number of workers required, in each time period of the planning horizon, may be known exactly or assumed according to a predictable pattern, is necessary for the purpose of planning. The flexibility in staff (or work) schedules has two components: (i) type of shift which specifies duration, breaks and their positioning within shift, etc., (ii) time gap between successive shifts, bounds on the number of shifts in the planning horizon, bounds on the total number of worker-hours, days-off, etc. The constraints in the latter component, part of the tour scheduling process, are imposed due to labour laws, company regulations, employees preferences and so on. The complexities and challenges are aggravated by the flexibilities of staff and task schedules. One of the factors that is ignored in much of the existing literature is not including breaks within shifts (see Thompson and Pullman (2007)). Breaks can be included using implicit formulations (see Sungur et al. (2017),Aykin (1996)), but this

would dramatically increase the size of the problem.

In this chapter, we are concerned with two versions of personnel scheduling problem over a discrete planning horizon. In the first version, staff schedules are flexible but the tasks are fixed and the demand of resources (number of personnel required) for each time period of the planning horizon is specified. The second version is an extension of the first and it allows tasks to be scheduled within specified time periods and the tasks may have precedence relationships. The workforce demand is a result of task scheduling. The objective in both versions is to minimize the number of workers or an associated cost. The second version is referred to as *integrated shift and task scheduling problem* (ISTSP). The problem is so complex that it calls for special formulations and methods for solving it. Stolletz (2010) computes the possible tours in a further restricted case of first version of the problem (shifts without breaks, shifts restricted to 4 am to 9 pm) to the tune of 10^{19}. ISTSP is intractable for exact solution approaches. A common mathematical programming approach to solving ISTSP uses set covering formulation or its variants (Dantzig (1954)). Solution approaches presented in Maenhout and Vanhoucke (2016) and Volland et al. (2017b) are some of the recent contributions in this direction.

To the best of our knowledge, the methods for staff scheduling or ISTSP in the existing literature have not considered a wide range of problems. For example, Stolletz (2010) and Brunner and Stolletz (2014) have considered discontinuous tour scheduling problems and not the problems with continuous demand. While the former considers shifts without breaks, the latter considers shifts with only one break. Similarly, Volland et al. (2017b) does not consider breaks within shifts. Moreover, these articles implicitly express that problems with larger demands (by classifying them under small, medium and large) are harder to solve. Against this backdrop, we believe that this chapter makes an important and significant contribution. The main contribution of this chapter is that we provide a new method for ISTSP that can

- reduce solution times drastically,
- solve problems with large demands in approximately the same time taken for problems with small demands, and
- handle wide flexibility in shifts resulting from multiple breaks and days-off.

The organisation of the rest of this chapter is as follows. In the next section, we start with the genesis of this work and present a brief discussion on the extensions of the model assumptions and their consequences. This will be followed by a brief literature review with focus on recent contributions relevant to this paper. In Section 2.3, we present the problem description, our formulations, solution approach and a discussion on their applications. Section 2.4 describes our numerical experiments with data from real-life problems and simulation. The simulation exercises are carefully planned so as to compare our approach with existing methods. Section 2.5 presents the summary of the experimental results. The chapter is concluded in Section 2.6, with a summary and possible scope for future research.

2.2 Motivation and Literature Review

This work is an extension of a problem that we received from a software company. For ease of cross referencing, we shall call this the Software Industry Problem (SIP) in this chapter. The requirement was to develop a method for determining staff schedules with flexible shifts to meet workforce demand specified for every 30-minute time period (TP) over one week planning horizon (336 TPs) with an objective of minimizing the number of workers. Demand for a selected week is shown in Fig. 2.1. The admissible shifts in this problem should satisfy four conditions: (i) shift has two tea breaks each of 15 minutes duration and one lunch break of 60 minutes, (ii) no break in the first 90 minutes, (iii) at least 90 minutes gap between any two successive breaks, and (iv) the duration of the shift including breaks is 9 hours.

This work is the outcome of our effort to solve the SIP in its full flexibility. Encouraged by the results and the nature of our approach, we noticed that it can be extended to ISTSP.

There is a vast literature on personnel scheduling problem. The problem has been classified into different categories depending upon the areas of applications, models and solution approaches. For a detailed review on personnel scheduling problems, see Ernst et al. (2004b) and Van den Bergh et al. (2013), and references therein. Different approaches are pursued for solving staff scheduling problems (see Alfares (2004),

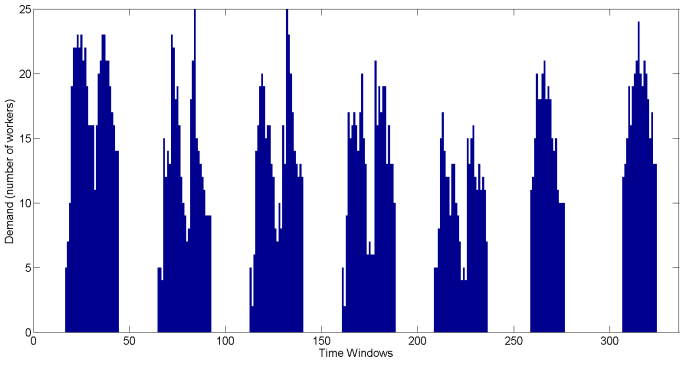


FIGURE 2.1: SIP demand for every 30-minute over one week (336 TPs). No demand during 10 pm to 8 am. Total demand 1258 worker-hours.

Bellenguez-Morineau and Néron (2007) and Brunner et al. (2010)). The main hurdle in solving staff scheduling problems is their size. In SIP, there are 260 different shifts satisfying the stated conditions. Since a shift can commence at the beginning of any of the TPs, there are 12480($= 48 \times 260$) possible shift schedules within a day. High scheduling flexibility results in a huge number of personnel schedules. Mathematical programming formulations for solving the staff scheduling problem are mostly based on the set-covering formulation of Dantzig (1954). As the set covering formulation requires the all personnel schedules, decomposition and column generation techniques through implicit formulations are commonly used to handle the situation. Implicit formulations are developed for several applications (see Thompson and Pullman (2007), Thompson (1995), Jarrah et al. (1994), Jacobs and Brusco (1996),Aykin (1996), Jacobs and Brusco (1996) and Brunner et al. (2009), and Sungur et al. (2017)). Yet, the problem remains complex as the implicit formulations often result in large number of constraints (see Bellenguez-Morineau and Néron (2007) and Brunner et al. (2009)). Decomposition technique is used to breakdown the problem into stages so as to reduce the size of the problem (see Jarrah et al. (1994), Alfares (2004), Stolletz (2010) and Brunner and Stolletz (2014)). Also see Brucker et al. (2011) for a discussion on models and complexities in personnel scheduling problems.

Geibinger et al. (2019) have proposed constraint programming methods for solving a problem similar to RCPSP with heterogeneous resources and restrictions on availability of resources. Another feature is of linked activities, all of which require identical assignment on the same subset of resources. The objective function is to minimize the

total time taken from the start of the first job to the end of the last job. The objective function also differs from the objective of minimizing the number of workers / shifts considered in this article. The authors use constraint programming to solve the problem. In constraint programming, users declaratively state the constraints on the feasible solutions for a set of decision variables. Constraints differ from the common primitives of imperative programming languages in that they do not specify a step or sequence of steps to execute, but rather the properties of a solution to be found. In additions to constraints, users also need to specify a method to solve these constraints. Geibinger et al. (2019) give various strategies for formulating resource constraints, the reduction of search space and search procedures based on heuristics.

The work in this chapter is closely related to three articles: (i) Stolletz (2010), (ii) Brunner and Stolletz (2014) and (iii) Volland et al. (2017b). The first two of these deal with staff scheduling with given resource input, and the the third one deals with ISTSP. First, we shall brief the contributions of these articles and other works related to them.

Stolletz (2010) introduced a *reduced set covering formulation* to solve a personnel touring problem for check-in systems at airports. His model considers a fortnightly planning horizon comprising 30-minute TPs. The staff requirements are needed only in the TPs confined to time between 4 am to 9 pm. The restrictions on the shifts are that they must start and end between 4 am and 9 pm, no breaks are allowed and their durations must be between 6 TPs to 20 TPs (i.e., between 3 hours to 10 hours). With these restrictions, there are 330 staff schedules; and using these, the problem was solved through a binary integer programming formulation. Brunner and Stolletz (2014) expanded the scope of the problem by incorporating one period lunch break in the shifts. They report poor convergence of the column generation subroutine and introduce stabilized column generation procedure.SIP is similar to the one considered by Brunner and Stolletz (2014) but with higher complexity as it involves multiple and more flexible breaks in the shifts (12480 personnel schedules per day). Though SIP is also discontinuous (i.e., workforce is required only between 8 am to 10 pm), we considered the more general problem of continuous case, that is, staff requirements may be there in all TPs. Therefore, our model is more general and more complex, in terms of the size of the problem, compared to that of Brunner and Stolletz (2014). Stolletz and Zamorano

(2014) develop a rolling planning horizon-based heuristic for the tour scheduling problem for agents with multiple skills and flexible contracts in check-in counters at airports.

When supply vector is fixed, ISTSP reduces to the well known resource constrained project scheduling problem (RCPSP) with personnel as resources. See Hartmann and Briskorn (2010) for a survey on RCPSP and its extensions. The published literature on ISTSP is limited. For applications of the problem see Beliën and Demeulemeester (2008), Maenhout and Vanhoucke (2013), Di Martinelly et al. (2014), Kim and Mehrotra (2015), Volland et al. (2017b) in health sector; Beliën et al. (2013) for scheduling problem in an aircraft maintenance company; and Bassett (2000) for a scheduling problem in an agro-based industry. On the solution methods for the problem, see Alfares and Bailey (1997), Bailey et al. (1995), Bassett (2000), Beliën and Demeulemeester (2008) and Beliën et al. (2013) for some early papers on the subject. Maenhout and Vanhoucke (2016) decompose the problem into a master problem and a personnel scheduling subproblem. The personnel schedules used in the restricted master problem are generated iteratively through the personnel scheduling subproblems. Thus, the approach comprises decomposition and column generation techniques. In their model, the TPs are days, and therefore, shifts within days are not considered.

Volland et al. (2017b) propose an ILP formulation (referred to as MIP in their article) for ISTSP with a weekly planning horizon and develop a column generation method to derive a good starting feasible solution with a lower bound for solving the MIP. The method uses implicit formulations for two subproblems - the shift scheduling subproblem (S-SP) and the task scheduling subproblem (T-SP). The two subproblems are linked to a restricted LP relaxation of the MIP to generate personnel and task schedules. The process is continued iteratively by augmenting the restricted master problem with newly generated personnel and task schedules until the optimum objective value of the LP relaxation is attained. Let FLPR stand for the final LP relaxation. After building the (personnel and task schedule) columns of FLPR, they drop a set of task columns (by retaining only a selected set of *high quality* task columns) from FLPR, and add additional personnel schedule columns to it if possible, and solve it as an ILP. Taking the optimal solution of this ILP as a warm start, they solve the MIP. Given below is a review of the existing literature on different variants of the problem.

Staff scheduling is like creating a time table for each employee of a company subject to workplace regulations and labour laws. Considering all the constraints on employee breaks, days-off and maximum allowed man hours in a week with the objective of reducing cost for the organization make this problem complicated. Lau (1996) and Brucker et al. (2011) discusses the complexity of assigning shifts to workers subject to demand and shift change constraints. Brucker et al. (2011) in addition discusses cases of the of the general personnel scheduling problems, both NP-hard and polynomially solvable.

Generally, project scheduling problems involve both staff scheduling and task scheduling. Such problems are split into two subproblems and solved for a cost-effective and labour saving project schedule. Shift scheduling reduces workers' idle time whereas task scheduling optimizes the requirement vector to ensure a minimum gap between worker availability and worker requirement. Shift scheduling for workers in the planning horizon and task scheduling of various activities in a project are integrated to achieve effective solutions. The general staff scheduling problem is addressed in this paper. The methods discussed can be applied for effectively solving problems in staff scheduling such as assigning breaks and days-off to staff.

Many different approaches have been used to solve staff scheduling problems (see Alvarez-Valdes et al. (1999) and Castillo-Salazar et al. (2016)). Alvarez-Valdes et al. (1999) developed an assignment system using tabu search for labour scheduling at airport refuelling installation. Different workers are subject to different conditions on work hours and terms of contract. Tabu search algorithm is used to find the best shifts and then workers are assigned to these shifts. Parisio and Jones (2015) describes a constraint based system using forecasts and stochastic techniques to generate schedules for sales personnel. Soukour et al. (2013) present a staff scheduling problem in airport security service. The problem is solved in three steps, first days off for the staff are scheduled, next, shifts are scheduled and finally a memetic algorithm is used to assign staff to the shifts. Bhulai et al. (2008) present an algorithm for shift scheduling in a call center for multiskilled workers. The objective is to meet the service level constraints with minimal cost. To achieve this, the authors propose a two step approach. First, the minimum staffing level is determined and then, the workers are assigned to different predefined shifts. Each shift is defined by a subset of working hours and a subset of the skillset. A worker with a better skillset is allowed to fulfill a lower skillset requirement. This

results in sub-optimality as the algorithm schedules more shifts than necessary in cases with large number of possible groupings of skills. This results in idle time. Ernst et al. (2004c) review rostering and scheduling problems in different industries.

The basic model for the staff scheduling problem requires a matrix of all possible schedules as input (Brunner and Stolletz (2014)). Including breaks and days-off in the schedules leads to a significantly larger solution space. Modelling and solving such problems as ILPs takes a long time and many cases an optimal solution cannot be attained (for example see the crew scheduling problem in Barnhart et al. (1998) and the physician scheduling problem in Huele (2015)). Decomposition approaches (introduced by Dantzig and Wolfe (1960)) and column generation are popular in finding a solution to large scheduling problems (see Volland et al. (2017a) and Barnhart et al. (1998)).

Most of the models for staff scheduling, therefore, use Dantzig-Wolfe decomposition and column generation methods (see Dantzig and Wolfe (1960), Dantzig and Wolfe (1961)). Column generation is an iterative procedure that considers a subset of feasible columns in each iteration and generates new columns by solving a subproblem of the original problem. Though column generation methods are supposed to help solve ILPs faster than traditional methods, a drawback of these models is that they have an exponential number of decision variables. As it is not practical to enumerate all decision variables, and they easily result in intractable models. There are fundamental difficulties in applying column generation techniques for linear programming in integer programming solution methods (Appelgren (1969), Barnhart et al. (1998)). Column generation methods has been used in scheduling in different industries. For applications of column generation in physician scheduling see Maenhout and Vanhoucke (2013), Beliën and Demeulemeester (2008), in ship scheduling see Appelgren (1969), in network flows see Lübbecke and Desrosiers (2005), vehicle routing see Liberatore et al. (2011), call centers see Bhulai et al. (2008), Avramidis et al. (2010)). After solving an LP relaxation of the original (master) problem, branch and price algorithms along with master problem variable branching lead to the optimal integer solution (see Dantzig and Wolfe (1961), Lübbecke and Desrosiers (2005) and Vanderbeck and Wolsey (1996). Column generation methods combined with branch and price algorithms are popular for solving scheduling problems (see Vanderbeck (2000), Vanderbeck (2011)). Branch and price algorithms consider all possible schedules and evaluate their costs to determine the optimal solution.

Different ways in which column generation can be combined with branch and bound has been discussed by Barnhart et al. (1998), Desrochers and Soumis (1989), Vanderbeck and Wolsey (1996). For a discussion on suitable models for column generation, see Barnhart et al. (1998). For a review of column generation in integer programming (see Wilhelm (2001)).

Branch and price algorithms in addition to column generation methods add further computational complexity. The algorithms and solution methods discussed above take a long time to converge. We aim at developing a simplified algorithm which reduces the computational burden and can be easily used to obtain exact solutions to the integrated staff and task scheduling problems.

Only a few publications exist that integrate shift and task scheduling problems (Volland et al. (2017a)). Maenhout and Vanhoucke (2016) also present an integrated project staffing and task scheduling problem. Maenhout and Vanhoucke (2016) consider only precedence relation among tasks, while Volland et al. (2017a) incorporates time windows. Maenhout and Vanhoucke (2016) include task scheduling in the master problem and have only one subproblem to generate shifts whereas Volland et al. (2017a) have two subproblems to generate shifts and task schedules. Volland et al. (2017a) computes lower bounds for the integrated problem and compares both the models. Scheduling problems in hospitals and healthcare have incorporated integrated approaches to scheduling staff and surgeries. Di Martinelly et al. (2014) integrate nurse scheduling with scheduling surgeries. Beliën and Demeulemeester (2008) present an integrated model for surgery and nurse scheduling. The scheduling problem is decomposed into two subproblems and the shift scheduling problem is solved by using a shortest path formulation. Brunner et al. (2009), Maenhout and Vanhoucke (2013), Huele (2015) also present integrated approaches for physician scheduling, the state-of-art in physician scheduling has been summarized by Erhard and Vanhoucke (2018).

Some related papers are discussed below: Bhulai et al. (2008) present a staffing and shift scheduling problem at a call center for multiskilled workers. For determining the minimum staffing requirement the authors use the model by pot et al and et al. For determining the set of shifts with least cost, the problem is modelled as a linear assignment problem. Their model has rigid tasks (with no scope for change in the task

vector). A major difference in the model proposed and our model is the added complexity in our model due to all possible shift types considered in the shift scheduling.

Zamorano et al. (2018) present a task assignment problem observed at check-in counters at airports. The problem is to assign multiskilled workers to tasks such as check-in, boarding operations etc. The objective is to obtain daily schedules of task assignments for each worker. The authors propose an MIP and a branch and price algorithm to determine task assignment, route assignment and arrival time of workers for each task optimally. The model is decomposed into master problem and pricing subproblems using Dantzig-Wolfe decomposition principle (see Dantzig and Wolfe (1960), Dantzig and Wolfe (1961)) and is solved by generic column generation. The main drawback is that a task may be assigned to a worker more than once. To overcome this drawback, a dynamic programming algorithm is used to model the program as a shortest path problem with resource constraints.

Maenhout and Vanhoucke (2016) present an exact algorithm for project staffing with resource scheduling constraints. Maenhout and Vanhoucke (2016) combine resource scheduling and project scheduling problems together to minimize the overall cost. An exact algorithm is proposed to staffing with resource constraints. The algorithm is based on the decomposition method by Dantzig and Wolfe (1960). Different branching and pruning strategies are explored to compare the proposed procedure with other optimization methods. A branching hierarchy is determined that gives the order of types of variables to be branched. Branching schemes by Barnhart et al. (1998), Raghavan and Stanojević (2011), Vanderbeck and Wolsey (1996), Vanderbeck (2011) and Beliën and Demeulemeester (2008) are used to branch on total number of assigned staff (workload) or subsets thereof. Start time branching strategy for every activity, activity set branching per time period, activity processing branching per time period and precedence relations branching are discussed. Different ways to select the branching variables are also provided. The algorithm proposed has a larger computational time than the algorithm proposed by Volland et al. (2017a).

Volland et al. (2017a) present a column generation approach for integrated shift and task scheduling problem of assistants at hospitals. The objective is to minimize the number of logistics assistants employed by a hospital, given the tasks to be performed and

workforce requirements. A lower bound is used to speed up the solution search procedure for the minimization problem (with an unknown number of workers), the bound is similar to that by Vanderbeck and Wolsey (1996). Alternative problem decompositions with one subproblem (as suggested by Maenhout and Vanhoucke (2016)) and two subproblems are presented and their performance compared. Though the run time of the algorithm is comparatively lower than the algorithm proposed by Maenhout and Vanhoucke (2013), it is still very high and requires solving ILPs iteratively to calculate lower bound for the original MIP. The authors find an initial solution and a lower bound for the original MIP.

Brunner and Stolletz (2014) address the problem of staff scheduling at airport check-in counters with time-varying demand. Their objective is to minimize a cost function based on assigned shifts to a given workforce subject to labour regulations and assignment of lunch breaks. Since the problem could not be solved by standard MIP software, they extend the formulation by Stolletz (2010) using a branch and price algorithm and apply stabilized column generation techniques presented in Du Merle et al. (1999) and Oukil et al. (2007).

Huele (2015) present heuristics to solve the integrated surgery scheduling and physician rostering problem. Operating room scheduling considers physician availability, skills and workload and surgery scheduling depends on the daily bed availability and the length of hospital stay required for the surgery. Priority- rule based scheduling heuristics are used to find feasible solutions. The heuristics have two components, a priority rule and schedule generation scheme. The priority rule signifies an ordering of the activities and schedule generation assigns to a location based on the precedence and other constraints. The drawback of these methods is that only feasible solutions to the problem can be found.

Table 1 gives a brief summary of the papers published on staff scheduling and type of problems addressed.

Additionally, some review papers published on column generation, staff scheduling problems are Van den Bergh et al. (2013), Ernst et al. (2004a), Barnhart et al. (1998), Van den Bergh et al. (2013) discusses different classification methods used in previously

References	Problem Type
Zamorano et al. (2018), Brunner and Stolletz (2014), Stolletz (2010)	Task assignment and staff scheduling at airport check-in counters
Zamorano et al. (2018), Bhulai et al. (2008),	Assigning multiskilled personnel to tasks
Maenhout and Vanhoucke (2016), Volland et al. (2017a), Beliën and Demeulemeester (2008)	Integrated project scheduling and staffing problem.
Brunner and Stolletz (2014),	Flexible assignment of lunch breaks
Van den Bergh et al. (2013),Ernst et al. (2004a),	
Ernst et al. (2004c), Alfares (2004), Castillo-Salazar et al. (2016), Lübbecke and Desrosiers (2005)	Staff Scheduling Problem Review
Maenhout and Vanhoucke (2017)	Non-cyclic scheduling of employees
Ernst et al. (2004a)	To assign specific tasks to staff
Alvarez-Valdes et al. (1999), Lau (1996)	Staffing and optimally rostering employees
Beliën and Demeulemeester (2008), Maenhout and Vanhoucke (2013), Brunner and Edenharter (2011), Brunner et al. (2009) Huele (2015), Erhard and Vanhoucke (2018), Di Martinelly et al. (2014)	Integrated surgery and staff (physician/nurse) scheduling in healthcare sector

published review papers on scheduling and provides classification of papers based on the type of labour contract, type of decisions taken such as assignment of tasks and shift sequence, overlapping of shifts and demand coverage constraints. Ernst et al. (2004a) provide a detailed review of staff scheduling models and rostering methods published prior to 2004.

Our model for ISTSP and the approach to solve it differ from those of Volland et al. (2017b) in three ways: (i) breaks within shifts are more flexible (Volland et al. (2017b) does not incorporate breaks), (ii) we do not use column generation approach, and (iii) we do not solve the MIP which is more complex. To get a solution for ISTSP, we decompose it into two ILP subproblems. Solving the two subproblems produces an optimal solution if the objective function depends only on the *shift patterns* and their positioning, and near optimal solutions if the objective is to minimize the number of workers. The decomposition scheme in our model is based on shift patterns. All shift patterns (allowing full admissible flexibility) can be listed using a simple computer program instead of deriving them through a complex traditional approach of using implicit ILP formulations. Further, we provide a lower bound for the number of workers when there is an upper limit on the number shifts per worker.

2.3 Problem description and Formulation

In this paper, we consider ISTSP over a cyclic planning horizon of one week split into T TPs of equal duration of length ω minutes ($\omega = 15$ or 30 are considered for the instances of this paper). In staff scheduling, a personnel schedule assigns shifts to a worker over the planning horizon fulfilling work schedule restrictions. Each personnel schedule will yield a binary vector in $\mathbb{R}^T$ with 1s representing availability of the worker, who is assigned the schedule, in the respective TPs. Sum of all assigned personnel schedule vectors is a nonnegative integer vector (the supply vector), and its j^{th} coordinate specifies the number of available workers in TP j. On the other hand, task scheduling involves determining start TP of each task satisfying precedence relationships. This will yield a non-negative integer vector in $\mathbb{R}^T$ (the demand vector) specifying the number of workers required in each TP. Under the considered ISTSP, the problem is to determine the personnel schedules (to be assigned to workers) and a task schedule so that the resulting supply vector is greater than or equal to the resulting demand vector. The objective is to minimize the number of assigned personnel schedules or sum of their given associated costs. See Table 2.1 for notation and input parameters.

The planning horizon is $\mathcal{T} = [1, 2, \ldots, T]$. Given K tasks, numbered 1 through K, task k has the following inputs: (i) start window $[l_k,\ u_k]$ in which the task must start, where $l_k,\ u_k \in \mathcal{T}$ with $l_k \leq u_k$, (ii) d_k, duration of the task specified as the number of TPs, and (iii) the resource vector $\mathbf{r}_k = (r_{k1}, r_{k2}, \ldots, r_{kd_k})$, where r_{kj} is the number of workers required in the j^{th} TP of task k, $j = 1, 2, \ldots, d_k$. For the precedence relationships among tasks, the input is a set of task pairs $\mathcal{P}$. If $(k,\ k') \in \mathcal{P}$, it means task k should precede task k'. We use the notation $k \prec k'$ to imply that $(k,\ k') \in \mathcal{P}$.

Table 2.1: Notation

Indices	
k	task number
j	time period (TP) number in the planning horizon
i	shift pattern index, $i = 1, 2, \ldots, q$
v	shift schedule index, $v = 1, 2, \ldots, \tau$
u	worker index, $u = 1, 2, \ldots, w$, where w is maximum number of workers

Parameters	
T	number of time periods in the planning horizon
K	Number of tasks
q	number of shift patterns
l_k	earliest start period of task k
u_k	latest start period of task k
d_k	duration of task k in number of TPs
τ	number of shift schedules from stage 1, $= \sum_{ij} x_{ij}$
$\mathbf{r}_k = (r_{k1}, r_{k2}, \ldots, r_{kd_k})$	demand vector of task k, where r_{kj} is the number of workers required in the j^{th} TP of task k
SL_{min}/SL_{max}	minimum/maximum limits on the length of a shift
SG_{min}	minimum gap (in number of TPs) to be maintained between two successive shifts
$\mathbf{s} = (s_1, s_2, \ldots, s_m)$	shift pattern of length m TPs, $s_1, \ldots, s_m$ are worker availabilities in the appropriate TPs
$(\mathbf{s}^i,\ j)$	shift schedule, shift pattern $\mathbf{s}^i$ starting at TP j
Sets and vectors	
$\mathcal{T} = [1, 2, \ldots, T]$	$\mathcal{T}$ is the planning horizon and T is the number of TPs
$[l_k,\ u_k]$	start time window of task k
$\mathcal{S} = \{\mathbf{s}^1, \ldots, \mathbf{s}^q\}$	set of shift patterns
$\mathcal{P}$	set of task pairs, $(k,\ k') \in \mathcal{P}$ means $k \prec k'$, that is, task k must be completed before starting task k'
$\mathbf{R} = (R_1, R_2, \ldots, R_T)^t$	the demand vector, R_j = the number of workers required in TP j
$\mathbf{S} = (S_1, S_2, \ldots, S_T)^t$	the supply vector, S_j = the number of workers available in TP j
$\mathcal{U}_1,\ \mathcal{U}_2,\ \ldots,\ \mathcal{U}_\tau$	assigned shift schedules from stage 1 arranged in the ascending order of their start TPs
Variables	
y_{kj}	indicator variable which is one if task k is assigned to TP j
x_{ij}	number of shift schedules $(\mathbf{s}^i,\ j)$ (to be assigned to x_{ij} workers in stage 2)
z_{uv}	indicator variable, $= 1$ if $\mathcal{U}_v$ is assigned to worker u

For the staff scheduling, the following inputs/flexibility types are considered: (i) shifts

with or without breaks (as specified) having length between a specified minimum (SL_{min}) and a maximum (SL_{max}), (ii) shift start window is the range of TPs within a day during which a shift can start, (iii) gap between any two successive shifts assigned to a worker in terms of number of TPs must be greater than or equal to a specified lower limit SG_{min}, and (iv) an upper limit either on the number of shifts or total hours assigned to any worker in the week. Note that (ii) above is pertinent to certain specific instances. For example, the 330 shifts referred to in Stolletz (2010) must start between 4 am and 6:30 pm but it depends on the shift length as well; the start window for a 3-hour shift is 4 am to 6:30 pm, and start window for a 3.5 hour length shift is 4 am to 6 pm, and so on. Even in the case of SIP, no shift can start from 10 pm to 8 am (from Fig. 2.1 it can be observed that there is no demand during this period).

The traditional approach to handle ISTSP with flexible schedules is to use implicit formulations and iterative methods using column generation techniques. In Volland et al. (2017b), a staff schedule is implicitly formulated for the entire planning horizon combining shifts and their assignment. In order to mitigate the complexity, Stolletz (2010) used a reduced set covering formulation where predetermined daily shifts are implicitly embedded in the planning horizon. In this paper, we reduce the complexity further. We present a two-stage approach to solve this problem directly without using implicit formulations for shift patterns, iterative procedures and the column generation techniques. We achieve this by using shift patterns as the key to the entire planning. We first define a shift pattern formally.

What is a shift Pattern?

A shift pattern of length m is a binary m-vector that satisfies all the shift constraints such as $SL_{min} \leq m \leq SL_{max}$ and the shift break period rules. We shall denote a shift pattern by $\mathbf{s} = (s_1, s_2, \ldots, s_m)$.

What is a shift schedule?

A *shift schedule*, denoted by $(\mathbf{s}, t)$, is a combination of a shift pattern $\mathbf{s}$ and a TP t. A shift schedule is used to specify that a worker who is assigned $(\mathbf{s}, t)$ must start a fresh shift at TP t and work according to shift pattern $\mathbf{s}$. The t in $(\mathbf{s}, t)$ may be specified relative to a day (in this case, t ranges from 1 to 48 with $\omega = 30$) or relative to the entire planning horizon (in this case, t ranges from 1 to 336 with $\omega = 30$). In our models in

this paper, t is relative to the entire planning horizon.

Stolletz (2010) used shift schedules relative to day and he has 330 of them. The shift patterns (embedded in his shift schedules) have only 1s as their coordinates (as no break periods are considered) and their lengths vary from 6 to 20. Similarly, in the model used by Volland et al. (2017b), there are 25 underlying shift patterns containing only 1s as their coordinates (as no break periods are considered). For SIP, we have 260 shifts patterns because we consider tea and lunch breaks. Each of these patterns can be described using shift patterns of length 18 with exactly fourteen 1s, two consecutive 0s and two 0.5s. For example, $\mathbf{s} = (1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0.5, 1, 1, 0.5, 1, 1, 1, 1)$. The two 0s ($s_5$ and s_6) stand for a lunch break and the two 0.5s (s_{11} and s_{14}) stand for the two tea breaks[1]. The numbers 0, 0.5 and 1 are the proportions of a TP that a worker is available. It must be noted that these shift patterns can be generated implicitly through ILP formulations but that becomes very complicated. Instead, we can use a simple computer program to generate all the shift patterns effortlessly as we did for this problem.

We are now ready to present our two-stage solution method for ISTSP. The basic idea is that we first determine the shift schedules in stage 1, and assign them to workers in stage 2. The stage 1 problem is described in Section 2.3.1. and stage 2 in Section 2.3.2.

2.3.1 Shift Pattern Subproblem - Stage 1

In this stage we consider two sets of decision variables. The first set of decision variables assigns the TPs to task starting times. The second set of variables decide the number of shift patterns assigned to TPs so as to meet the required workforce demands. These decisions yield the supply of workforce in each TP, and the two sets of decision variables are linked through supply-demand constraints. The objective function of the problem will be taken as the cost of shifts.

[1]Firstly, we are abusing the definition of shift pattern by allowing the fraction 0.5. This is only done to handle breaks of half TP. This will not cause any hinderance in solving the problems using methods of this paper. Next, it might appear to violate the condition that the gap between two successive breaks must be at least 90 minutes. Note that this can still be upheld by allowing the first tea break in the first 15 minutes of the corresponding TP and allowing the 2nd tea break in the last 15 minutes of the corresponding TP.

Let $\mathcal{S} = \{\mathbf{s}^i : \ i = 1, 2, \ldots, q\}$ be the set of all shift patterns and let m_i be the length of $\mathbf{s}^i, \ i = 1, 2, \ldots, q$. Let y_{kj} be 1 if task k starts in TP j, and equal to 0 otherwise. Let x_{ij} be the number of shift schedules $(\mathbf{s}^i, j)$, $i = 1, 2, \ldots, q$ and $j \in \mathcal{T}$.

The task assignment $Y = (y_{kj})$ induces a demand vector $\mathbf{R} = (R_1, R_2, \ldots, R_T)^t$, where R_j is the number of workers required in TP j. The expression for R_j is given by

$$R_j = \sum_{k=1}^{K} \sum_{i=1}^{d_k} r_{ki} y_{k\theta(j-i+1)}, \tag{2.3.1}$$

where $\theta(\cdot)$ is the *wrap function* for the cyclic time horizon, that is, $\theta(0) = T, \ \theta(-1) = T-1, \ldots,$ and $\theta(T+1) = 1, \ \theta(T+2) = 2, \ldots$.

Similarly, $X = (x_{ij})$ induces a supply vector $\mathbf{S} = (S_1, S_2, \ldots, S_T)^t$, where S_j is the number of workers available in TP j. The expression for S_j is given by

$$S_j = \sum_{i=1}^{q} \sum_{t=1}^{m_i} s_t^i x_{i\theta(j-t+1)}, \tag{2.3.2}$$

where $\mathbf{s}^i = (s_1^i, \ldots, s_{m_i}^i)$ and $\theta(\cdot)$ is the wrap function defined above.
Let c_{ij} be the cost of shift schedule $(\mathbf{s}^i, j)$. Then our **stage 1 problem** for ISTSP, is given by

$$\texttt{Minimize} \quad \sum_{i=1}^{q} \sum_{j=1}^{T} c_{ij} x_{ij} \tag{2.3.3}$$

`subject to`

$$\sum_{i=1}^{q} \sum_{t=1}^{m_i} s_t^i x_{i\theta(j-t+1)} \geq \sum_{k=1}^{K} \sum_{i=1}^{d_k} r_{ki} y_{k\theta(j-i+1)}, \ \texttt{for} \ j = 1, 2, \ldots, T, \tag{2.3.4}$$

$$\sum_{j=1}^{T} (j + d_k - 1) y_{kj} \leq \sum_{j=1}^{T} j y_{k'j} \ \texttt{for all} \ (k, \ k') \in \mathcal{P}, \tag{2.3.5}$$

$$\sum_{j=1}^{T} y_{kj} = 1 \ \texttt{for} \ k = 1, 2, \ldots, K, \tag{2.3.6}$$

$$\sum_{j=1}^{l_k - 1} y_{kj} + \sum_{j=u_k+1}^{T} y_{kj} = 0, \quad \texttt{for all} \ k \tag{2.3.7}$$

$$y_{kj} \in \{0, \ 1\} \ \texttt{for all} \ i, \ j, \tag{2.3.8}$$

$$x_{ij}\texttt{s are nonnegative integers for all} \ i, j. \tag{2.3.9}$$

Above, (2.3.4) is the supply-demand constraints, (2.3.5) takes care of the precedence relationships, (2.3.6) and (2.3.7) ensure that all tasks start in their designated start windows $[l_k,\ u_k]$[2]. The objective function is the total cost of assigned shifts.

2.3.2 Staff Assignment Problem - Stage 2

From stage 1 solution, we have the shift schedules that will meet the staff demand requirements satisfying the shift constraints. We now assign these shift schedules to workers, maintaining staff scheduling constraints involving minimum/maximum number of shifts/hours per worker, days-off per worker, etc. The stage 2 formulation requires preparation of inputs. This process will be described first.

From stage 1 output, collect all shift schedules $(\mathbf{s}^i, j)$ for which $x_{ij} > 0$ and sort them according to the ascending order of j. The number of such shift schedules is $\tau = \sum_{ij} x_{ij}$. Let $\mathcal{U}_1 = (s^{i_1}, j_1),\ \mathcal{U}_2 = (s^{i_2}, j_2),\ \ldots,\ \mathcal{U}_\tau = (s^{i_\tau}, j_\tau)$ be the τ shift schedules. Note that $j_1 \leq j_2 \leq \ldots \leq j_\tau$, and a shift schedule $(\mathbf{s}^i,\ j)$ with corresponding x_{ij} is repeated x_{ij} times in the list.

Choose a large positive integer w representing maximum number of workers available for scheduling during the planning horizon. Label the workers as $1, 2, \ldots, w$. Define the decision variables of stage 2 as follows: $z_{uv} = 1$ if worker u is assigned shift schedule $\mathcal{U}_v$, $z_{uv} = 0$ otherwise, $u = 1, 2, \ldots, w,\ v = 1, 2, \ldots, \tau$.

In order to meet the supply-demand constraints, we must assign each of the τ shift schedules to workers. Note that $f(v) = \sum_{u=1}^{w} u z_{uv}$ is worker label to which shift schedule v is assigned to. Therefore, stage 2 objective function is $\max_v f(v)$ which we minimize to minimize the number of workers. This objective function is linearized by introducing a dummy variable ξ and a constraint as follows.

$$\texttt{Minimize} \quad \xi \tag{2.3.10}$$

$$\texttt{subject to}$$

$$\sum_{u=1}^{w} u z_{uv} \leq \xi, \quad v = 1, 2, \ldots, \tau. \tag{2.3.11}$$

[2]In the actual implementation of the model for solving the problem, y_{kj}s were defined only for $l_k \leq j \leq u_k$.

Next, we formulate the constraints of stage 2 problem.

Assignment Constraints: Each of $\mathcal{U}_1$ to $\mathcal{U}_\tau$ must be assigned to workers. This translates to

$$\sum_{u=1}^{w} z_{uv} = 1 \texttt{ for } v = 1, 2, \ldots, \tau. \tag{2.3.12}$$

Maximum number of shifts: Suppose a worker can have at most b shifts in the planning horizon. This translates to

$$\sum_{v=1}^{\tau} z_{uv} \leq b, \texttt{ for } u = 1, 2, \ldots, w. \tag{2.3.13}$$

Rest period: Rest period, the gap between any two successive shifts assigned to a worker, must be at least g TPs. For this, we first define a overlapping pair of shift schedules $\mathcal{U}_v$ and $\mathcal{U}_{v'}$. For $v < v'$, say that $\mathcal{U}_v$ and $\mathcal{U}_{v'}$ are overlapping if $j_{v'} \leq j_v + m_{i_v} - 1 + g$. Call (v, v') an overlapping pair if $\mathcal{U}_v$ and $\mathcal{U}_{v'}$ are overlapping, $1 \leq v < v' \leq \tau$. To ensure rest period, any worker can be assigned at most one of $\mathcal{U}_v$ and $\mathcal{U}_{v'}$ if (v, v') is a overlapping pair. This translates to

$$z_{uv} + z_{uv'} \leq 1 \texttt{ for every } u \texttt{ and every overlapping pair } (v, v'). \tag{2.3.14}$$

Total Hours: The total time of any worker must not exceed H TPs. Note that $\sum_{v=1}^{\tau} m_{i_v} z_{uv}$ is the total duration (m_{i_v} =number of TPs in shift i_v), in number of TPs, of worker u over the planning horizon. Therefore, the constraints are

$$\sum_{v=1}^{\tau} m_{i_v} z_{uv} \leq H, \quad u = 1, 2, \ldots, w. \tag{2.3.15}$$

Days-Off: We shall assume that day-off must start from the first TP of a day. In a 5-day week, a worker must get two consecutive days off. We shall formulate the constraints taking $\omega = 30$ and one week planning horizon (we can imitate the same for other values of ω). Constraints for a 6-day week can be derived in a similar fashion. A two-day period can be represented by the shift pattern $\bar{\mathbf{s}}$ with $\bar{s}_i = 1$ for $i = 1, 2, \ldots, 96$. Introduce dummy shift schedules $\mathcal{U}_v = (\bar{\mathbf{s}}, j_v)$, $v = \tau + 1, \ldots, \tau + 7$, where $j_1, j_2, \ldots, j_7$ are the starting TPs of days 1 to 7 respectively (i.e., $j_1 = 1$, $j_2 = 49$, $j_3 = 97$, and so on). With an abuse of convention, we

shall interpret the dummy shift schedules as days-off. That is, $z_{u(\tau+1)} = 1$ will be interpreted as worker u having first two days of the week off. Interpret $z_{u(\tau+2)} = 1$ as 2nd and 3rd days of the week off, and so on. With this, the two-days-off constraints for worker u, $u = 1, 2, \ldots, w$, can be written as

$$\sum_{v=\tau+1}^{\tau+7} z_{uv} = 1, \quad \texttt{and for every overlapping pair}(v, v') \tag{2.3.16}$$

$$z_{uv} + z_{uv'} \leq 1, \quad \texttt{where } 1 \leq v \leq \tau, \quad \tau + 1 \leq v' \leq \tau + 7. \tag{2.3.17}$$

Thus, constraints for stage 2 problem can be picked from (2.3.12) to (2.3.17) depending upon the context, and perhaps can be augmented with some more if necessary.

Optimality of solutions obtained by the **two-stage method** (TSM) depends upon the nature of objective function. The following theorems are useful in this regard.

Theorem 2.3.1. *If the objective function of ISTSP is a function of shift schedules, then the two-stage method produces an optimal solution.*

Theorem 2.3.2. *If the objective of ISTSP is to minimize the number of workers with one of the constraints as (2.3.13), then* $\lceil \frac{B}{b} \rceil$ *is a lower bound for the number of workers, where* B *is any lower bound for stage 1 objective function, the number of shift schedules assigned.*

One of the factors that appears to have a significant bearing on the solution time of ISTSP is the total demand $\sum_j R_j$. In the existing literature, the problem instances are classified as small, medium and large based on this factor. We introduce a *split technique* to handle problems with large demands. It has a cascading effect on reducing the solution time of ISTSPs with large demands.

2.3.3 The Split Technique

Consider an ISTSP and suppose that $\mathbf{R}$ is a demand vector that is optimal or near optimal. We split the demand vector $\mathbf{R}$ into sum of two new demand vectors $\mathbf{R}^1$ and $\mathbf{R}^2$ so that $\mathbf{R} = \mathbf{R}^1 + \mathbf{R}^2$. Then, we solve two new subproblems with fixed demand vectors $\mathbf{R}^1$ and $\mathbf{R}^2$ separately using the two-stage approach and combine the solution to get a

solution to the original problem. If w^1 and w^2 are optimal (or near optimal) objective values of the two subproblems, then we have a solution for the ISTSP with $w^1 + w^2$ workers. We shall explain this approach with the help of some examples.

One of the problem instances (corresponds to P4 in Table 2.3) is a problem with fixed $\mathbf{R}$ (no task scheduling) and has a total demand 3736 worker-hours. Solving this using two-stage method, stage 1 was solved to near optimality in 32 seconds with a lower bound of 499; but stage 2 got abandoned due to insufficient memory. Then, we solved the two subproblems taking $R^1_j = \lfloor \frac{R_j}{2} \rfloor$ and $R^2_j = R_j - R^1_j,\ j = 1, 2, \ldots, 336$. The resulting subproblems have demands 1836 (for $\mathbf{R}^1$) and 1900 (for $\mathbf{R}^2$). Solving these two problems using two-stage method yielded the following results. The $\mathbf{R}^1$-subproblem resulted in a near optimal solution (in 201 seconds) with 52 workers, and the $\mathbf{R}^2$-subproblem resulted in a near optimal solution (in 198 seconds) with 54 workers. Combining the solutions of the two subproblems, we have a solution to the original problem with 106 workers. From Theorem 2.3.2, 100 ($= \lceil \frac{499}{5} \rceil$) is a lower bound for the problem. Therefore, the solution with 106 workers is at least 94% ($= 100 - \frac{106-100}{100} \times 100$) optimal, and the problem is solved in less than 7 minutes.

How to solve faster?

Consider a case where the total demand is so large that even after splitting the demand vector, we still have a problem. Even for such cases, we solve only two subproblems to get a solution. Consider the problem with a total demand of 5136 worker-hours (see P20 in Table 2.3). For this problem, we take $R^1_j = \lfloor \frac{R_j}{3} \rfloor$ and $R^2_j = R_j - \frac{2R^1_j}{3},\ j = 1, 2, \ldots, 336$. With this, the demands for R^1 and R^2 subproblems are 1669 and 1798 worker-hours respectively. Note that $\mathbf{R} = 2\mathbf{R}^1 + \mathbf{R}^2$. Solving the two subproblems, we found a solution for $\mathbf{R}^1$-subproblem with 57 workers, and for $\mathbf{R}^2$-subproblem with 58 workers. To obtain a solution for the original problem, we apply the solution of $\mathbf{R}^1$-subproblem to two sets of 57 workers each, and apply the solution of $\mathbf{R}^2$-subproblem to another set of 58 workers. The resulting allocation is a solution to the original problem with 172 ($= 2 \times 57 + 58$) workers. To find the optimality percentage, we use the lower bound of the stage 1 problem with original demand vector. For the instance in question, the stage 1 problem with the original demand vector with demand of 5136 worker-hours produced an optimal solution (in 2 seconds) with 852 shift schedules. From Theorem 2.3.2, the

number of workers is at least 171, and hence the solution obtained using split technique is at least 98.8% optimal. The whole process took 6 minutes and 22 seconds.

Consider another instance with a total demand of 5615 worker-hours (P21 in Table 2.3). Splitting $\mathbf{R} = 2\mathbf{R}^1 + \mathbf{R}^2$ with $\mathbf{R}^1 = \lfloor \frac{\mathbf{R}}{3} \rfloor$ and solving this problem took 10 minutes 18 seconds. The number of workers in this case is 183 and the lower bound from stage 1 solution is 181 (stage 1 took 2 seconds). Taking $\mathbf{R} = 3\mathbf{R}^1 + \mathbf{R}^2$ with $\mathbf{R}^1 = \lfloor \frac{\mathbf{R}}{4} \rfloor$ and solving this problem (P22) took only 4 minutes 2 seconds. The number of workers in the resulting solution is 182. In general, we can use $\mathbf{R}^1 = \lfloor \frac{\mathbf{R}}{\rho} \rfloor$, where the ***splitting factor*** $\rho > 1$. Choosing large ρ will reduce the solution time but will affect the optimality. Therefore, we should choose ρ judiciously.

Remark 2.3.1. *Under the split technique, we solve only two subproblems with demand vectors* $\mathbf{R}^1$ *and* $\mathbf{R}^2$ *and use the solutions to derive a solution to the original problem.*

Remark 2.3.2. *The split technique is found to be very effective in solving problems with large demands. However, this method requires the demand vector* $\mathbf{R}$*. For problems of ISTSP, the optimal demand vector is to be obtained first in order to apply the split technique. It must be noted that stage 1 of our approach produces optimal or near optimal demand vector* $\mathbf{R}$ *very efficiently even for the cases where the demand is very high (see Table 2.3). Thus, our two stage approach clubbed with the split technique (if needed) can solve ISTSPs even with large demands very efficiently.*

Based our empirical experience, we make the following hypothesis and wish to explore it in future.

Hypothesis: The total demand does not appear to be a factor that affects the complexity of ISTSP.

2.4 Real-Life Instances and Numerical Experiments

In this section we assess the performance of the two-stage approach clubbed with split technique (where necessary) with a number of live and simulated instances. For this, we consider two categories of problems. The first one corresponds to the type of problems considered in Stolletz (2010) and Brunner and Stolletz (2014) where the tasks are

already scheduled and we have a demand vector $\mathbf{R}$ as input to the problem. The second category of problems corresponds to the type of problems dealt with in Volland et al. (2017b) where both tasks and shifts have to be scheduled, that is, proper ISTSPs. The real-life instances for the first category are taken from requirements from software industry, airport check-in counter staff requirements and call center data. For the second category of problems, we use simulated data. For the purpose of comparison, the data are simulated using the distributions specified in Volland et al. (2017b) as well as some new distributions. We also have real-life data from emergency medical services (108 service in India) for this category. For the first category of problems, we consider $\omega = 30$ and $T = 336$, and for the second category, $\omega = 15$ and $T = 672$. All problems are treated with cyclic planning horizon.

Types of shift patterns

For our numerical experiments, we considered four types of shift patterns described below. The first three of them are used in problems with $\omega = 30$ and $T = 336$. The fourth one, FX29, is used in problems with $\omega = 15$ and $T = 672$.

FX260 Under this, all shifts have fixed duration of 9 hours (18 TPs of length $\omega = 30$) with breaks satisfying conditions (i) to (iv) stated at the beginning of Section 2.2. The number 260 is the number of shift patterns under FX260.

FL15 There are 15 patterns under this with durations varying from 3 hours to 10 hours. Relief breaks are incorporated at appropriate positions depending on duration of the shift (3-5h: no break, 5.5-6h: one 15-minute break, 6.5-8h: one 30-minute break, 8.5-10h: two 15-minute and one 30-minute breaks).

FL135 Brunner and Stolletz (2014) considered lunch breaks in the shift patterns. The duration of the shifts varies from 3 to 10 hours with exactly one 30-minute break with the condition that no break in the first one hour and in the last one hour of the shift. There are 135 such shift patterns.

FX29 These are shift patterns with durations varying from 3 hours to 10 hours without breaks. These are the 29 patterns considered in Volland et al. (2017b) for their numerical experiments.

In all, we solved 40 instances (see tables 2.3 and 2.4). All problems are solved using the LINGO professional solver Version 13.0 on an i7 64-bit processor with 2.80GHz clock speed and 16 GB RAM running on a Windows 10 platform. Unless specified otherwise, the objective for all the problems is taken as minimizing the number of workers. The instances are described in the following subsections. Their results are discussed in Section 2.5.

2.4.1 The Software Industry Problem

The background of this problem was described at the beginning of Section 2.2. For this problem, $\omega = 30$, $T = 336$, $q = 260$, $m_i = m = 18$ for $i = 1, 2, \ldots, q$. This problem is similar to the discontinuous tour scheduling problem of Stolletz (2010). There is no demand during the periods 10pm to 8am on all days. For an instance of this problem, the total demand is 1258 worker-hours (see P1 in Table 2.3), and the number of workers required over TPs varies from 0 to 25 (see Fig. 2.1) with an average of 6.4 workers per TP. The problem is solved using FX260 shift patterns without imposing any restriction on the start times of shift patterns. Additional restrictions imposed on the problem are: (i) at most 5 shifts per worker and (ii) at least 12 hours gap between any two successive shifts of any worker. For simplicity, we have not imposed the day-offs to be consecutive. The stage 1 problem was solved in 11 seconds with an optimum objective value of 220 shift schedules. Stage 2 problem was solved in 77 seconds with the optimum objective value of 44 workers. Since this objective value is equal to the lower bound ($44 = \lceil \frac{220}{5} \rceil$), the solution is optimal for the problem.

2.4.2 Airport Check-in Counter Requirement Problem

In this problem, we consider agent requirements to man the check-in counters. Airline departures over a season (spanning about 6 months) are planned in advance based on weekly roster. The number of counters allocated to each airline varies over time depending on the departures of that airline. In Lalita et al. (2020), these requirements were worked out for various airlines' schedules from a major international airport in India. The weekly departures of an airline gives rise to agent requirements over the

week. Taking domestic and international departures separately of three airlines, coded as JAW, AAW and BAW, we formed five demand vectors for JAW-I, JAW-D, BAW-I, BAW-D and AAW-D. From these, we derived 10 instances by combining them with the three shift pattern types FX260, FL15 and FL135, objective function type and the type of constraint on the worker load. These 10 instances correspond to P2 to P15 in Table 2.3. For example, P2 instance is formed by taking the departures of BAW-I, FX260 shift patterns, worker-load constraint as the maximum number of shifts per worker in the week and the objective function as the number of workers. For details of other instances, see the note under Table 2.3. P7 to P10 correspond to the instance but solved under different constraints and different objective functions (see Fig 2.3). The demands vary from 608 to 3752 agent-hours (see Table 2.3). Unlike the discontinuous tour scheduling problem considered in Stolletz (2010), P2 to P15 have continuous requirements over the planning horizon.

We shall discuss the results of our approach of solving two instances - P4 and P5 with demands 3736 and 1467 agent-hours respectively. Assuming that the agents can start their shifts at the beginning of every half hour, the requirements were computed based on 30-minute TPs for weekly planned departures. Fig 2.2 presents the demand patterns for the two problems.

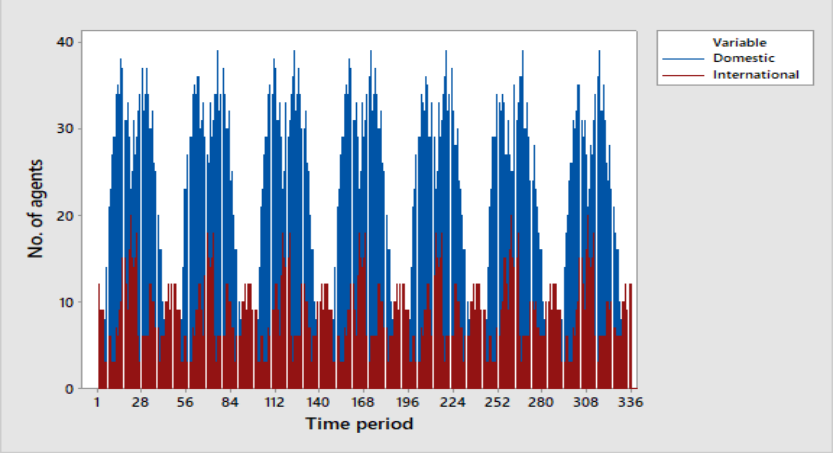


FIGURE 2.2: Agent requirements for JAW domestic and international departures

Note that Stolletz (2010) starts with 330 shift schedules (per day) and uses them in his model to obtain staff schedules over the entire planning horizon. For problems with

FX260, $\omega = 30$ and $T = 336$, there are 12480($= 260 \times 48$) shift schedules per day. Brunner and Stolletz (2014) observed that with one flexible lunch break, the number of shift schedules (per day) rises to 2690 and point out that they were not able to solve the problem with their model using standard MIP software. The major difference between the two problems (P4 and P5) and the discontinuous tour scheduling problem considered in Stolletz (2010) is the continuous requirement of agents over the planning horizon. With FX260 patterns and continuous requirements over one week cyclic planning horizon, the number of possible shifts combinations per worker is approximately 10^{19}. The results of solving the two problems, P4 and P5, are summarized below.

Instance P5

The total demand for this problem is 1467 agent-hours. The stage 1 model for this problem produced a solution with 227 shift schedules in 59 seconds with a lower bound of 224 on the objective function. The best objective value remained at 227 even after five minutes CPU running time. We aborted the solver and took the solution with 227 shift schedules and solved stage 2 problem. This produced an optimal solution in 78 seconds with 46 agents. Applying Theorem 2.3.2 on the lower bound 224, the number of agents must be at least 45. Therefore, solution obtained for this problem with 46 agents is at least 97.7% optimal.

Instance P4

Recall the discussion about this problem under the introduction of split technique. While solving stage 2 model for this problem, the solver aborted the solution process reporting insufficient memory. We observed this phenomenon whenever we tried to solve stage 2 problem with huge demand. The reason is that high demand requires large number of shift schedules which in turn results in large number of overlapping shift schedule pairs. As a result, the number of constraints under (2.3.14) increases dramatically (for this problem the number of constraints is 5054445 and the number of variables is 87041). Applying the split technique, this problem was solved in less than 7 minutes and the optimality gap is at most 6%.

Instances with cost objective and work load constraints

Since there are shifts with short lengths, we solved stage 2 problem once with the constraint on the number of shifts per week per worker (maximum 5 shifts) and once with the constraint on maximum number of worker-hours per week per worker (maximum 50 hours). Again, with respect to objective function, we have two options, number of workers and cost. Thus, we have four combinations which are represented by instances P7 to P10 (see Fig 2.3). For simplicity, cost is taken as a function of shift duration alone. We took the costs as shown in Fig 2.3.

Cost objective function parameters

	Shift duration (hours)		
	Less than 8	Between 8 and 9	Above 9
Cost	1.5	1	1.75

Types of instances

		Objective	
		Number of workers	Cost
Constraint type	Shifts	P7	P9
	Hours	P8	P10

FIGURE 2.3: Parameters of instances P7 to P10

2.4.3 Call Center Data

We have data on number of agents worked, hour-wise, for 36 weeks from a call center. Like in check-in counters problem, the requirement of agents is round the clock. We combined two weeks (14 days) data to form an instance. For simplicity, we treat the hourly requirements as requirements for 30 minute periods. Data are taken from two different streams with total demand varying from large to very large. There are six instances, P16 to P21 in Table 2.3. The variation in the demand pattern is shown in Figure 2.4. The total demands vary from 2155 to 5615 agent-hours. As demands are high, all the six instances had to be solved using split technique. The results are summarized in Table 2.3. The solutions to these instances demonstrate the efficacy of the split technique. Since P21 took a long time (10 minutes and 18 seconds), we solved this problem again (P22) with $\mathbf{R} = 3\mathbf{R}^1 + \mathbf{R}^2$ and $\mathbf{R}^1 = \lfloor \frac{\mathbf{R}}{4} \rfloor$ (see Section 2.3.3). As a result, the problem could be solved in less than 5 minutes.

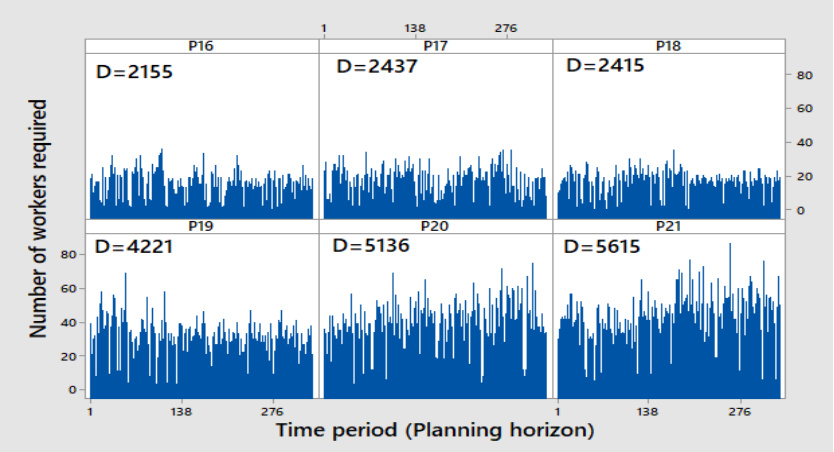


FIGURE 2.4: Agent requirements for call center problems

2.4.4 Instances for ISTSP

In this section we describe the data for instances on ISTSP which requires task scheduling as well. We have one real-life data set from emergency medical services (108 service). For the other instances, P23 to P40 of Table 2.4, data are simulated following Volland et al. (2017b). All instances under this case are solved assuming FX29 patterns.

Medical emergency data (P41)

We have historical data on the 108 service pertaining to a province in Andhra Pradesh, India. The service brings patients needing emergency medical care to a hospital. The most commonly reported emergencies (about 70% of the cases) are related to pregnancy, acute abdomen, trauma (vehicular), fevers (infections) and cardiac/cardio vascular issues. Of these, pregnancy cases alone accounted for 23%. Therefore, we took data (number of patients arriving in every 15 minutes) on pregnancy cases for one week (7 consecutive days) of a month. There were 588 such cases. We took the duration of redressal of these cases (tasks) to the nearest 15 minutes, and used the seriousness of the cases to set the start windows and the precedence relations. We took the earliest start time of a task as the arrival TP of the patient, and set the latest start time based on the seriousness of the case. The resulting instance has the following characteristics: $K = 588$, $T = 672$, $\omega = 15$, maximum width of start window of task that is not involved in precedence relationships is 4ω, total demand is 1151 worker-hours ($\sum_{j=1}^{672} R_j = 4603$); number of tasks involved in precedence relationships is 116 with a total of 113 precedence relationships. The demand pattern is shown in Fig. 2.5.

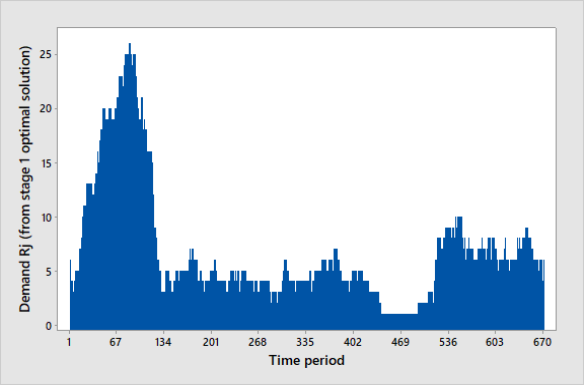


FIGURE 2.5: Staff requirements for medical emergency problem

The stage 1 model for this problem produced a near optimal solution with objective value of 118 shift schedules in 119 seconds with a lower bound of 116. We terminated stage 1 at this time and solved stage 2 problem which produced an optimal solution in 235 seconds with optimum objective value of 37 workers. Applying Theorem 2.3.2 on the lower bound 116, the minimum number of workers is at least 24. In order to find a better lower bound, we took the stage 1 objective function as $\max_j R_j$ and minimized it. The optimum objective value for this was 26. Therefore, the number of workers for this problem cannot be less than 26. We then solved stage 1 problem once again with the original objective function but this time by adding an additional constraint $\max_j R_j \leq 26$. This resulted in the same objective value and lower bound as before (118/116) but the solution was different. The resulting solution was used to solve stage 2 problem, and that produced a global optimum objective value of 36 workers. Thus, the final solution was at least 62% $(= 100 - \frac{36-26}{26} \times 100)$ optimal.

Simulated data

We simulated data for ISTSP following the procedure described in Section 5.2.2 of Volland et al. (2017b). Under this procedure, three types of tasks (day long, peak and precedence) and three problem sizes (small (600 hours), medium (1000 hours) and large (1400 hours)) are considered. For each size, three different distributions of task types (S1, S2 and S3) were used. The parameters for simulation are summarized in Table 2.2. Thus, there are nine scenarios under this situation. We simulated nine instances, P23 to P31 of Table 2.4, following the procedure for these nine parameter settings. The

corresponding instances from Volland et al. (2017b) are listed in Table 2.5. We ignored the additional instances considered by Volland et al. (2017b) (presented in Table 5 of their paper) because those instances are more restricted (either shift patterns are limited to two or start window lengths are reduced by 50%). We simulated the nine instances using the same shift patterns used in Volland et al. (2017b), namely FL29 shift patterns.

TABLE 2.2: Parameters for simulation of instances for ISTSP

Demand(x) (in hours)	Tasks and their distribution types								
	S1			S2			S3		
	Day long	Peak	Precedence	Day long	Peak	Precedence	Day long	Peak	Precedence
600	83%	15%	2%	79%	15%	6%	53%	45%	2%
1000	83%	15%	2%	79%	15%	6%	53%	45%	2%
1400	83%	15%	2%	79%	15%	6%	53%	45%	2%
Start windows	6am - 10am	6am - 8 am 10am - 12 pm 2pm - 4pm	Mon 6am to Fri 1pm	6am - 10am	6am - 8 am 10am - 12 pm 2pm - 4pm	Mon 6am to Fri 1pm	6am - 10am	6am - 8 am 10am - 12 pm 2pm - 4pm	Mon 6am to Fri 1pm
Duration range	6 to 8 hrs	1 to 2 hrs	2 to 4 hrs	6 to 8 hrs	1 to 2 hrs	2 to 4 hrs	6 to 8 hrs	1 to 2 hrs	2 to 4 hrs

Notes: 1. If x is the number of hours per week, then the load on week days (Mon to Fri) is x/6.4 hrs and 0.7x/6.4 on week ends.
2. Work hours to peak tasks in three start windows - early (6am - 8 am), mid day (10am - 12pm), and late (2pm -4pm) are equally distributed.

Additionally, we created nine more instances by considering three more distributions for the three types of tasks, say S4, S5 and S6. These distributions are $(81, 17, 2)$, $(79, 17, 4)$ and $(72, 16, 12)$. These are the resulting distributions if we apply the S1, S2, S3 distributions to number of tasks instead of applying them to number of hours. These additional 9 instances are P32 to P40 in Table 2.4. Besides the differences in the distributions, one major difference between the two sets, P23 to P31 and P32 to P40, is that in the latter the task start time windows of all tasks have been chosen uniformly throughout the days. However, we have not changed the characteristics of the start window widths and task durations.

2.5 Summary of Experimental Results

In this section we shall present the results of our numerical experiments. We have solved 39 problem instances and the results are summarised in Tables 2.3 and 2.4. Table 2.3 presents the results of problems with fixed demand vector where task scheduling is not required. These problems are similar to the ones considered in Stolletz (2010) and Brunner and Stolletz (2014). Table 2.4 presents the results for problems with task scheduling requirements involving precedence relationships. These problems are similar

to the ones considered in Volland et al. (2017b). The parameters affecting the complexity of ISTSP are: (i) the length of planning horizon T, (ii) number of tasks, K, (iii) demand and its pattern (d_is, r_{ik}s), (iv) number of precedence relationships and (v) number of shift patterns. The range of these parameters in our instances are such that the results can be compared with the results of the respective papers mentioned above.

To assess the merit of any solution, we consider four parameters: the total demand, solution time, optimality metric and utilization metric. For problems where task scheduling is involved, one should also look at the number of tasks involved in the precedence relationships and the number of precedence relations. Total demand, expressed as total number of worker-hours required, is equal to $(\sum_j R_j)\omega/60$. For any solution with objective value O_s and lower bound O_L, the percentage optimality gap is at most $\frac{O_S - O_L}{O_L} \times 100$. Therefore, we take $\mu = 100 - \frac{O_S - O_L}{O_L} \times 100$ as the measure of optimality. Utilization metric is taken as 100 times the ratio of total demand to total supply. Stage 1 model plays a crucial role in our solution approach. We shall first discuss the results with respect to stage 1 problems.

2.5.1 Results for stage 1 model

In order to apply split technique for large demands in the case of ISTSP, solving stage l model efficiently is crucial (see Remark 2.3.2). Fortunately, our experiments show that stage 1 model is solved very efficiently despite the fact that it is more complex in the case of problems involving task scheduling with precedence relationships compared to those for which the demand vector is an input. Fig.2.6 presents the stage 1 model performance. All solutions are at least 96% optimal (65% are 100% optimal), and found in less than two minutes (with one exception which took 228 seconds). Average demand is 2117 worker-hours. It should be noted that the high demand instances took smaller times (see tables 2.3 and 2.4).

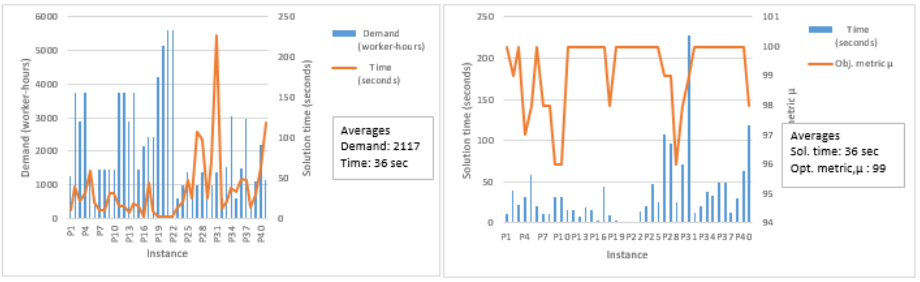


FIGURE 2.6: Performance of stage 1 model

2.5.2 Results of problem instances with given demand vector

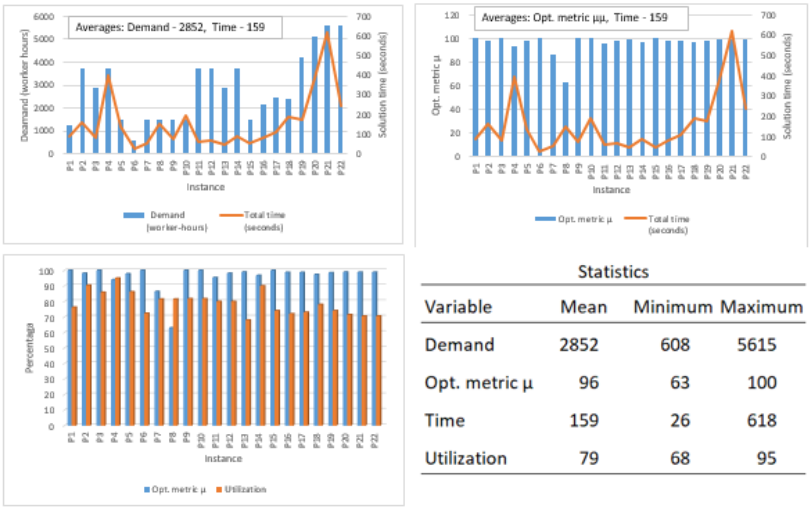
Instances of P1 to P22 are under this category. For each of these instances, $\omega = 30$, $T = 336$ and the demand varies from 608 to 5615 worker-hours. For all instances with demand (number of worker-hours) less than 1500, we could get solution directly. For the other instances, minimum demand is above 2000. For these instances, the problems had to be solved using the spit technique (see the discussion under Instance P4 on page 46). The method used ('Direct' or 'Split(ρ)') is specified in Table 2.3. Instance P11 is solved twice with $\rho = 3, 4$. In both cases, the solutions are near optimal (95% and 98%), and the solution times are also close (75 and 81 seconds). Similarly, P21 was solved twice with $\rho = 3,\ 4$. Split(3) took 618 seconds and split(4) took 242 seconds. In both cases, the solutions are at least 99% optimal. The necessity for splitting is arising from large demand. To highlight this, the number of variables and constraints of stage 2 model with the original demand vector are presented in Table 2.3. From the table, it can be seen that for the instances solved with split technique, the number of constraints ranges from 1.7 millions to 16.8 millions. The performance metrics of two stage method (with split technique where needed) as applied to instances of P1 to P22 are presented in the last three columns of Table 2.3 and in Fig.2.7. In all but two of the instances, the optimality was at least 94%. In one case, P7, it is 86% and in the other case, P8, it is 63%. The optimality metric μ for P8 is computed using a poor lower bound, namely total demand by the maximum number of hours that a worker can be assigned (recall that P8 constraints are based on maximum number of hours and not the number of shifts, see Fig.2.3). The average solution time is 2 minutes 40 seconds and the average utilization is 79%.

TABLE 2.3: Results for staff scheduling problem instances

Problem Characteristics				Stage 1			Stage 2		Overall		Performance Metrics		
Instance ID	Source	Demand (worker-hours)	Shift patterns type	Best Objective	Lower bound	Time (seconds)	No. of variables	No. of constraints	No. of workers	Technique	Optimality metric μ	Total time (seconds)	Utilization
P1	SIP	1258	FX260	220	220	11	16061	366609	44	Direct	100	88	76
P2	BAW-I	3752	FX260	545	542	40	98646	6058256	111	Split(3)	98	162	90
P3	BAW-D	2873	FX260	446	446	22	66009	3249937	90	Split(3)	100	81	86
P4	JAW-D	3736	FX260	512	499	32	87041	5054445	106	Split(2)	94	399	95
P5	JAW-I	1467	FX260	227	224	59	17026	454730	46	Direct	98	137	86
P6	AAW-D	608	FX260	112	112	20	4145	49916	23	Direct	100	26	72
P7	JAW-I	1467	FL15	220	217	10	20173	570147	50	Direct	86	56	81
P8	JAW-I	1467	FL15	220	217	10	20173	570147	41	Direct	63	150	81
P9	JAW-I	1467	FL15	246	238	31	20173	570147	50	Direct	100	78	82
P10	JAW-I	1467	FL15	246	238	31	20173	570147	42	Direct	100	193	82
P11	BAW-I	3752	FL135	539	539	15	96482	6284695	113	Split(5)	95	62	80
P12	BAW-I	3752	FL135	539	539	15	96482	6284695	110	Split(4)	98	68	80
P13	BAW-D	2873	FL135	446	446	7	66009	3118661	91	Split(4)	99	48	68
P14	JAW-D	3736	FL135	470	470	18	73321	4234469	97	Split(4)	97	91	90
P15	JAW-I	1467	FL135	216	216	16	15553	411193	44	Direct	100	51	74
P16	CCW1	2155	FL15	343	343	3	39103	1718439	70	Split(2)	99	84	72
P17	CCW16	2437	FL15	384	384	44	49153	2349825	78	Split(2)	99	111	73
P18	CCW23	2415	FL15	360	356	9	43201	1946041	74	Split(2)	97	191	78
P19	CCFN1	4221	FL15	649	649	3	140185	11684739	132	Split(3)	98	176	74
P20	CCFN10	5136	FL15	852	852	2	241969	16476506	172	Split(3)	99	382	72
P21	CCFN18	5615	FL15	904	904	2	272105	16776696	183	Split(3)	99	618	71
P22	CCFN18	5615	FL15	904	904	2	272105	16776696	182	Split(4)	99	242	71

Note: P1 is SIP, P2 to P15 are check-in counter problems and P16 to P21 are call center problems. For all problems, P1 to P21, with the exception of P8 and P10, the worker load constraint is on the number of shifts, that is, each worker is assigned a maximum of 5 shifts; for P8 and P10, it is on the number of hours, a maximum of 50 hours per week. Similarly, for all patterns other than P9 and P10, the objective function is number of workers, and for P9 and P10, it is the cost. P11 and P12 are same instance but solved differently. Likewise, P20 and P21 are same instance but solved differently. The columns under Stage 2 present the size of the problem for the stage 2 problem with the original demand vector.

Stolletz (2010) reports the solution times for three different cases. Though our case (continuous demand) is more complex, a comparison is presented in Fig.2.8 with respect to solution times.



Statistics

Variable	Mean	Minimum	Maximum
Demand	2852	608	5615
Opt. metric μ	96	63	100
Time	159	26	618
Utilization	79	68	95

FIGURE 2.7: Performance metrics of two stage method for P1 to P17

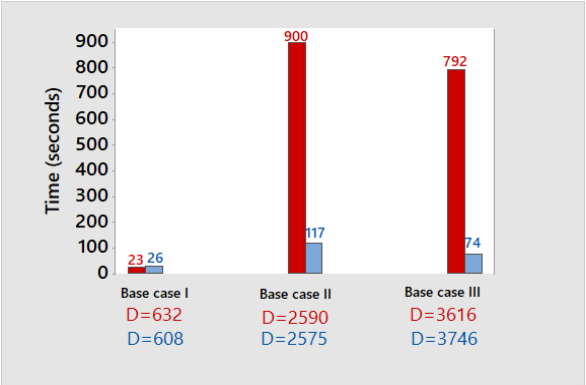


FIGURE 2.8: A comparison of solution times

2.5.3 Results of ISTSP problem instances

Instances P23 to P41 are under this category. Task scheduling is a part of the problem. Results are presented in Table 2.4. For these problems, $\omega = 15$ and $T = 672$, number of tasks K varies from 100 to 588, and the demand varies from 450 to 3032 with an average of 1266 worker-hours. Instances P23 to P40 are simulated, and P41 is based on a real-life problem. All simulated instances with the exception of P30 have been solved to optimality by the two stage method (without the need for split technique). The solution to P30 is at least 95% optimal. Utilization in the solutions varied from 75% to 98% with an average of 85%. Solution times varied from 13 to 427 seconds with an average of 111 seconds. The solution for the instance with real-life data (P41) is at least 58% optimal but the utilization is 98%. The demand for this problem is 1151

TABLE 2.4: Results for ISTSP instances

Instance ID	Demand (worker-hours)	Distribution by hours			No.of Tasks	Stage 1				Stage 2			Overall		
		Day long	Peak	Precedence		Best objective	Supply (worker-hours)	Lower bound	Time (seconds)	Best objective	Lower bound	Time	TotalTime	Optimality metric μ	Utilization
P23	595	83	15	2	136	76	692	76	14	16	16	1	15	100	86
P24	990	83	15	2	226	123	1179	123	20	25	4	4	24	100	84
P25	1384	83	15	2	316	168	1627	168	47	34	34	4	51	100	85
P26	592	77	15	6	226	72	717	72	25	15	15	1	26	100	83
P27	990	77	15	6	233	115	1124	114	108	23	23	3	111	100	88
P28	1393	77	15	6	383	163	1603	162	97	33	33	4	101	100	87
P29	588	53	45	2	230	64	637	62	25	13	13	1	26	100	92
P30	994	53	45	2	383	106	1046	104	71	22	22	3	74	95	95
P31	1388	53	45	2	536	143	1430	142	228	29	29	3	231	100	97
P32	611	78	17	4	100	85	748	85	12	17	17	1	13	100	82
P33	1531	78	17	4	250	209	2006	209	21	42	42	9	30	100	76
P34	3032	78	17	4	500	417	4046	417	38	84	84	389	427	100	75
P35	601	72	16	12	100	81	748	81	33	17	17	1	34	100	80
P36	1478	72	16	12	250	198	1889	198	49	40	40	8	57	100	78
P37	2979	72	16	12	500	400	3933	400	48	80	80	350	398	100	76
P38	450	81	17	2	100	56	522	56	13	12	12	1	14	100	86
P39	1106	81	17	2	250	139	1285	139	30	28	28	5	35	100	86
P40	2208	81	17	2	500	270	2639	270	63	54	54	26	89	100	84
P41	1151				588	118	1176	116	119	37	37	235	354	62	98

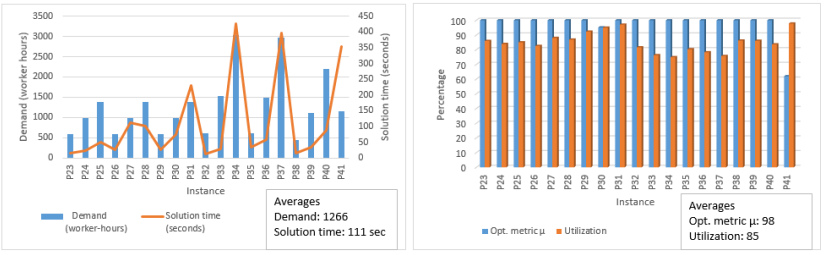


FIGURE 2.9: Performance metrics of two stage method for P23 to P41

worker-hours and it took 354 seconds to solve. Fig.2.9 presents the performance of two stage method.

We shall compare the performance of the two stage method with that of Volland et al. (2017b). For this, we use the results of instances P23 to P31. Since we do not have the data used in Volland et al. (2017b), we use the simulation approach. Recall that instances P23 to P31 are simulated following the procedure stated in Volland et al. (2017b) in toto. It must be pointed out that the comparison is not based on exact instances but

TABLE 2.5: Comparison with VF w.r. to time wise performance

Size	Demand	Correspondence		Time (seconds)		Reduction
		VF	TSM	t_{VF}	t_{TSM}	Percent
Small	595	SMA-S1-LW-FL	P23	240	15	93
	592	SMA-S2-LW-FL	P26	360	24	93
	588	SMA-S3-LW-FL	P29	900	51	94
Medium	990	MED-S1-LW-FL	P24	10800	26	99
	990	MED-S2-LW-FL	P27	600	111	81
	994	MED-S3-LW-FL	P30	3180	101	96
Large	1384	LAR-S1-LW-FL	P19	10800	26	99
	1393	LAR-S2-LW-FL	P28	300	74	75
	1388	LAR-S3-LW-FL	P31	2760	231	99

Note: t_{total} is the total time extracted Table 5 of Volland et al. (2017b); t_{TSM} is the total solution time by two stage method (TSM) taken from Table 4. Comparison is made based on similar but not the same instances.

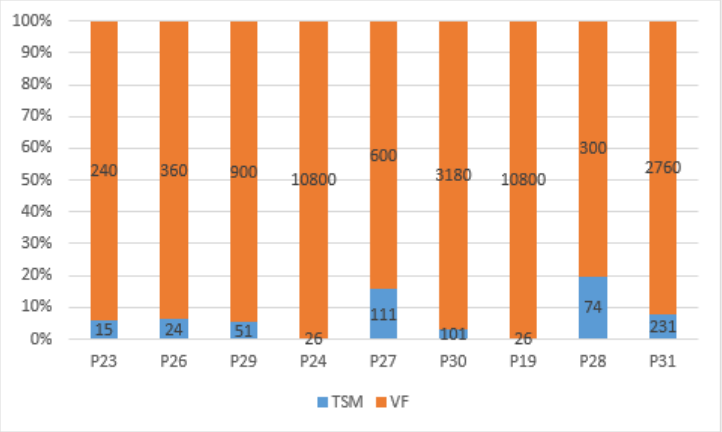


FIGURE 2.10: Comparison of solution times of two stage method (TSM) with Volland et al. (2017b) (VF) method.

on similar instances. Table 2.5 presents the one-to-one correspondence between the two sets of problem instances along with respective solution times. The solution times for Volland et al. (2017b) are taken from their article. Both methods produced optimal solutions for all the nine instances. The last column of the table presents the reduction percentages in the solution times. The solution times are also shown in Fig.2.10

2.6 Summary

In this chapter, we considered the integrated staff and task scheduling problem. The problem is hard to solve even for a predetermined task schedule. Several authors have considered the problem and proposed column generation methods to solve. In this chapter, we propose a two stage approach to the problem and introduce the split technique to handle problems with large demand. We have demonstrated the efficacy of the two stage method with split technique through a number of numerical experiments in reducing solution times dramatically. In the existing literature, solution methods are assessed at different demand sizes such as small, medium and large. Through the split technique introduced in this chapter, we are able to handle problems with large demands efficiently. This raises a question that whether demand size has any influence on the complexity of the problem. This point needs to be explored theoretically. Another direction for future research is extending the methods introduced in this chapter to multi-skill personnel staff scheduling problems.

Chapter 3

Planning Airport Check-in Counter Allocation

3.1 Introduction

Check-in counter allocation is an important problem, particularly at large airports. Airlines plan flight departures on a seasonal basis using weekly rosters, each season comprising about six months, and submit their plans to the airport operator for facilities. The airport operator explores the possibility of accommodating airlines' requests considering available resources at the airport and makes commitments to the airlines for the entire season. In order to make seasonal commitments, the airport operator studies optimal utilization of resources at macro level. One of the important resources is check-in counters. This chapter is concerned with finding solutions to the check-in counter allocation problem primarily at the planning stage. Modelling this problem to find implementable solutions is difficult and challenging. A number of authors have studied this problem using Operations Research (OR) and Simulation tools [Chun and Mak (1999), Atkins et al. (2003), Dijk and Sluis (2006), Bruno and Genovese (2010), Araujo and Repolho (2015), Trakoonsanti (2016), Bruno et al. (2018), etc].

At present, there are no exact formulations available for controlling waiting time theoretically and implementing queue disciplines. Further, the existing exact formulations

for check-in counter allocation maintaining adjacency constraints (Duin and Sluis (2006) and Dijk and Sluis (2006)) are not suitable for large scale real-world problems. The main contribution of this chapter is to provide effective solutions to these problems through exact formulations. We present real-life examples to demonstrate the application of these formulations.

The organization of this chapter is as follows. Section 3.2 introduces the CAP in general and the specific problem instance at the international airport for which a solution is sought. Section 3.3 presents the literature survey. Section 3.4 introduces the notation and the basic framework required to formulate the problem. Section 3.5 presents new formulations and approaches used in solving the problem. Numerical comparisons are also presented here. Section 3.5.1 deals with stage 1 optimization problem. New formulation is proposed to address some issues in the existing formulation. Section 3.5.2 deals with stage 2 problem and addresses scheduling large scale CAPs. Section 3.6 is devoted to solving real-life problems. The chapter is concluded in Section 3.8.

3.2 The Check-in Counter Allocation Problem

The counter allocation problem is important in planning resource usage in an airport. Different airlines (also referred to as carriers) operate their flights to various destinations from a given airport. Passengers of each flight undergo the mandatory *check-in process* in which they check in their baggage, if any, and obtain boarding passes from the *check-in counters*. The time period for which planning is required is called the planning horizon which may be a day, a week, a month or a season. For the sake of convenience, the planning horizon is discretized into N contiguous time-periods (TPs) of equal length, say, ω minutes each. The TPs are labelled 1 through N with 1 being the earliest and N being the latest. Each flight to be scheduled in the planning horizon has a unique identification number (ID) with the particulars of carrier, number of seats and scheduled time of departure (STD).

Each flight requires check-in counters for a certain period (call it the *check-in period*) for checking in the passengers of that flight. It is customary in the literature to identify

the check-in period for a flight with a set of contiguous TPs, from four hours before STD to 45 minutes or one hour before STD.

The number of arrivals in each TP, written in chronological order as a vector, is referred to as the **arrival pattern** for a flight. Also, a two dimensional space is used as a representation of the counters in the planning horizon (see Chun (1996) and Dijk and Sluis (2006)). The length of the TP has to be chosen for modelling. Too small TPs mean less time for staff to handle operations and large TPs can consist of large variations in passenger arrivals. If $(r_1, r_2, \ldots, r_k)$ is the arrival pattern of a flight with number of seats S, then $L = \frac{100 \sum_i r_i}{S}$ is called the **load factor**. Here, k is the number of time periods in the check-in period of the flight.

Check-in counters are owned by the airport operator and suitably leased to airlines. The arrangement of check-in counters varies from airport to airport. For the purpose of this study, it is assumed that the counters are relabelled such that counters j and $j+1$ are adjacent. The airport considered has 168 counters arranged in 12 rows within six islands. The airport operator assigns the counters to airlines based on requirements. If a counter is assigned to check in passengers of a single flight, it is called **dedicated counter**. If a counter is assigned to check in passengers of a group of flights of a carrier during their common check-in periods, it is called a **common counter**. All the 168 counters at the airport have the facility of being utilized as dedicated or common counters depending on the requirements. If the number of counters assigned to a flight (or a group of flights) remains same over all TPs of the check-in period, then it is known as **constant or fixed counter allocation**; otherwise it is known as **variable counter allocation**.

The CAP is concerned with the allocation of counters to departure(s) of any carrier within the planning horizon so that counters assigned to departures are adjacent. The airport schedules various airline departures on a weekly basis. The weekly schedule is repeated over a season (one season is from April to October and the other from November to March). At present, there are about 2500 departures of about 60 airlines in a week. The airlines are categorized into two classes, domestic (country based) and international. Under the current practice, each of the Indian carriers is assigned common counters.

Despite having many check-in counters, the airport is unable to meet airline demand. Therefore, this study was initiated to answer the following questions:

1. Whether the existing counters can accommodate all of the presently committed departures?
2. Does the current method of constant counter allocation provide sufficient counters to airlines?
3. Whether there is scope for scheduling more departures?

The airport operator provided necessary input data and restrictions for modelling the problem: (a) deterministic inputs for computing arrival patterns of passengers and service (check-in) times, (b) to control passenger waiting time as per (IATA-ADRM (2014)) norms, and (c) to consider a single class (i.e., no priority lines for business class). We, therefore, address the management objectives outlined above with formulations that can handle large size CAPs with deterministic inputs and control waiting times as desired.

3.3 Literature

A multitude of models exist in literature because of variations in the optimization criteria, modelling, airport requirements and airport layouts. We discuss relevant mathematical models, approximation algorithms, heuristic and exact approaches from literature. After reviewing literature, we find that most articles do not consider check-in area space restrictions, queue lengths and FIFO queue discipline. Initial attempts to solve the counter allocation problem were made by simulation of resource requirement and allocation at airports. Constraint satisfaction algorithms were presented by Chun (1996) and Chun and Mak (1999). Simulation was also used by Brunetta et al. (1999) and Snowdon et al. (2000).Wibowo and Fadilah (2018). Subsequently, simulation was used to study terminal planning by Wong and Liu (1998) and Kiran et al. (2000) and analyse check-in and delays by Appelt et al. (2007), Felix and Reis (2017) and Bevilacqua and Ciarapica (2010).

Mathematical Models for Counter Determination:

There are many mathematical models for predicting counter requirement: Park and Ahn (2003) aim to assign counters based on passenger arrival distribution at the airport. Cumulative arrival distribution (based on time before departure) is estimated by a regression model. The authors then allocate counters to airlines directly in proportion to the regression estimates. Due to this, the counter requirement for each departure is large when arrivals peak. Our model generates flatter counter assignments and avoids schedules with peak demands. This ensures lower personnel (and counter) changes and better utilization of counters. Waiting time of passengers is also limited, which indirectly limits queue length. Park and Ahn (2003) do not consider the number of counters in the airport, FIFO queue discipline and queue length.

OR formulations to the stage 1 problem with deterministic inputs were first proposed by Bruno and Genovese (2010). Models are presented for both static and dynamic counter allocation. Araujo and Repolho (2015) extend their model by adding a service level constraint. This formulation was further extended by Bruno et al. (2018) to schedule staff at check-in counters. The service level constraint imposed in these formulations does not limit waiting time directly. It ensures that a small percentage of passengers remain in queue at the end of a TP. The choice of service level and width of TPs effect waiting time. In accordance with IATA guidelines for check-in, airports need to ensure that airport space is optimally utilized and queue lengths are minimized. Due to this, models are expected to limit queue lengths, and simultaneously maximize the level of service. The formulations discussed above maximize the service level but fail to limit queue lengths and guarantee FIFO queue discipline.

Hsu et al. (2012) propose an ILP (based on the dynamic model by Nikolaev et al. (2007)) for dynamic allocation of different check-in facilities and dynamic assignment of passengers at airports to minimize the number of check-in facilities and passenger waiting time. Their model assigns the n^{th} passenger a check-in facility (counters, kiosks etc) based on service requirement and assignment of the $(n-1)^{th}$ passenger. Due to this dynamic assignment of passengers by the model takes more than 3 hours for 15 passenger arrivals. In view of this the authors use clustering algorithms. The main drawback is at airports where passengers have strong preferences and ignore assignments. This may

cause some check-in facilities operating at the airport to be insufficient.

Hwang et al. (2012) present a mathematical model for optimizing check-in counters, kiosks, part-time staff and full-time staff required during each shift in a week. Their model assumes static counter allocation. The authors compute the ratio of counters to kiosks that would be best suited for serving a flight. The authors conclude that usage of kiosks reduces operational costs.
Parlar and Sharafali (2008) propose a stochastic dynamic model to determine optimal counters for a single flight. Parlar et al. (2013) propose static counter allocation policy for a single flight. Their objective is to minimize the expected total cost. The models presented are complex and have long computational times.

Models for Adjacent Counter Allocation: For allocating adjacent counters different models have been used:
Simulation Approaches: Chun (1996) allocated adjacent counters by defining structures called counter profiles, two dimensional shapes that define check-in counter requirement and allocation for a flight. The shape of a counter profile is changed to fit in the counter-TP space. If the order of flight allocation does not have a feasible solution, the algorithm backtracks. This results in a time consuming process for counter allocation.

ILPs for Adjacency: In the static counter allocation problem considered by Yan et al. (2004) and variable counter allocation considered by Yan et al. (2005), each counter has one or two service lines to provide check-in service. To allocate adjacent counters to flights, the authors define a block as a set of service lines and then allocate flights to these blocks. Tang (2010) defines each block as a set of adjacent counters and uses a network model for the same. This means more blocks added as input may result in better flight assignments, but increase complexity of the problem. The objectives of Yan et al. (2004) are to allocate counters to minimize passenger walking distance and reduce inconsistency in counter location. For 490 flights in a week, 140 counters and 260 service lines, allocating five service lines to each flight results in 80000 variables and 70000 constraints. Due to the complexity involved in computing, the authors propose a heuristic algorithm. These models involve a lot of preprocessing to remove redundant constraints. Due to this, running time varies exponentially with input size.

Dijk and Sluis (2006) model counter allocation problem by combining simulation and integer programming. The problem is solved in two stages. In the first stage terminating simulations are run to determine counter requirement till the solutions satisfy service level requirements. In the second stage, an ILP is solved for adjacent allocation of counters. An increase in number of flights and planning horizon results in increased problem size.

Some authors have proposed genetic and evolutionary algorithms for the check-in counter allocation problem (Yeung and Chun (1995), Mota (2015), Mota and Zuniga (2013)). The efficiency of these techniques relies highly on parameters that drive the selection procedure (see Mota (2015) and Affenzeller et al. (2009)). GAs search from a set of points called a population and various biologically inspired operators like selection, crossover and mutation are applied to obtain better solutions (Bandyopadhyay and Pal (2007)). In Mota (2015), flights are allocated sequentially to the Gantt chart, taking into account all the constraints in flight allocation such as no overlap etc. In order to generate the population, the flight order is changed. Crossover operations are performed to improve the existing solutions. The resulting solutions with a high measure of fitness are then retained and again crossed over. The objective function is computed as a fitness measure for each generation and checked for improvement. In the algorithm proposed, the solutions are improved till a stop condition is reached. In the study by Yeung and Chun (1995), each individual represents a check-in counter allocation plan for one day. Fitness measure is the number of overlaps found in an allocation plan. The fittest individual is the best allocation plan for that day. These techniques are expected to provide good solutions, i.e. solutions that are close to optimal but may not be optimal (see Goldberg (1989) and Mota (2015)).

Problem Complexity: The complexity of the problem has been studied in detail by Duin and Sluis (2006). The counter allocation problem with the adjacency restriction is NP-complete (Duin and Sluis (2006), Dijk and Sluis (2006)) and cannot be solved in polynomial time. Duin and Sluis (2006) develop an ILP based on the models for Resource Constrained Project Scheduling Problem (RPSP) (see Pinedo and Chao (1998)) leveraging the similarity in the two problems. The techniques discussed in this paper for allocating adjacent counters can be applied to other adjacent resource allocation problems, such as: warehouse space optimization berthing problem at ship yards, (see

Bierwirth and Meisel (2010)), for assignment of computer hard disk memory (if contiguous allocation of memory is required (see Duin and Sluis (2006))) and other resource scheduling problems (at call centers and hospitals) where jobs cannot be shifted in time but resource requirements may be satisfied by any set of adjacent resources and may vary with time. For a detailed review of existing literature, see Appendix A.

3.4 Notation

The inputs to the check-in counter allocation problem are, the flight departure particulars, the planning horizon, arrival pattern of passengers and service particulars. The notation for these are described in this section.

Consider a planning horizon with N TPs, each of width ω minutes. Unless stated otherwise, the unit of time is minutes throughout this chapter. Let D be the the total number of departures of K carriers in the planning horizon. We use the index i for the TP number, j for departure ID, and k for counter number. Let M be the number of counters at the airport, and we assume that counters k and $k+1$ are adjacent, $1 \leq k < M$. For each departure j, let $i_o(j), i_o(j)+1, \ldots, i_c(j)$ be the check-in TPs. Two departures j and j' of a carrier are said to be overlapping if they have at least one common check-in TP. A subset of departures T of a carrier is called a task, provided for any $j,\ j' \in T$, there is a sequence $j_1, j_2, \ldots, j_g \in T$ such that $j = j_1$, $j_g = j'$, and j_l and j_{l+1} are overlapping for each $1 \leq l < g$. It is expected that assigning common counters to departures in a task may reduce counter requirement. For the purpose of this chapter, we redefine a **task** as a set of departures of a carrier that are allocated common counters. By definition, a single departure requiring dedicated counters is also a task comprising only that departure. Next, each departure belongs to one of the two categories - Indian (or domestic) and international.

The number of passengers that arrive in TP i for departure j will be denoted by d_{ji}. Define the measure **counter-window** (CW) as ω minutes of one counter. Note that there are NM CWs in the planning horizon. If f_{jl} is the number of counters assigned to departure j in the TP l, then the total number of CWs assigned to j is

$\sum_{l=i_o(j)}^{i_c(j)} f_{jl}$. Therefore, $\frac{100\sum_j \sum_l f_{jl}}{NM}$ is a performance measure, namely, the percentage of CWs allocated against the total number of CWs available.

Let x_{itk} be a binary variable equal to 1 if counter k serves passengers of task t in TP i, and equal to 0 otherwise. The x_{itk}s are the **decision variables** which must satisfy a number of constraints. Note that $\{kx_{itk} : i = 1, 2, \ldots, N,\ k = 1, 2, \ldots, M\}$ is the set of counters that serve passengers of task t, and the **adjacency constraint** stipulates that this set must be a set of consecutive integers.

Next, $c_i = \sum_{t,k} x_{itk}$ is the total number of counters assigned to all tasks in TP i, and this cannot exceed M. This leads to the set of **counter availability constraints**: $\sum_{t,k} x_{itk} \leq M$ for each TP i.

Clearly, number of counters assigned affects the number of passengers checked in as well as waiting times of the passengers in the queue. Therefore, x_{itk}s must be chosen to facilitate check-in of all passengers in the queue and to ensure that waiting times are controlled as desired.

Formulations with objective function involving cost aspects have been considered in the past. As far as this study is concerned, the main objective is to minimize the number of counters allocated or assigned. Our approach to achieve this objective is given below:

(a) For each task t, first determine the number of counters $c_i(t) = \sum_k x_{itk}$ in TP i to check-in all passengers of t so that the largest of $c_i(t)$s is as small as possible (that is, by minimizing $\max_i c_i(t)$). Let $w_0(t)$ denote this largest $c_i(t)$.

(b) Redetermine the number of counters, $c_i(t)$s for task t, by minimizing $\sum_i c_i(t)$ with the additional constraints $c_i(t) \leq w_0(t)$ for all i (see Example 3.5.1 for an illustration).

(c) After obtaining $c_i(t)$ for each t as above, assign the counters to tasks so that task t gets the required $c_i(t)$ adjacent counters in TP i for each i.

3.5 New Formulations

In stage 1 of CAP, the number of counters is determined for every task. Using the results of stage 1 optimization as inputs, stage 2 optimization determines the physical assignment of counters to tasks satisfying the adjacency constraints.

Further, it has been observed that while simulation is useful in analyzing the effectiveness of solutions obtained and improving them, ILP formulations are key to deriving optimal or near-optimal solutions. Irrespective of the solution methodology, the effectiveness of the solutions depends on the accuracy of arrival patterns and service time distributions provided as inputs in deriving the solutions. An important factor that influences prediction accuracy is the width of the TP. While larger widths are expected to have smaller prediction errors, they restrict the scope for optimization and result in wastage of counter time.

In this section, new ILP formulations are presented for problems arising in the two stages, and various formulations are compared. The formulation for stage 1 addresses controlling waiting time theoretically and ensures FIFO. The formulation for stage 2 reduces the complexity in the model by Dijk and Sluis (2006), by restricting the assignments to special structures (see Section 3.5.2) which are more pragmatic and amenable to useful modifications if necessary. More importantly, the formulation helps in solving large scale problems in reasonable time.

3.5.1 Stage 1: Determining Number of Counters

In this section, we shall deal with the problem of determining the number of counters required for a given task.

Allocation of Counters to a Task

Consider a task with $\hat{D}$ departures. Without loss of generality, assume that these departures are $1, 2, \ldots, \hat{D}$ with STDs in chronological order (ties, if any, are broken arbitrarily). Let P denote the time horizon for these departures. Then, $P = \{i_o(1), i_o(1) + 1, \ldots, i_c(\hat{D})\}$. For $j = 1, 2, \ldots, \hat{D}$, define $P_j = \{i_o(j), i_o(j) + 1, \ldots, i_c(j)\}$. Then, $d_{ji} = 0$ for all (j, i) such that $i \notin P_j$.

Following are the model assumptions:

1. The system has a single queue for serving passengers of all departures in a task. There may be multiple counters for check-in. The queue discipline followed is FIFO.
2. The service time per passenger is s_D for domestic departures and s_I for international departures; the arrivals d_{ji}s are estimated and given as inputs.
3. Allocation policy is to allow *variable* number of *common* check-in counters.

The uv-Formulation

The main challenge in formulating the CAP is controlling the waiting time for every passenger. The uv-formulation proposed below ensures that no passenger waits for more than a pre-specified limit on waiting time, say τ minutes.

For (j, i) such that $i \in P_j$, let u_{ji} of the d_{ji} passengers be checked-in in TP i and the remaining $v_{ji} = d_{ji} - u_{ji}$ in TP $i + 1$. If $i = i_c(j)$, set $v_{ji} = 0$, as passengers of flight j cannot be checked in TP $i_c(j) + 1$. Let s_j denote the check-in time for departure j, where $s_j = s_D$ if departure j is domestic, $s_j = s_I$ otherwise. For notational convenience, set $u_{ji} = v_{ji} = 0$ for all (j, i) such that $i \notin P_j$. Recall that c_i is the number of counters used in TP i for the task in question. Therefore, $\max_i\{c_i\}$ is the number of counters required to check-in all the passengers of the task, and the objective is to minimize $\max_i\{c_i\}$. With this, consider the uv-formulation with decision variables u_{ji}, v_{ji} and c_i given below.

The uv-Formulation:

`Minimize` z

`subject to`

$$u_{ji} + v_{ji} = d_{ji}, \forall\ i \in P_j,\ j = 1, 2, \ldots, \hat{D}, \tag{3.5.1}$$

$$v_{ji_c(j)} = 0\ \forall\ j = 1, 2, \ldots, \hat{D}, \tag{3.5.2}$$

$$\sum_j s_j(u_{ji} + v_{j(i-1)}) \leq \omega c_i\ \forall\ i \in P, \tag{3.5.3}$$

$$c_i \leq z\ \forall\ i \in P, \tag{3.5.4}$$

u_{ji}, v_{ji} `and` $c_i\ \forall\ i \in P$, `are nonnegative integers.`

Note that the above formulation minimizes the objective function $\max_i\{c_i\}$ (indirectly) using the usual trick of introducing the dummy variable z and the constraints (3.5.4). Constraints (3.5.1) and (3.5.2) ensure that all passengers will be checked in the allowed TPs; $\sum_j(u_{ji} + v_{j(i-1)})$ is the number of passengers checked in TP i and LHS of constraint (3.5.3) is the time required to check in these passengers; constraint (3.5.3) ensures that adequate number of counters (c_i) are provided. Once this problem is solved and the optimum z, say w_0, is found, we solve the problem again by changing the objective function to $\sum_i c_i$ (the total number of CWs) and changing RHS of constraint (3.5.4) to w_0.

Remark 3.5.1. *From the above formulation, it is clear that the maximum waiting time for any passenger is $\tau = 2\omega$, where ω is the width of TP. Further, in the solution, the implementation of the queue discipline, FIFO, is built-in, as in TP i, $\sum_j v_{j(i-1)}$ passengers are checked in before $\sum_j u_{ji}$ passengers, in the order of their arrivals.*

Remark 3.5.2. *The expression on the left-hand side of constraint (3.5.3) is the total time required for checking in $\sum_j(u_{ji} + v_{j(i-1)})$ passengers in TP i. Theoretically, the constraint is an* approximate one *and may fail to yield a feasible solution sometimes. However, empirical experience has shown that this works fairly well. This constraint can be replaced with an exact constraint when all the departures in the task are of the same type, either all are domestic or all are international. To see this, consider the case where all departures are domestic. Then, the constraint $\sum_j(u_{ji} + v_{j(i-1)}) \leq \nu c_i$, where ν is the integral part of $\frac{\omega}{s_D}$ will meet the exact requirement.*

Remark 3.5.3. *Determination of the number of counters required in each TP has a difficulty from the point of implementation. Consider a solution to the problem in which $c_i = 4$, $c_{i+1} = 5$, $c_{i+2} = 4$. In this case, an additional counter is required to operate in TP $i + 1$ for only ω minutes. This could be a difficult proposition to implement in practice, particularly when ω is small. Therefore, a pragmatic approach to this problem is to impose a constraint such as: once a counter is opened, it must remain open during a specified number of TPs. This helps carriers as they are not interrupted too frequently. The uv-formulation can be amended easily to incorporate this new constraint. Suppose we wish to impose a constraint that once a counter is assigned, it should be made available for 4 consecutive TPs. To impose this, we can modify the formulation as follows. Redefine c_i as the number of TPs to be* opened *at the beginning of TP i, and change the*

constraint (3.5.3) to

$$\sum_j s_j(u_{ji} + v_{j(i-1)}) \leq \omega(c_i + c_{i-1} + c_{i-2} + c_{i-3}) \ \forall \ i \in P, \tag{3.5.5}$$

where $c_h = 0$ *for* $h \notin P$. *Note that* $c_i + c_{i-1} + c_{i-2} + c_{i-3}$ *is the number of counters available for serving passengers in TP* i.

Comparison of Stage 1 Formulations

We now compare the solutions of various formulations for stage 1 problem. One of these is the formulation (3.2) of Araujo and Repolho (2015) which we shall refer to as AR-formulation henceforth. We pointed out earlier that AR-formulation does not follow FIFO and that it controls waiting time implicitly. This is illustrated below.

Example 3.5.1. *Consider the task comprising three departures with common counters and check-in periods spread over 18 TPs. For details see Table 3.1. The UV-1 row provides the solution (the* c_i *values) of the uv-formulation. This gives the minimum* z *as 8 (=* w_0*). The problem is again solved by replacing the objective function with* $\sum c_i$ *and constraint (3.5.4) by* $c_i \leq 8 \ \forall \ i$. *Solution (values of* c_i*s) to this problem is shown in UV-2 row of the table. The solution of AR-formulation (as* q_{ji}*s and* c_i*s in the AR-row, where* q_{ji} *is the number of passengers of flight* j *to be checked in TP* i *as defined in the formulation) is also provided in the table.*

Note that 3 (= d_{17}) passengers who arrived in TP 7 are checked in TP 9 (included in $q_{19} = 16$), but 18 (= d_{28}) passengers of departure 2 who arrived in TP 8 are checked in TP 8 ($q_{28} = 18$). Thus, FIFO is violated in the solution obtained from AR-formulation.

To examine the waiting time aspect, let $\theta_i(g)$ be the number of passengers that arrive in TP i and are checked in TP $i + g$, $g = 0, 1, 2, ..$, and let $N_0 = \sum_{j,i} d_{ji}$ be the total number of arrivals of all flights. Then, $\theta(g) = \sum_i \theta_i(g)/N_0$, is the proportion of passengers that arrive in TP i and are checked in TP $i + g$ for some i. Further, $\eta(k) = \sum_{g \geq k+1} \theta(g)$ is the proportion of passengers who wait at least $k\omega$ minutes, $k = 1, 2, \ldots$.

For uv-formulation, $\eta(k) = 0$ for all $k \geq 2$, that is, no passenger has to wait more than 2ω minutes. For Example 3.5.1, the AR-formulation yields $\theta(1) = 30/675$ and

TABLE 3.1: Problem of three departures with common counters

TP	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
d_{1i}	0	0	0	0	0	0	3	5	13	27	40	46	54	62	68	40	27	27
q_{1i}	0	0	0	0	0	0	0	0	16	32	40	41	58	60	60	45	30	30
u_{1i}	0	0	0	0	0	0	0	5	13	27	4	10	8	14	12	4	24	27
d_{2i}	1	2	4	8	12	14	16	18	20	12	8	8	0	0	0	0	0	0
q_{2i}	1	2	4	8	12	14	16	18	20	12	6	10	0	0	0	0	0	0
u_{2i}	0	2	4	8	0	0	3	6	14	4	4	8	0	0	0	0	0	0
d_{3i}	0	1	2	5	9	14	15	18	21	23	14	9	9	0	0	0	0	0
q_{3i}	0	1	2	5	9	14	14	19	21	23	14	9	9	0	0	0	0	0
u_{3i}	0	1	0	5	0	1	0	18	21	23	14	2	9	0	0	0	0	0
AR	1	1	1	2	3	4	4	5	8	9	8	8	9	8	8	6	4	4
UV-1	1	1	2	3	4	4	5	6	8	8	8	8	8	8	8	8	8	8
UV-2	0	1	1	2	0	3	4	8	8	8	4	8	8	8	8	8	8	4

Note: d_{ji}s, are the arrivals; $\omega = 15$, service time is 2 minutes per passenger; q_{ji}s are the solutions from AR-formulation; u_{ji}s are the solution according to uv-formulation ($v_{ji} = d_{ji} - u_{ji}$); AR row provides optimal number of counters required as per the solution of AR-formulation; UV-1 is the solution of uv-formulation with maximum number of counters as the objective function, and UV-2 is the solution of uv-formulation with the objective function as the total number of CWs and with the additional constraint that the maximum number of counters does not exceed maximum of UV-1 row. The q_{ji}s are the decision variables in AR-formulation, stand for number of passengers of flight j checked in TP i.

$\theta(2) = 9/675$. This is because there is no constraint in AR-formulation that limits the waiting time directly. To examine the effects of various solutions we use real data of Problem 1 given below. For Problem 1, the solution by AR-formulation results in $\theta(g) > 0$ for $g \leq 4$. The θ-values are presented in Table 3.2. Problem 1 is another instance which indicates that AR-formulation does not control waiting time directly.

TABLE 3.2: Waiting time metric for SOL2 and SOL3

Solution	$\theta(0)$	$\theta(1)$	$\theta(2)$	$\theta(3)$	$\theta(4)$
SOL2	13892/17151	3259/17151	0	0	0
SOL3	15779/17151	1178/17151	168/17151	19/17151	7/17151

Problem 1. *Consider the problem of determining number of counters for a task with 67 domestic and 46 international departures of one carrier on common counter basis and with load factor of 85%. For this task,* $N = 111$, $D = 113$, $\omega = 15$. *This is a subproblem of a real-life problem (Section 3.6.1).*

Problem 1 is solved using four different methods and the solutions are denoted by SOL1 to SOL4 as follows.

SOL1. SOL1 is the solution obtained using uv-formulation. The optimal objective value of this problem is $w_0 = 57$. This solution uses 3739 CWs.

SOL2. Solve the modified version of uv-formulation with objective replaced by $\sum_i c_i$, the total number of CWs, with the additional constraint $c_i \leq 57$ for all i. This solution is denoted by SOL2. SOL2 uses 2882 CWs and a maximum of 57 counters.

SOL3. This is the solution obtained using AR-formulation. According to SOL2, $\theta(0) = 0.87$. Therefore, for a meaningful comparison between SOL2 and SOL3, we took $\alpha = 0.87$, a parameter that is used in AR-formulation.

SOL4. This is the current allocation at the airport, and it uses 42 counters in each TP (fixed counter allocation).

For simulation, we first determined the number of arrivals of each TP using seat capacity and the arrival patterns supplied by the airport operator obtained through a survey (see Fig. 5 for the arrival patterns). Once the number of arrivals d_{ji} in TP i is determined, the d_{ji} arrival times in that TP are simulated using uniform distribution. Although SOL3 violates FIFO, we used FIFO to check in passengers for all the four solutions using respective counter allocations in our simulation program. Some performance metrics of the solutions are listed in Table 3.3. To compare the solutions, we repeated the simulation exercise 100 times (i.e., 100 simulation runs) so that 100 observations are available on each of the metrics. The averages of the performance metrics are presented in Table 3.3.

From the results presented in the table, the following inferences may be drawn (our primary interest is in the comparison between SOL2 and SOL3).

SOL2 Vs. SOL3: AR-formulation (SOL3) requires more counters ($\max_i c_i$) compared to modified uv-formulation (SOL2); in terms of counter-hours, the performance of SOL2 and SOL3 is approximately the same; with respect to waiting time parameters, while SOL3 is slightly better, both perform well. The number of counters required over different TPs for SOL2 and SOL3 is plotted in Fig. 3.1.

Others Vs. SOL2 and SOL3: SOL4 is best among all with respect to the main objective of minimizing the number of counters. However, with respect to other

TABLE 3.3: Summary of simulation results for comparison

Metric \ Solution	SOL1	SOL2	SOL3	SOL4
Max Counters ($= \max_i c_i$)	57	57	69	42
Counter Hours (hrs)	935	720	721	1015
Average WT (min)	1.53	2.42	1.04	10.37
Std.Dev. of WT (min)	3.21	3.22	0.75	15.68
Third Quartile of WT (min)	2.66	2.68	1.48	15.70
95-percentile point of WT (min)	10.13	10.51	2.34	49.95
Maximum WT (min)	12.73	15.83	9.77	57.19
NPMF	35.85	70.57	28.22	741.31
PPWT0	0.35	0.89	0.90	0.54
PPWT20	0.00	0.00	0.00	0.17

Note: WT = waiting time, PPWT0 = proportion of passengers with waiting time equal to zero minutes, PPWT20 = proportion of passengers with waiting time greater than 20 minutes, NPMF = number of passengers missing flights, counter-hour = 4 CWs (as $\omega = 15$).

parameters it is poor. SOL1 performs well with respect to waiting time parameters, but it uses a large number of counter-hours.

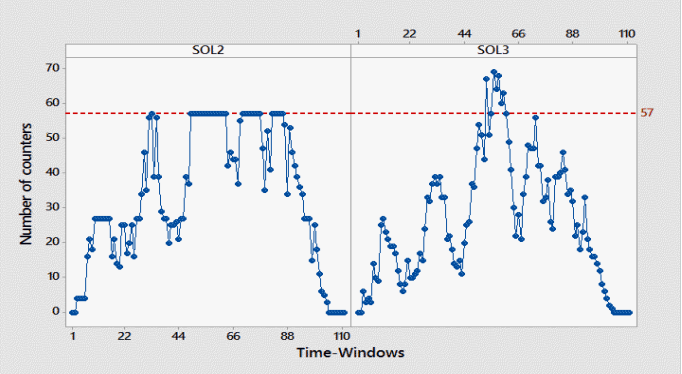


FIGURE 3.1: Number of counters as per SOL3 and SOL2

3.5.2 Stage 2: Scheduling Tasks with Adjacency Constraint

To assign adjacent counters for every task, task structures are defined as follows:

Assignment Structures

Stage 1 output $c_i(t)$s are the inputs for stage 2 problem. Given $c_i(t)$s, the number of counters required for a task t in TP i, the physical assignment of counters to tasks can be made in several ways satisfying the adjacency constraint. Consider a task that has check-in period spread over 16 TPs and the required number of counters given by $c(t) = (2, 3, 5, 6, 6, 6, 6, 6, 6, 5, 4, 4, 2, 1, 1, 1)$. Four different ways of assigning counters 1 to 6 are exhibited in Fig. 3.2. Each of them satisfies the adjacency constraint. However, it is natural that one would prefer either (a) or (d) to implement, as these involve least disturbance. We refer to structures of this type as **top-down Structures** and structures of the type (d) of Fig. 3.2 as **bottom-up Structure**.

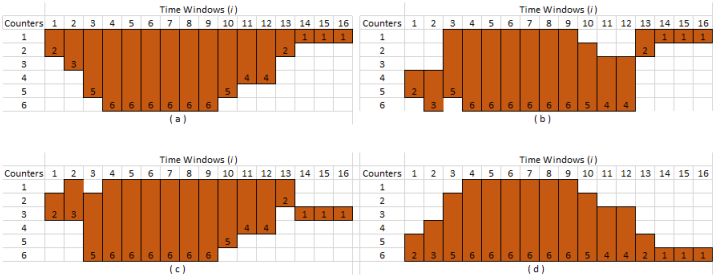


FIGURE 3.2: Four different ways of assigning counters to a task.

Task Array

Consider a task t with check-in TPs $i_0, i_0+1, \ldots, i_1$. Let $c_i(t)$ be the number of counters required for task t (obtained in stage 1), $i = i_0, i_0 + 1, \ldots, i_1$. Define the **Task Array** Q_t with $(k, i)^{th}$ element $q^t_{ki} = 1$ for $k = 1, 2, \ldots, c_i(t)$; $i = i_0, i_0 + 1, \ldots, i_1$. In this manner, the top-down structure of a task is uniquely identified with a task array. For $i_0 = 1, i_1 = 16$ and $c(t) = (2, 3, 5, 6, 6, 6, 6, 6, 6, 5, 4, 4, 2, 1, 1, 1)$, the task array is given by:

$$Q_t = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & & & \\ & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & & & & \\ & & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & & & & \\ & & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & & & & & & \\ & & & 1 & 1 & 1 & 1 & 1 & 1 & & & & & & & \end{bmatrix}$$

Assignment of Counters for All Tasks

Consider the problem of assigning adjacent counters to T tasks. Label the tasks as $1, 2, \ldots, T$. Let $i_o(t)$ and $i_c(t)$ denote the earliest and latest TPs of task t respectively. Let $c_i(t)$ be the number of counters required for task t in TP i, $i = i_o(t), i_o(t)+1, \ldots, i_c(t)$. Let Q_t be the task array of task t.

For each task-counter pair (t, k), define the decision variable y_{tk} as: $y_{tk} = 1$ if task t starts at counter k, and $y_{tk} = 0$ otherwise. This results in assignment of counters $k, k+1, \ldots, k+c_i(t)-1$ to task t in TP i, $i = i_o(t), i_o(t)+1, \ldots, i_c(t)$. Fig. 3.3 illustrates assigning three tasks to counters in two different ways. The first assignment in panel (a) corresponds to $y_{11} = 1$, $y_{2,10} = 1$ and $y_{3,16} = 1$, and the second assignment in panel (b) corresponds to $y_{11} = 1$, $y_{26} = 1$ and $y_{3,10} = 1$. From this it is evident that the three tasks can be managed with 16 counters (as in panel (b)). Actually, it is

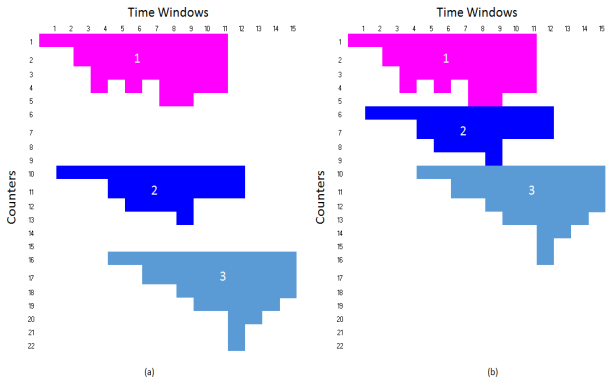


FIGURE 3.3: Two different ways of assigning counters to 3 tasks

.

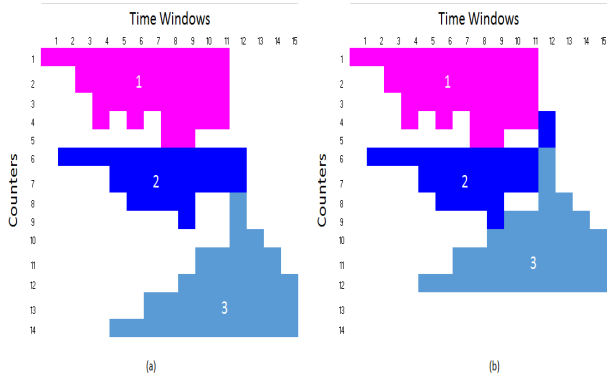


FIGURE 3.4: Two more ways of assigning counters to the 3 tasks

.

possible to reduce the number of counters further as shown in Fig. 3.4. Note that (a) in

Fig. 3.4 is an acceptable structure with 14 counters but uses a combination of top-down and bottom-up structures, where as (b) is not a pragmatic structure since the airline has to put up with more interruptions, but uses only 12 counters. In view of the practical implementation and computational complexity, we shall restrict counter assignments to only top-down structures. We shall now present our formulation for stage 2 problem.

Fix a counter-TP pair $(k,\ i)$ and consider the variables w_{ki} and m_t defined by

$$w_{ki} = \sum_{t\in\mathcal{T}_i}\sum_{h=1}^{c_i(t)\wedge k} y_{t(k-h+1)}, \tag{3.5.6}$$

$$m_t = \max\{c_i(t) : i_o(t) \le i \le i_c(t)\}, \tag{3.5.7}$$

where $\mathcal{T}_i$ is the set of all tasks t such that $i_o(t) \le i \le i_c(t)$. Note that w_{ki} is the number of tasks that use counter-TP combination $(k,\ i)$, and $\sum_k ky_{tk} + m_t - 1$ is the largest counter number used by task t. The YTK-formulation given below minimizes the number of counters in stage 2 optimization problem by minimizing the upper bound s (the objective function) on the largest counter number used for assignment.

The YTK-Formulation:

`Minimize` s

`subject to`

$$\sum_{t\in\mathcal{T}_i}\sum_{h=1}^{c_i(t)\wedge k} y_{t(k-h+1)} \le 1 \ \texttt{for all}\ (i,k), \tag{3.5.8}$$

$$\sum_k y_{tk} = 1 \ \texttt{for all}\ t, \tag{3.5.9}$$

$$\sum_k ky_{tk} + m_t - 1 \le s, \ \texttt{for all}\ t, \tag{3.5.10}$$

$$y_{tk} \in \{0,\ 1\} \ \texttt{for all}\ t,\ k, \tag{3.5.11}$$

As in the case of uv-formulation, the true objective function in this formulation is $\max_t \sum_k ky_{tk} + m_t - 1$ which is handled through the introduction of the dummy variable s and the constraints (3.5.10). The above formulation helps achieve this objective in a linear fashion. Constraint (3.5.8) ensures that no counter is assigned to more than

one task in any TP, counters assigned to tasks are contiguous in all TPs and that the assignment of counters to tasks is made using the top-down structure; constraint (3.5.9) ensures that all tasks are accommodated.

Remark 3.5.4. *Note that the above problem is always feasible. If the optimal objective value is more than M, it means that the existing resources are not adequate. In such a case, the formulation can be modified to accommodate maximum number of tasks by maximizing $\sum_{t,k} y_{tk}$ subject to constraints (3.5.8), (3.5.11), and the new constraints $\sum_k y_{tk} \leq 1$ and $\sum_k k y_{tk} + m_t - 1 \leq M$ for all t.*

Remark 3.5.5. *Note that $\sum_t c_i(t)$ is the minimum number of counters required in TP i. Therefore, $\max_i \sum_t c_i(t)$ is a lower bound on the objective function s in the YTK-formulation.*

A Comparison with the Formulation of Dijk and Sluis

We shall refer to the formulation given in (4.3) of Dijk and Sluis (2006) as DS-formulation. This formulation does not allow the type of undesirable task structures such as the one shown in panel (b) of Fig. 3.4. The last two constraints of the formulation ensure maximum overlap among the counters assigned to a task in any two adjacent TPs. This aspect is also built in our model through the structure of task arrays. However, DS-formulation is more general in the sense that any feasible assignment in YTK-formulation is a feasible assignment in DS-formulation but the converse is not true. In other words, the set of feasible solutions of YTK-formulation is a proper subset of feasible solutions of DS-formuation. This actually turns out to be a disadvantage in solving problems, particularly the large ones, because of the complexity induced by the constraints in DS-formulation. The formulation by Dijk and Sluis (2006), assuming 2500 flights in a week, 687 TPs and 168 counters, results in $17,17,500$ variables and more than $62,500$ constraints (constraints for each overlapping TP have been excluded from this calculation). In comparison, our model has $4,20,000$ variables and $1,20,416$ constraints.

To compare the two formulations, we considered two real-life instances I1 and I2. I1 has 82 tasks comprising 360 departures over 111 TPs (see Problem 2 in Section 3.6.1). I2 has 182 tasks comprising 782 departures over 229 TPs. The two instances are solved using both formulations (see Section 3.6.1 for more details). The results are summarized in Table 3.4. The size of the problem (number of variables plus constraints) is significantly

large in DS-formulation. From the results, we see a substantial computational benefit of YTK-formulation over DS-formulation. In fact, using YTK-formulation clubbed with judicious strategies, we have been able to solve a large problem of planning a week's schedule with 2539 departures over 687 TPs (see Section 3.6.2).

TABLE 3.4: A Comparison Between Results of YTK- and DS-Formulations

Problem Instance	YTK-Formulation	DS-Formulation
I1: 82 tasks comprising 360 departures over 111 TPs. The minimum counter requirement for this problem is 148 counters.	This has size 38766 (= 16401 variables plus 22365 constraints). Solver produced first feasible solution in 1 minute with objective value of 168 counters. In 26 minutes the best objective was 159, and no further improvement even after 2 hours. On adding the lower bound constraint $s >= 148$, the best feasible solution produced in 2.5 hours had objective value $s = 157$ counters.	This has size 51002 (=16241 variables plus 34761 constraints). The first feasible solution appeared in 10s with objective value 465 counters. In 1 minute objective value reached 293, and remained there even after 15 hours at which time the solver was interrupted.
I2:182 tasks comprising 782 departures over 229 TPs. The minimum number of counters required is 171.	This has size 82566 (=36401 variables plus 46165 constraints). The first feasible solution appeared in 2 minutes with an objective value of 189 counters. At the end of 1 hour objective value remained 189. With the introduction of lower bound constraint $s >= 171$, a feasible solution with objective value 189 was found in one hour.	This has size 191062 (=61695 variables plus 129367 constraints). The first feasible solution appeared in 30s of running with objective value of 227 counters. In 40s objective value reached 226, and remained there even after 16 hours. Upon introducing the lower bound constraint $s \geq 171$, the solver found a feasible solution with objective value 214 in one hour.

3.6 Solving Real-World Problems

We consider two problems. In the first problem, the airport operator makes commitments to the airlines for two seasons; April to October and November to march, as the departures are scheduled using weekly rosters, our first problem aims at solving check-in counter allocation problem over one week planning horizon. To explore execution module, we took up problems with one day planning horizon.

For these problems, we have considered the following inputs: (i) one arrival pattern for all domestic flights and one for all international flights (as per airport operator's survey), (ii) load factor is 85%, (iii) fixed service times ($s_D = 2,\ s_I = 3$), (iv) maximum waiting time per passenger is 30 minutes, (v) width of the TP is 15 minutes and (vi)

queue discipline is FIFO. Data is analysed using international standards for service time in addition to deterministic inputs provided by the airport operator (based on a survey).

The arrival pattern for 15-minute TPs is shown in Fig. 3.5 (percentage values). The passenger arrival pattern for a departure is calculated using the number of seats of that flight departure and the load factor. Professional solver LINGO 13.0 was used on a quad core computer with a processor speed of 3.4 GHz and 16 GB RAM to perform all computations.

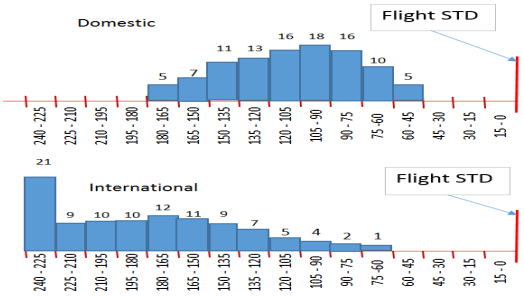


FIGURE 3.5: Distributions of arrivals percentages over TPs for domestic and international departures.

3.6.1 The One-Day Problem

This instance has 360 ($= D$) departures of 46 ($= K$) carriers. Of these, 287 departures belong to the five Indian carriers. The departures of carriers 1, 2 and 5 are taken as one task each, departures of carrier 3 are grouped into 4 tasks, and departures of carrier 4 are grouped into two tasks. The remaining departures of non-Indian carriers form 73 tasks of which 68 are single-departure tasks. Thus, this one-day instance has 82 tasks in all.

Problem 2. *This problem is to determine the counter assignment for the one-day instance explained above, taking* $\omega = 15$ *minutes.*

Since $\omega = 15$, there are 96 TPs in a day. As the check-in period starts 4 hours before a departure, we add 15 TPs prior to the first TP of the day. Thus, $N = 111(= 96+15)$. For each task $t \in \{1, 2, \ldots, 82\}$, we determine the minimum number of counters $c_i(t)$ required in the i^{th} TP using the uv-formulation (as in SOL2). This resulted in $\max_i \sum_t c_i(t) = 148$ which is the lower bound for the one-day problem.

The next step is to physically assign counters to all the 82 tasks ensuring adjacency. Using YTK-formulation a feasible solution with objective value 157 was found in 2.5 hours (see Table 3.4 for more details). This solution is at least 94% optimal as the lower bound for the objective function of this problem is 148 ($= \max_i \sum_t c_i(t)$, see Remark 3.5.5). The final counter allocation with 360 departures, with one day planning horizon is provided as a part of section 3.7).

3.6.2 One-Week Problem

For this problem, $\omega = 15$ minutes, $N = 687$ $(= 96 \times 7 + 15)$, $D = 2539$ and $M = 168$. Of the 2539 departures, 2007 belong to the five domestic carriers and require common counters as per the existing policy. Each of these carriers has counter requirement in almost all TPs, consequently corresponding departures are grouped into the first five tasks. The remaining 532 departures formed 497 tasks with 462 single-departure tasks.

The lower bound, $\max_i c_i(t)$, for this problem is 171. In terms of formulation and solution methodology, the one-week problem and the one-day problem are similar. The major challenge in the one-week problem is in handling its size.

Strategy to tackle the Large Problem

The large tasks corresponding to domestic carriers caused infeasibility. To overcome this problem, we first solved the problem with five tasks.

The solver produced a global optimum solution in 24 minutes. This solution used 167 counters and made the following assignment: $y_{1,1} = y_{2,78} = y_{3,130} = y_{4,152} = y_{5,160} = 1$. We then solved the entire one-week problem using YTK-formulation by adding this allocation as a constraint. The solver produced a global optimal solution to this problem in 42 minutes with optimum objective value 179 counters. Comparing with the lower bound 171, this solution is at least 95% $(= (1 - \frac{179-171}{171}) \times 100)$ optimal for the one-week problem. Since the airport has only 168 counters, the objective function was changed to $M \leq 168$ (see Remark 3.5.4) to fit as many tasks as possible. The solver produced a global optimal solution to this problem in 14 minutes with the optimal objective value of 523 tasks. We then prepared the assignment layout using an excel macro, and using this we could accommodate 10 of the 14 leftover tasks (with a mild tampering

of few task structures). In the final solution, the percentage of assigned CWs is 68, leaving 32% of counter time free for assigning maintenance work, scheduling new flights or for accommodating cancellations and unforeseen changes to the flight schedule. We developed a physical layout in MS Excel to create a visual display of the solution (see section 3.7). The resultant pictures are a powerful tool for the airport operator to understand at a cursory glance the counter allocation status on any day of the week. This allows the management to plan maintenance activities, make adjustments in the allocation, incorporate new tasks and improvise the existing allocation (the final solution with 2539 departures is provided as part of section 3.7).

3.7 Additional methods for Check-in Counter Planning

The excel solution layout of the check-in counter allocation problem and a heuristic suggested for solving large instances of the problem are presented below. We present the assignment layout and consider the one day and one week problems presented in chapter 3.

Assignment Layout

The problem of assigning tasks/departures to counters can be presented using an *Assignment Layout* as shown in Fig.3.6.

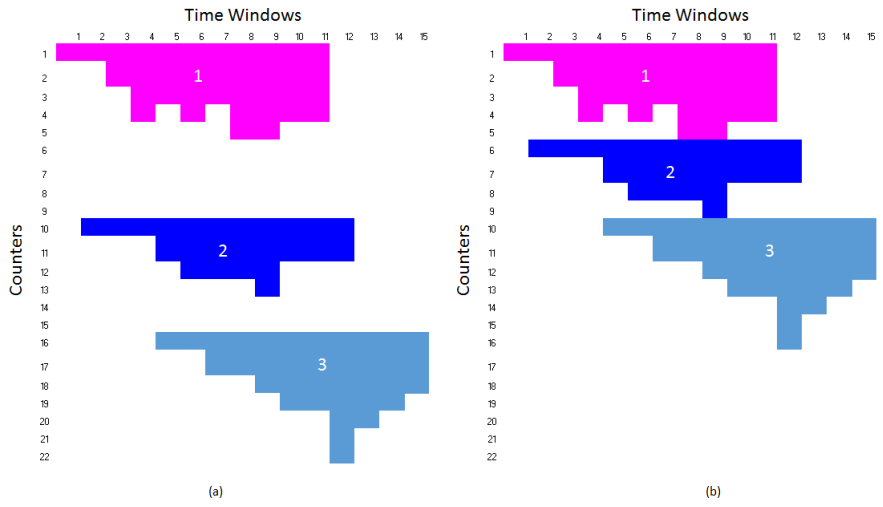


FIGURE 3.6: Two different ways of assigning counters to 3 tasks

.

In the assignment layout x-axis denotes the time windows and the y-axis denotes the physical counter numbers. It is assumed that counters numbered k and $k+1$ are adjacent for all k. The layout is an array whose $(k,\ i)^{th}$ cell represents the k^{th} counter during i^{th} time window. Tasks and departures appear as their structures and are colored. The uncolored (white) cells in the layout are free counters in the respective time windows. The white cells in the array remain vacant and may be used for accommodating new departures (provided there is sufficient space), maybe allocated to airlines for further easing congestion at the airport or may be used for maintenance activities. The layout can be presented in an excel spread sheet, and with the help of excel viewing tools for magnification and compression, one can have detailed views of large layouts such as the ones that represent even weekly planning.

One-Day Problem

For the one-day planning horizon problem discussed in chapter 3, the total number of tasks is 82. Of these, 14 are with two or more departures, and the remaining 68 are with one departure each. Minimum number of counters is determined for the 82 tasks using the uv-formulation and minimizing the total number of counters as done for SOL2. The next step is to make the physical assignment to all the 82 tasks ensuring adjacency and using minimum number of counters.

The one-day planning horizon problem is solved using the second stage optimization formulation with the inputs from the first stage solution SOL2. The solution required 163 counters. To understand the solution, consider Fig.3.7. This figure shows a portion of the assignment array which covers counters 92 to 123 and time windows between 32 and 53. Departure IDs are shown in the structures. It can be seen from this figure that departure with ID 46 is assigned counters 104 to 106 during the time windows 35 to 42. Note that counter 106 is not assigned to the departure during the time windows 35, 36, 38 and 42 (white spaces). Further, no two structures overlap. The complete assignment of 360 departures is shown in Fig.3.8. For this solution, the ratio of assigned counter-windows to total number of counter-windows is 49.6% ($= \frac{9254}{168\times111} \times 100$).

One-Week Problem

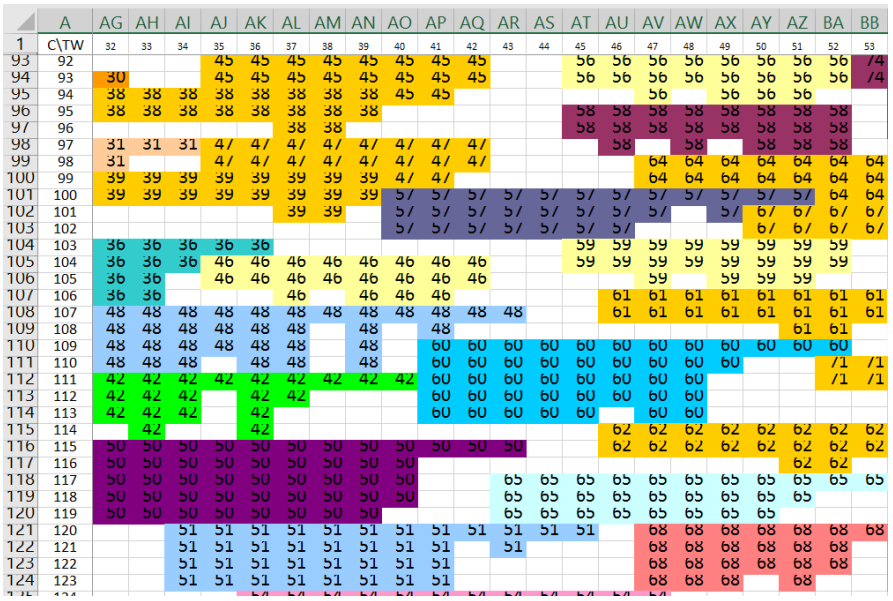


FIGURE 3.7: A portion of assignment of counters to one-day planning horizon problem.

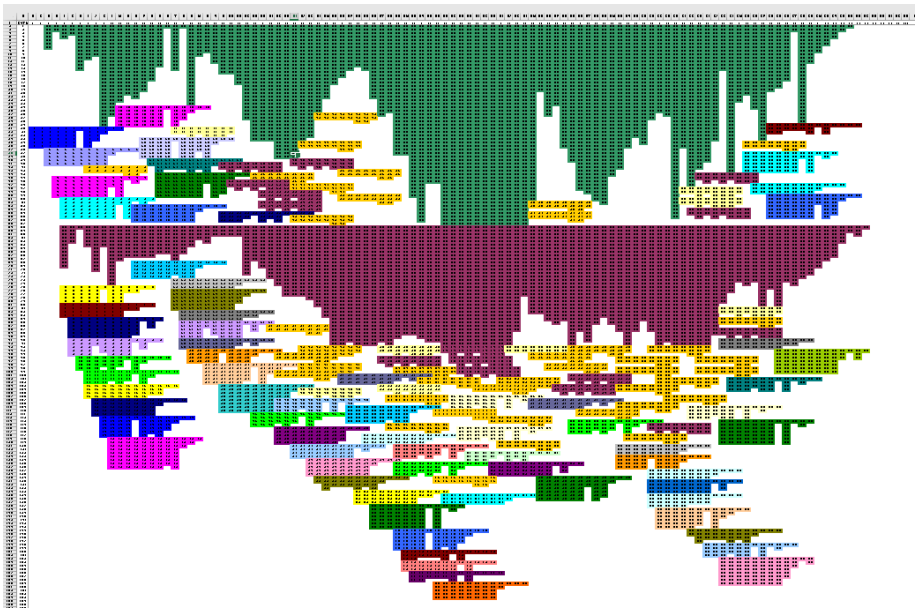


FIGURE 3.8: Physical assignment of counters to the 360 departures of the one-day planning horizon problem.

In terms of formulation and solution methodology, the one-week problem and one-day planning problems are same. The only issue with one week problem is that it is too large compared to one-day problem. The one-week problem addressed to us has a total of 2539 departures. For solving this, the same time window ($\omega = 15$ minutes) length is used. With this the total number of time windows is 687 ($= 96 \times 7 + 15$). The number of counters in the first stage optimization were determined for each departure

(for simplicity). Using these first stage outputs, the second stage optimization problem was solved. When the program was run on a quad core computer with a processor speed of 3.4 GHz and 16 GB RAM, the solver took nearly two hours and stopped with a message that memory was insufficient. To overcome this hurdle, we propose a rolling horizon heuristic. The main purpose of this heuristic is to divide the entire planning horizon of the original problem into subproblems with smaller problem size (lesser departures) and much lower computation times. For the one week problem, the planning horizon was divided into three subproblems and each subproblem was solved separately. In each iteration of the heuristic, time horizon may be fixed or the number of vessels to be included may be fixed. Since, in this case, the flight departures were evenly distributed, we chose to fix the planning horizon for each subproblem, this also resulted in an even distribution of the flight departures in each subproblem. In each iteration all the variables ($y_{tk}s$) obtained in the previous subproblems were fixed. The solution to the original problem was obtained by pooling the solutions of the three subproblems (described as a heuristic below). The assignment layout of the problem is shown in Fig.3.9. Upon minimizing the number of counters utilized, the solution could accommodate 2535 out of the 2539 departures. Thereafter, using the assignment layout it was possible to accommodate the left over 4 departures as well by slight tampering of the task structures. In the optimal solution, the percentage of assigned counter-windows is 67.9% ($= \frac{78385}{687 \times 168} \times 100$).

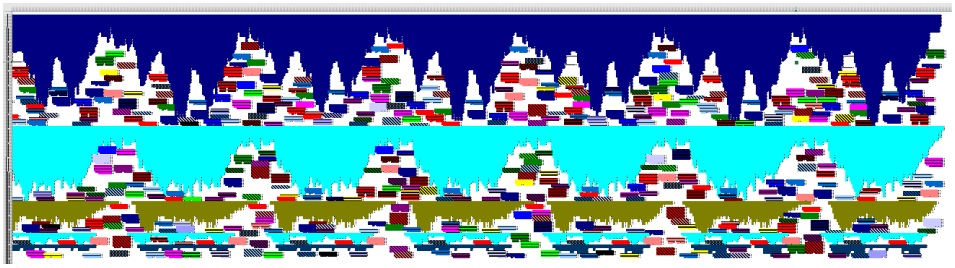


FIGURE 3.9: Physical assignment of counters to the one-week planning horizon problem involving 2539 departures and 687 time windows.

Algorithm 1: Rolling Horizon Heuristic for large problems

Result: Rolling Horizon Heuristic

1. t: Number of time windows to consider in each subproblem.
2. d_i: Number of flight departures before $i * t$ considered in subproblem i.
3. nc $<-$ Number of counters available for assignment in the planning horizon for the i^{th} subproblem.
4. it $\leq -$ time windows 1 to it are considered in the i^{th} iteration.
5. Itn $<-$ Number of iterations to cover the planning horizon.
6. Solve the problem for the first t time windows and d_1 departures.
7. **for** ($i = 2;\ i < Itn;\ i = i + 1$) {
   8. fix variables y_{tk} for departures in d_i requiring counters in time windows (1 to (i-1)t).
   9. solve the i^{th} subproblem with d_{i-1} y_{tk}s from subproblem (i-1) as input.

}

Comparison of ILP models for adjacent resource scheduling of Dijk and Sluis (2006) and of Lalita et al. (2020)

In this section, we compare the ILP formulation for adjacent resource scheduling proposed by Dijk and Sluis (2006) and the Y_{tk} formulation presented in 3.5.2, under the assumption that all tasks have rectangular structure (i.e. the ARS-R variant of adjacent resource scheduling problem). In the adjacent resource scheduling problems, we have as input a set of tasks and also renewable resources. Each task t has to be assigned a constant set of adjacent resources between the task starting time and ending time. For convenience to the reader, we present both the formulations below.

Formulation (DS) proposed by Dijk and Sluis (2006)

$$\text{minimize } D \tag{3.7.1}$$

$$\text{subject to} \tag{3.7.2}$$

$$n_f <= d_f \text{ and } \qquad d_f <= D\ \forall f \tag{3.7.3}$$

$$d_f + n_g <= d_g \text{ or } \qquad d_g + n_f <= d_f \tag{3.7.4}$$

$$\forall f, g \in 1, 2, .., T \qquad \text{with } I_f \cap I_g \neq \phi, \tag{3.7.5}$$

$$d_f \in \mathbb{Z}^+, f \neq g \tag{3.7.6}$$

In this formulation, n_f is the number of counters required for check-in of passengers

of flight f in the check-in time interval I_f, d_f is the largest counter number assigned to flight f and D is the total number of counters assigned to all the flight f. In this formulation, d_f and D are the decision variables.

Formulation (Y_{tk}) proposed by Lalita et al. (2020)

$$\text{minimize } s \tag{3.7.7}$$

$$\text{subject to} \tag{3.7.8}$$

$$\sum_{t\in\tau_i}\sum_{h=1}^{m_t\wedge k} y_{t(k-h+1)} <= 1 \quad \forall(i,k) \tag{3.7.9}$$

$$\sum_{k} y_{tk} = 1 \quad \forall t \tag{3.7.10}$$

$$\sum_{k} k.y_{tk} + m_t - 1 <= s \quad \forall t, \tag{3.7.11}$$

$$y_{tk} \in \{0,1\}, \forall\ t = 1,2,..,T \text{ and } k = 1,2..,K \tag{3.7.12}$$

In this formulation, m_t is the total number of counters required for check-in of passengers of flight t. τ_i is the set of all tasks t requiring a counter in time period i, k is the counter number for possible allocation to a flight t, s is the total number of counters required for assigning all the flights to counters and y_{tk} equals 1 if task t is assigned to counter k. In this formulation, s and y_{tk} are decision variables.

Theorem 3.7.1. *The ILP models, (DS) of Dijk and Sluis (2006) and (Ytk) of Lalita et al. (2020) are equivalent for the ARS-R case. In particular, given an optimal solution to the ILP (DS), we can construct a feasible solution to the ILP (Ytk) with the same objective value, and vice-versa.*

Proof. Suppose $(\bar{y}_{tk}, \bar{s})$ (with indexed components) is an optimal solution to the (Y_{tk}) formulation. Below, we construct a feasible solution $(\hat{d}_f, \hat{D})$ to the (DS) formulation.

- Since $\hat{d}_f$ is the last counter occupied by task f, it is given by $\sum_k k.\bar{y}_{fk} + m_f - 1$. Therefore, $\hat{d}_f = \sum_k k.\bar{y}_{fk} + m_f - 1$. Also, $n_t = m_t$, $\forall t$ as both denote the number of counters required for task t.
- Since $\hat{D}$ is the number of counters for scheduling all the tasks, $\hat{D} = \bar{s}$.

- From the above equivalent definitions, we see that $n_f \leq \hat{d}_f \leq \hat{D}$. Hence, constraint (3.7.3) of the (DS) formulation is satisfied.

- Let t and $t\prime \in \tau_i$. Then, from 3.7.9,
$\sum_{h=1}^{n_t \wedge k} \bar{y}_{t(k-h+1)} + \sum_{h=1}^{n_{t\prime} \wedge k} \bar{y}_{t\prime}(k-h+1) \leq 1 \ \forall \ (i,k)$.

- Let $\bar{y}_{tk} = 1$, i.e. let the task t start at counter k. Then, $\bar{y}_{t\prime j} = 0$, for $j = (k - m_t\prime + 1, k - m_t\prime + 2, .., k) \cup (k+1, .., k + m_t - 1)$. This implies that $\sum_j j.\bar{y}_{t\prime j} \leq k - m_t\prime$ or $\sum_j j.\bar{y}_{t\prime j} \geq k + m_t$. Using the equivalent definition of $\cap d_t$ in terms of y_{tk}, we get, $\hat{d}_t = k + m_t - 1$, i.e., $\hat{d}_t + n_t\prime <= \hat{d}_t\prime$ *or* $\hat{d}_t\prime + n_t <= \hat{d}_t\prime$.

- Hence, constraint 3.7.4 is satisfied and we get a feasible solution to the (DS) formulation.

Suppose $(\bar{d}_t, \bar{D})$ ($\bar{d}$ with indexed components) is an optimal solution to the (DS) formulation. Below, we construct a feasible solution $(\hat{y}_{tk}, \hat{s})$ to the (Y_{tk}) formulation.

- Let t and $t\prime$ be two tasks such that $I_t \cap I_t\prime \neq \phi$, i.e. the tasks or flights have counter requirements in overlapping time intervals. Let task t start at counter k.

- Then, $\bar{d}_t = \sum_k k.\hat{y}_{tk} + m_t - 1 = k + n_t - 1$, as n_t is the counter requirement for task t. This implies, $n_t \leq \sum_k k. \cap y_{tk} + n_t - 1 \leq \cap s$. Therefore, constraint (3.7.11) is satisfied.

- From constraint (3.7.4), substituting, $\cap d_t = \sum_k \hat{y}_{tk} + m_t - 1$, we get, $\sum_k k. \cap y_{t\prime k} \geq k + m_t$ or $\sum_k k. \cap y_{t\prime k} \leq k - m_t\prime$. Therefore, $\sum_{h=1}^{m_t \wedge k} y_{t(k-h+1)} + \sum_{h=1}^{m_t\prime \wedge k} y_{t\prime(k-h+1)} \leq 1 \forall t$. This implies constraint (3.7.3) is satisfied.

- $\sum_k \cap y_{tk} = 1$, as d_t has one value and any task t ends at counter d_t. This satisfies constraint (3.7.10)

With the above discussion, we prove that the two formulations have the same optimal objective function values. This suggests that in solving the two models, the difference in solution times is primarily due to the large number of variables and constraints in the (DS) formulation. For (DS) formulation, given T tasks, the number of variables $= T(T-1) + T + 1$ (as variables are added because of the 'or' constraints (3.7.10))

Formulation	Variables	Constraints
(DS) Formulation	31,24,000	62,50,000
(Y_{tk}) Formulation	4,25,000	1,20,000

TABLE 3.5: This table provides a comparison of the formulations in terms of number of variables and constraints in the worst case scenario for 2500 departures

and the number of constraints $= T(T-1)$ (in the worst case scenario). For the (Y_{tk} formulation, given T tasks and C counters, the number of variables $= TC$ and the number of constraints $= NC + 2T$ (where N is the number of time periods considered). There is a general agreement among scientists today that an algorithm is a practically useful solution to a computational problem only if its complexity grows polynomially with respect to the size of the input (see Papadimitriou and Steiglitz (1998)). This is because often we are interested in the the behaviour of algorithms with large inputs sizes, which is characteristic of real-world problems. In studying the complexity of the two models described above, the number of constraints of the (Y_{tk} formulation for the ARS-R case grows in proportion to the number of departures (T, whereas for the (DS) formulation, the number of variables and constraints is $O(T^2)$. A drastic reduction in both the number of variables and constraints in the Y_{tk} formulation result in smaller computation times. □

Given T departures, N time periods and C counters, for (DS) formulation the number of variables are O(T^2+T) and the number of constraints are O(T^2). Our formulation has O(TC) variables and O($NC+2T$) constraints. The Y_{tk} model has much lower variables and constraints.

3.8 Summary

In this chapter, we address the check-in counter allocation problem with deterministic inputs and variable counter allocation. Models from available research are suitable for small scale problems and do not consider FIFO and waiting time explicitly. We address these issues and provide exact solutions to large size real-world problems. As seasonal commitments by the airport operator are based on weekly schedules, it is essential to

determine check-in counter allocation for flight departures over one week. In this context our models become important as we provide a solution to the problem in the planning stage, where the airport operator commits to an airline schedule for a season. Scheduling for one day also involves a large number of flight departures at big airports. To test how well the proposed models can be applied to real-world problems, we studied two problems at an international airport in India. The one-week planning problem involved 2539 departures in 687 time periods. We also provide a solution for the execution module for one-day with 360 departures in 111 time periods. The solution for the one week problem resulted in a reduction of passenger waiting time and indirectly reduced the queue length. Also, the model solutions increased utility of the counters and resulted in 30% free counter time as compared to the current static counter allocation at the airport.

Chapter 4

Berth Assignment and Crane Scheduling at Ports

4.1 Introduction

Scheduling problems are very frequently encountered in a variety of industries and are crucial for business management. Berth and crane scheduling at cargo terminals of ship yards is one problem that has been attracting the attention of many researchers over the past three decades. This is evinced by numerous articles in various journals on this subject. Due to the complexities involved in the two subproblems, the berth allocation problem (BAP) and the quay crane assignment problem (QCAP), were studied separately by several researchers. As this approach can lead to inconsistencies in the combined solutions of the two subproblems, the research focus has drifted to solving the integrated problem, the berth allocation and quay crane assignment (specific) problem BACAP or BACASP. The operations involved in the problems just mentioned are referred to as seaside operations. The objectives and constraints encountered in planning the seaside operations vary largely due to the characteristics of the problems engendered by port layouts and infrastructure, business agreements with ship liners, *landside* operations, etc. A large number of models and solution methods have been proposed to address variants of seaside operational problems. The problems have been classified using selected attributes of the problems (Bierwirth and Meisel (2010)). Similarly,

the solution methods to address these problems are also classified into exact methods, heuristic based approaches, genetic algorithms and so on.

This chapter is concerned with a specific problem classification and the solutions using mathematical models to address it. One of the attributes for classifying BAP is the berth positions on the quay. If berth positions are predefined and fixed, then the problem is known as discrete, otherwise it is called continuous (see Bierwirth and Meisel (2010)). Similarly, in the crane assignment and scheduling problems, if a set of cranes assigned to a ship are not allowed to serve other ships while serving the assigned ship, the problem is known as time-invariant (see Imai et al. (2008)). Most of the problems addressed in the literature consider continuous BAP and *time-variant* QCAP or QCASP. These problems are more challenging and the exact methods to address them (using integer linear programming models) have issues in solving large scale (large number of ships over long planning horizon) instances. One way to address this issue is to add additional restrictions to the class of problems and evaluate the quality of solutions. Imposing these additional restrictions not only helps solve large scale problems, but also yield solutions that are often preferred from the implementation point of view. For example, in check-in counter allocation problem at airports, imposing additional restrictions on the structure of service facility has dramatically decreased the size of the problem that can be handled efficiently (see Lalita et al. (2020)).

In this chapter, we introduce a new class of solutions and evaluate their performance with the conservative solutions available in the current literature. More specifically, we consider the BACASP studied in Türkoğulları et al. (2014) and Agra and Oliveira (2018), and propose an alternative solution approach and compare the results with the existing ones. The main contribution of this chapter is in the new compact formulations and their ability to solve large size instances. We use an MILP model that is similar to the one used in Imai et al. (2008). The major difference is that their model is meant for discrete berths and due to its complexity, the model is not suitable for deriving exact solutions using commercial solvers. Our model is simple and is suitable for finding exact optimal solutions within the class of solutions defined by us. These solutions are simple to implement which is important from an operational point of view.

The organisation of this chapter is as follows. Section 4.2 presents literature review

and Section 4.3 presents a brief introduction to cargo ports and operations. Section 4.4 describes the specific variant of BACASP relevant to this chapter. In Section 4.5, we present our new class of solutions and discuss variants of the same. Section 4.6 presents the results of empirical experiments. Section 4.7 concludes the chapter with summary and remarks.

4.2 Literature

There is a vast literature on the berth allocation problem, quay crane assignment and scheduling problems and the integrated problem. Our interest is mainly in one variant of the integration problem. Therefore, we shall briefly mention the articles on the first two problems and focus more on the articles on the integrated problem relevant to our work. Firstly, there are two survey articles on this subject, Bierwirth and Meisel (2010) and Bierwirth and Meisel (2015), which classify the seaside operation problems by various attributes. Research on BAP started in the early 1990s (see Lai and Shih (1992); Brown et al. (1994)). Lim (1998) showed that the problem is NP-hard. Two different versions of BAP were considered - the operational BAP (OBAP) and the tactical BAP (TBAP). While the former is concerned with the problem of assigning and scheduling ships to berthing positions along the quay, usually over shorter planning horizons such as a week, the latter is concerned with other aspects, besides assigning ships to berth positions and scheduling their service times, such as quality of service and managerial decisions. See Imai et al. (1997), Imai et al. (2003), Imai et al. (2007b), Imai et al. (2007a) ,Imai et al. (2001), Imai et al. (2005), Guan and Cheung (2004), Lim (1998), Nishimura et al. (2001), Kim and Moon (2003), Cordeau et al. (2005), Monaco and Sammarra (2007), Mauri et al. (2016), Ursavas and Zhu (2016), Zhen (2015), Zhen et al. (2011), Wang and Lim (2007) for work on OBAP; and see Moorthy and Teo (2007), Cordeau et al. (2007), and Giallombardo et al. (2010) for work related to TBAC. With regard to crane operations, the QCAP deals with decisions on the number of cranes assigned to each ship along with time specifications. The QCASP problem, on the other hand, must determine the complete details of the tasks to be performed by the cranes assigned to each ship. Most of the papers dealing with QCASP specify only which cranes are operating on which ships and during what periods. This problem is known to be NP-hard in the strong

sense when there are three or more cranes with no preemption allowed or with different processing rates (see Pinedo (2002)). Unlike the job scheduling problems on machines, the QCASP has in addition the non-crossing constraints. Giallombardo et al. (2010) introduced the concept of *QC profiles* for solving the integrated problem. A QC profile specifies the number of cranes (assumed to be identical implicitly) to be operating during each time. This was found to be useful (tactically) and was used in models developed subsequently, see, for instance, Wang et al. (2018). See Al-Dhaheri and Diabat (2015), Daganzo (1989), Diabat and Theodorou (2014), Theodorou and Diabat (2015), Guan et al. (2013), Kim and Park (2004), Lim et al. (2004), Liu et al. (2006), Moccia et al. (2006), for work on QCASP.

In the integrated problem, BACAP and BACASP are two different problems. While BACAP specifies only the number of cranes assigned to ships, BACASP specifies which cranes are assigned to the ships in each time period. The integrated problem has attracted many researchers. See, for instance, Kim and Moon (2003), Guan and Cheung (2004), Ak (2008), Meisel and Bierwirth (2009), Imai et al. (2008), Meisel and Bierwirth (2013), Iris et al. (2015), Iris et al. (2017), Agra and Oliveira (2018) and Xie et al. (2019). For more references and brief details on the contributions, see Agra and Oliveira (2018) and Xie et al. (2019). Two other recent articles that consider yard management in the integration are Wang et al. (2018) and Liu et al. (2019), and the latter tabulates various contributions over the past two decades. Much of the work is centered around the integer programming formulations. Majority of these formulations are mixed integer linear programming (MILP) involving large number of binary variables and constraints making them complex and unsuitable for solving with commercial solvers. To overcome these problems, methods such as set partitioning formulations, introduction of valid inequalities, column reduction and column generation techniques are deployed. Besides exact methods, a number of heuristic and meta heuristics such as genetic algorithms have been proposed for the integrated problem.

Correcher et al. (2019) propose an iterative procedure for the BACASP in which the BACAP model is solved, and whenever its solution is not feasible for the BACASP, specific constraints are added until an optimal solution for the BACASP is found. Correcher and Alvarez-Valdes (2017) address the variant of both BACAP and BACASP consisting

of a continuous quay, with dynamic arrivals and time-invariant crane-to-vessel assignments. The authors propose a metaheuristic approach based on a Biased Random-key Genetic Algorithm with memetic characteristics and several Local Search procedures.

Our problem is closely related to Agra and Oliveira (2018). Agra and Oliveira (2018) consider BACASP with continuous version of BAP. Starting with an MILP based on relative position formulation, they propose a new formulation by discretizing the time and space variables to obtain exact solutions. They further enhance their model by adding valid inequalities and upper bounds for better performance. To obtain qualitative upper bounds, a rolling horizon heuristic is proposed. Unlike in many other models, Agra and Oliveira (2018) work with cranes which are heterogenous with respect to their processing rates.

4.3 Port Operations

Majority of the international transport of cargo is carried by ships. Container ships carry goods packed in containers whereas bulk material is carried by bulk ships. In this chapter, we are concerned with cargo ports equipped with cranes to handle the loading and unloading operations from ships. In ports where the ships are moored to the port for these operations, the port is equipped with quay cranes mounted on rails along the quay. Depending upon the length of the quay and other features such as draft, multiple ships can be moored to the quay for parallel operations (see Figure 4.1). Quay cranes

FIGURE 4.1: Quay at a cargo port

are mounted on rails along the quay (see Figure 4.2) and because of this their movement is restricted to the extent that they cannot bypass their adjacent ones. The cranes have capacities, the speed at which they can perform the loading and unloading operations.

For bulk material, the capacity is commonly specified as tons per hour, whereas in the case of cranes handling containers, the capacity is measured as number of containers per hour. As regards the containers, they come in different sizes. Their sizes are usually referred in terms of TEU (twenty-foot equivalent unit). For operational convenience, the quay may be divided into several berths where each berth position is fixed. In such a scenario, the number of ships handled parallelly is restricted to maximum number of berths. This is known as **discrete berth spacing**. On the other hand, depending upon the ship lengths, it may be possible to handle variable number of ships parallelly. In this scenario, the berths refer to the actual places occupied by the ships, and this is known as **continuous berth spacing**. For modelling purposes, quay is divided into berth sections of unit length. For instance unit length may be 25 meters. Taking this as yardstick, ship lengths are specified in terms of number of berth sections.

FIGURE 4.2: Quay cranes.

On the yard side, the terminals have huge storage space for storing containers in transit. The storage space is divided into zones, blocks within zones and subblocks within blocks where the containers are stored. The **yard cranes** are used to load and unload containers from trucks. The trucks carry the containers between yard locations and the berths. The yard is used as temporary storage space for the transit containers. Containers which are to be exported are first brought to the yard and from there they are loaded onto ships. The containers unloaded from ships are shifted to yard locations and from there they are moved to their respective destinations. Some of the containers unloaded from a ship are to be loaded into other ships depending upon the requirements. These containers are also first moved to yard locations and from there they are loaded into respective ships.

4.4 Problem Description

In this section we shall describe the integrated BACASP. The acronym was introduced in Türkoğulları et al. (2014). In the problem set up, ships arrive at a cargo port for service (transhipment operations of unloading and loading bulk material or containers). The service of ships is performed by quay cranes (QCs) at the quay. The QCs, numbered as 1 to K, possibly with different processing rates (the heterogenous case), are mounted on rails along the quay. QCs cannot bypass their adjacent ones on the rails. For the purpose of modelling, the quay is divided into berth sections of unit length which are numbered 1 to B. The service of ships is planned over a period of time, the planning horizon, which is discretized into T time periods numbered 1 to T. Each ship comes with a specification of its length (specified as the number of berth sections including a safety margin), workload (measured as number of containers in the case of container ships and as weight in tons in case of bulk material) and the arrival time at the port. The BACASP is to assign, for each ship, a contiguous set of berth sections (equal in number to the length of the ship) and a contiguous set of time periods along with the specification of crane numbers that serve the ship during each time period. The assignment must satisfy the following requirements: (i) crane assignments must satisfy non-crossing constraints, (ii) service must start in arrival time period or later, (iii) the total capacity of the assigned cranes must be adequate to complete the workload and (iv) no berth section is shared by two or more ships at the same time. This problem is classified under continuous berth allocation with time-invariant crane assignment. In the time-invariant models, any crane assigned to a ship cannot serve other ships until the service of the assigned ship is completed. Time-invariant models were introduced by Imai et al. (2008). The description of BACASP can also be found in Türkoğulları et al. (2014) and Agra and Oliveira (2018). One can consider different objectives for the problem. In this chapter, we shall take it as the sum of completion times of all ships.

The assignment can be presented geometrically in the *time-space* diagram. This is done in Figure 4.3 which presents a feasible assignment. In this assignment, each ship occupies a rectangular block in the time-space diagram where the length (along berth section axis) of the block is equal to the length of the ship and the width (along the time axis) of the block is equal to service period of the ship. A feasible assignment must

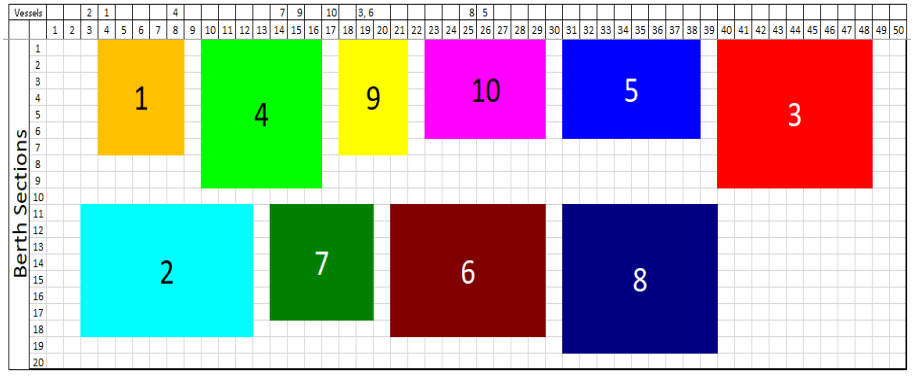


FIGURE 4.3: A solution for a BACASP instance with 10 ships. Data: $\tau = (7, 8, 9, 9, 6, 8, 7, 9, 7, 6)$, $q = 100(46, 77, 86, 68, 77, 75, 50, 74, 35, 70)$ (in tons), $a = (4, 3, 19, 8, 26, 19, 14, 25, 15, 17)$. From the figure, it means the ship 1 start time is TP 4 and end time is TP 8; for ship 2 start time is TP 3 and end time is TP 12 and so on (starting time period is the TP corresponding to the respective color and service end time is the last TP corresponding to the respective color). The optimal solution was obtained in 48 seconds but took 97 minutes to confirm optimality. Formulation has 464 variables and 1360 constraints.

satisfy the non-overlapping of the ship blocks. The service time (width of the block) depends upon the crane schedule assigned to the ship during its service time. The structure of the crane schedule is referred to as the quay crane (QC) profile, a concept introduced by Giallombardo et al. (2010)(see also Wang et al. (2018) for a description of QC profiles). It specifies the number of cranes assigned to a ship in each time period of the ship's service time. The QC profile is useful for the case of homogenous cranes. For the case of heterogeneous cranes, it is not sufficient to specify the number of cranes, but we must also specify which particular cranes are assigned. Later in this chapter, we shall introduce the concept of *berthing profiles* which will take care of this aspect.

4.5 Formulations

Following models in Ak (2008), Guan and Cheung (2004), Meisel and Bierwirth (2009), an MILP formulation for BACASP is presented in Agra and Oliveira (2018) using relative position formulation. Their model considers continuous berth allocation with time-variant assignment of cranes. In this chapter we consider a restricted model. In this model, berths are formed on a continuous basis with appropriate selection of cranes for each berth to suit the requirements of the problem instance. We consider the resulting solutions as a new class. In other words, consider all feasible solutions to BACASP

in which berth positions and the cranes assigned to them remain invariant throughout the planning horizon. We shall call this class of solutions as the Berth and Crane Invariant (BCI) solutions. The idea is to find the best of BCI solutions as a solution for the original BACASP. Clearly, BCI solutions are suboptimal. It may be noted that the problem of finding a BCI solution is different from the discrete BACASP. Also, our concept of invariant crane assignment is different from the one considered in Türkoğulları et al. (2014). The motivation for this approach comes from the fact that our formulation for finding a BCI solution is relatively simpler and suitable for commercial solvers compared to the existing MILP formulations for BACASP. More importantly, the general solutions obtained using formulations for optimal solutions are sometimes undesirable from practical point of view (see Example 4.5.4). We also propose a method to expand BCI solutions to a larger class by relaxing the invariance restrictions partially. We now present a formulation for finding a BCI solution. Refer to Table 4.1 for notation.

4.5.1 Berthing Profiles

To find a BCI solution, we first introduce the concept of berthing profiles. One of the constraints in BACASP is the specification on the number of cranes that must serve any ship at any given time. Let κ_m and κ_M be the limits on this number(see Table 4.1). Since cranes are ordered on the rails, we can partition them into contiguous sets and assign each such set to a contiguous set of berth sections (called a *berth*). Then, assign cranes to each berth thus formed. We shall illustrate this with some examples.

Example 4.5.1. *This example is used in Agra and Oliveira (2018), originally taken from Oliveira et al. (2016). In this problem, there are 7 cranes and the quay is discretized into 34 berth sections, each of length 25 meters. The crane data are summarized in Table 4.3 (see Table 4.1 for notation).*

Assuming $\kappa_L = 2$ and $\kappa_U = 4$, there can be either 2 berths or 3 berths. The capacity of a berth is determined as the sum of the processing rates of the cranes in that berth. For the above example, if the cranes are distributed into 3 berths, then there are three possible distributions.

TABLE 4.1: Notation

Symbol	Description
t	t is index in the planning horizon $\mathcal{T} = \{1, \ldots, T\}$,
B	number of berth sections in the quay; the berth sections are numbered from 1 to B,
u, v, V	u, v are ship indices, V = number of ships,
$N,\ \mathcal{B},\ \eta,$ $\beta_\eta,\ \pi_{\eta b}$	N is the number of berthing profiles indexed by $\eta \in \mathcal{B} = \{1, 2, \ldots, N\}$ (see Subsection 4.5.1 for the definition of berthing profile); β_η is the number of berths in berthing profile η, $\pi_{\eta b}$ is the number of berth sections in berth b of profile η,
b	b is the berth index; the range of b depends on the berthing profile; in berthing profile η, $b = 1, 2, \ldots, \beta_\eta$,
c	c is the capacity matrix, $c_{\eta b}$ is the capacity of berth b in berthing profile η,
k, K	K = number of QCs, $\mathcal{K} = \{1, 2, \ldots, K\}$ is the crane index set, crane k is between crane $k-1$ and crane $k+1$ on the rails, $k = 2, \ldots, K-1$,
$L_k,\ U_k$	crane range, crane k can serve in berth sections $L_k,\ L_k + 1, \ldots,\ U_k$ only,
p_k	processing rate of crane k,
$\kappa_m,\ \kappa_M$	number of cranes assigned to any ship must be at least κ_m and at most κ_M,
a_v	arrival time of ship v,
q_v	workload of ship v; for bulk it is by weight and for container shipping, it is by number of containers,
$\tau_v,\ \tau_M$	τ_v is the length of ship v in number of berth sections, and τ_M is the maximum ship length,
$y_{\eta bv}$	berth assignment indicator, = 1 if ship v is assigned to berth b of berthing profile η,
$s_{\eta bv}$	starting time period for loading ship v in berth b of berthing profile η,
$x_{\eta bv}$	length of service time for ship v in berth b of berthing profile η.

Berthing-Profile. We define a berthing-profile as a pair of vectors $(c,\ \pi)$, where c is the vector of berth capacities and π is the vector of berth lengths.

TABLE 4.3: Crane infrastructure at a bulk terminal

Crane Index	1	2	3	4	5	6	7
L_k	1	1	1	1	14	14	14
U_k	26	26	26	26	34	34	34
p_k (tons)	263.6	263.6	263.6	263.6	319.0	263.6	263.6

A berthing profile is the result of grouping contiguous berth sections and assigning each group a selected set of contiguous cranes. As an example, consider the data in Table 4.3. Suppose, we take berth sections 1 to 13 as one berth and 14 to 26 as another berth; and assign cranes 1,2 and 3 to the first berth (berth sections 1 to 13) and the rest to the second berth (berth sections 14 to 26). This decision results in two berths with capacities 790.8 and 1109.8 with corresponding berth lengths as 13 and 13 respectively. Thus, for this berthing profile $(c_1,\ \pi_1)$, $c_1 = (790.8,\ 1109.8)$ and $\pi_1 = (13,\ 13)$. Consider another berthing profile $(c_2,\ \pi_2)$ in which berth sections 1 to 7 are in one berth and 14 to 23 are in the other berth. Assign cranes 1 to 4 to the first berth (berth sections 1 to 7) and the rest to the other berth (berth sections 14 to 23). Then, $c_2 = (1054.4,\ 846.2)$ and $\pi_2 = (7,\ 10)$.

Potential Berthing Profiles

We shall call a berthing profile as potential berthing profile if the minimum length of the corresponding berths is greater than or equal to maximum ship length. In the above examples, $(c_1,\ \pi_1)$ is potential where as $(c_2,\ \pi_2)$ is not, provided maximum ship length is 9. Say that two potential berthing profiles are similar if their capacity vectors are equal after arranging their elements in the descending order.

Assuming $\kappa_L = 2,\ \kappa_U = 4$ and $\tau_M = 9$, there are 5 berthing profiles for Example 4.5.1 to consider. All of them are potential. They are listed in Table 4.4. Note that $(c_4,\ \pi_4)$ and $(c_5,\ \pi_5)$ are essentially same. Therefore, Example 4.5.1 has four berthing profiles $(c_\eta,\ \pi_\eta),\ \eta = 1, 2, \ldots, N$, where $N = 4$. For a berthing profile η, the berth capacities are $c_\eta = (c_{\eta 1},\ c_{\eta 2},\ \ldots,\ c_{\eta\beta_\eta})$ and $\pi_\eta = (\pi_{\eta 1},\ \pi_{\eta 2},\ \ldots,\ \pi_{\eta\beta_\eta})$, where β_η is the number of berths in the berthing profile η.

Example 4.5.2. *Consider the same data as in Example 4.5.1 with only processing rates modified as follows:* $p_1 = 219,\ p_2 = 219,\ p_3 = 319,\ p_4 = 319,\ p_5 = 262, p_6 = 262,$ *and* $p_7 = 262$. *Assuming* $\kappa_L = 2,\ \kappa_U = 4$ *and* $\tau_M = 9$, *we have five berthing profiles, namely,*

TABLE 4.4: Berthing Profiles for Example 4.5.1

Berthing Profiles	Berth sections (Cranes) Assigned
$c_1 = (790.8,\ 1109.8),\ \pi_1 = (13,\ 13)$	1 - 13 (1,2,3); 14 - 26 (4,5,6,7)
$c_2 = (1054.4,\ 846.2),\ \pi_2 = (13,\ 13)$	1 - 13 (1,2,3,4); 14 - 26 (5,6,7)
$c_3 = (790.8,\ 582.6,\ 527.2),\ \pi_3 = (13,\ 10,\ 11)$	1 - 13 (1,2,3); 14 - 23 (4,5); 24 - 34 (6,7)
$c_4 = (527.2,\ 846.2,\ 527.2),\ \pi_4 = (13,\ 10,\ 11)$	1 - 13 (1,2); 14 - 23 (3,4,5); 24 - 34 (6,7)
$c_5 = (527.2,\ 527.2,\ 846.2),\ \pi_5 = (13,\ 10,\ 11)$	1 - 13 (1,2); 14 - 23 (3,4); 24 - 34 (5,6,7)

(1076, 786), (757, 1105), (438, 638, 786), (438 , 900, 524) and (757, 581, 524). All of them can be chosen to be potential berthing profiles.

Capacity-Equivalent Berthing Profiles

Say that two berthing profiles are capacity-equivalent if their capacity vectors are equal after arranging their elements in the descending order.

The number of berthing profiles increases with increasing number of cranes and their heterogeneity with respect to their capacities. One way to reduce this number is to consider only one profile from each of the capacity-equivalent classes. This reduces the number substantially.

Example 4.5.3. *Consider the same data as in Example 4.5.1 and take* $\kappa_L = 1$ *and* $\kappa_U = 4$. *In this case, the cranes can be distributed to berths in 52 different ways. But if we take one berthing profile from each capacity-equivalent class, then we have only 22 berthing profiles.*

Without loss of generality, we shall assume that the elements in the berthing profile vectors are in the descending order of their capacities. Further, we designate each berthing profile with an index. Thus, we shall assume $\mathcal{B} = \{1, 2, \ldots, N\}$ as the set of distinct berth profiles, N being the number of distinct berthing profiles.

We are now ready for presenting our formulations. We present two formulations: one for the case of heterogenous cranes and the other for the homogenous cranes. Refer to Table 4.1 for all the notation.

4.5.2 Formulation for Heterogeneous Cranes

In this model, we assume that ship arrival times are the given inputs and no commitments on departure times. No service priorities are assumed. Berth clearance time is assumed to be one time period, that is, if t is the last service time period for a ship, then the berth sections occupied by the ship are available for other ships from time period $t+2$ onwards. Treating $t+1$ as the ship's service completion time, we take the sum of completion times of all ships as the objective function. The formulation is an MILP. Following are the decision variables:

- λ_η is 1 if berthing profile η is selected, 0 otherwise, $\eta \in \mathcal{B} = \{1, 2, \ldots, N\}$,
- $y_{\eta bv}$ is the index variable which is 1 if ship v is served in berth b of berthing profile η, $b = 1, 2, \ldots, \beta_\eta$; $\eta = 1, 2, \ldots, N$, where β_η is the number of berths in the berthing profile η,
- $s_{\eta bv}$ is the service starting time period of ship v in berth b of berthing profile η,
- $x_{\eta bv}$ service duration of ship v in berth b of berthing profile η.

The main idea of our formulation is that we select an appropriate berthing profile for an instance of the problem and serve all the ships according to the selected profile. The model is presented below. The range of b in the formulation is not explicitly stated for brevity. The expression $\forall\ \eta,\ b,\ v$ stands for $\eta = 1, 2, \ldots, N$, $b = 1, 2, \ldots, \beta_\eta$ and $v = 1, 2, \ldots, V$.

Heterogeneous Cranes Model

$$\text{Minimize} \sum_{v=1}^{V} \sum_{\eta=1}^{N} \sum_{b=1}^{\beta_\eta} (s_{\eta bv} + x_{\eta bv}) \tag{4.5.1}$$

subject to

$$\sum_{\eta=1}^{N} \lambda_\eta = 1, \tag{4.5.2}$$

$$\sum_{\eta=1}^{N} \sum_{b=1}^{\beta_\eta} y_{\eta bv} = 1 \quad \forall\ v, \tag{4.5.3}$$

$$s_{\eta bv} \geq a_v y_{\eta bv} \quad \forall\ \eta,\ b,\ v \tag{4.5.4}$$

$$s_{\eta bv} + x_{\eta bv} \leq T y_{\eta bv} \quad \forall\ \eta,\ b,\ v \tag{4.5.5}$$

$$\tau_v y_{\eta bv} \leq \pi_{\eta b} \quad \forall\ \eta,\ b,\ v, \tag{4.5.6}$$

$$\sum_{\eta=1}^{N} \sum_{b=1}^{\beta_\eta} c_{\eta b} x_{\eta bv} \geq q_v, \quad \forall\ v, \tag{4.5.7}$$

$$T(2 - y_{\eta bu} - y_{\eta bv}) + s_{\eta bu} \geq 1 + s_{\eta bv} + x_{\eta bv} - T w_{uv} \quad \forall\ \eta,\ u < v \tag{4.5.8}$$

$$T(2 - y_{\eta bu} - y_{\eta bv}) + s_{\eta bv} \geq 1 + s_{\eta bu} + x_{\eta bu} - T(1 - w_{uv}) \quad \forall\ \eta,\ u > v \tag{4.5.9}$$

$$\sum_{v=1}^{V} \sum_{b=1}^{\beta_\eta} y_{\eta bv} \geq \lambda_\eta \ \forall\ \eta \tag{4.5.10}$$

$$\sum_{v=1}^{V} \sum_{b=1}^{\beta_\eta} y_{\eta bv} \leq V \lambda_\eta \ \forall\ \eta \tag{4.5.11}$$

$s_{\eta bv},\ x_{\eta bv}$ are nonnegative integers, and

$\lambda_\eta,\ w_{uv},\ y_{\eta bv} \in \{0,\ 1\} \quad \forall\ \eta,\ b,\ v.$

Note that (4.5.3) and (4.5.5) ensure that $s_{\eta bv} + x_{\eta bv} = 0$ for all but one $\eta \in \mathcal{B}$. Under this condition, $\sum_{\eta=1}^{N} \sum_{b=1}^{\beta_\eta} (s_{\eta bv} + x_{\eta bv})$ represents the completion time of ship v which includes one additional time period for clearance. Therefore, the objective function is the sum of completion times of all ships. Constraint (4.5.2) ensures that exactly one berthing profile is used, that is, $\lambda_\eta = 1$ for exactly one $\eta \in \mathcal{B}$. Constraint (4.5.3) ensures that $y_{\eta bv} = 1$ for exactly one pair $(\eta,\ b)$, $\eta \in \mathcal{B}$ and $b \leq \beta_\eta$ (that is, ship v is assigned to exactly one berth). We must ensure that this η is same as the η for which $\lambda_\eta = 1$. This is ensured by (4.5.3), (4.5.10) and (4.5.11). Constraint (4.5.4) ensures that the ship service starts only after its arrival. Constraint (4.5.5) ensures that ship v's start time and service duration are accounted under the same berth in the selected profile η (for which $\lambda_\eta = 1$ and $y_{\eta bv} = 1$). Constraint (4.5.6) ensures that ship v's length fits into its assigned berth. Constraint (4.5.7) ensures that the assigned service duration $x_{\eta bv}$ of ship v is adequate to complete the ship's service. Constraints (4.5.8) and (4.5.9) together ensure that service periods of any two ships assigned to the same berth are non-overlapping. Here, the variables w_{uv}s are auxiliary. It may be observed that it suffices to define these variables independent of η and b. This observation helps in reducing the number of variables. Constraints (4.5.10) and (4.5.11) are crucial in this formulation. The idea of utilising these constraints is derived from a formulation to a scheduling

problem in a windmill power application (see constraints (39) and (40) in Lalita and Murthy (2020b)).

Figure 4.3 presents a solution to an instance of the problem with 10 ships. The problem is solved using LINGO. The MILP has 464 variables and 1360 constraints. The optimal solution was found in 48 seconds but it took 97 minutes to confirm optimality. The same problem was tried using DRPF and DRPF++ formulations cited in Agra and Oliveira (2018). The number of variables and constraints in DRPF formulation are 4968 and 2,37,286 respectively, and in DRPF++ they are 7130 and 3,52,751 respectively. At the beginning of this section, we stated that one of the motivations for BCI solutions is their simplicity from practical view point. Example 4.5.4 is a simple illustration of this.

Example 4.5.4. *Crane particulars are as in Example 4.5.1 with only processing rates modified as follows:* $p_1 = 263.6,\ p_2 = 263.6,\ p_3 = 319,\ p_4 = 319,\ p_5 = 212.1, p_6 = 212.1,$ *and* $p_7 = 319$. *Vessel data are as follows: Vessel 1 - length 8, arrival 2, load 4660, Vessel 2 - length 9, arrival 3, load 3810, Vessel 3 - length 8, arrival 9, load 3490. Take* $\kappa_L = 2,\ \kappa_U = 4$.

Figure 4.4 presents two solutions to the problem in Example 4.5.4 - a solution obtained using DRP formulation (objective value 26) and the other using BCI formulation (objective value 28). The movement of cranes in the former solution is rather undesirable while there is a mild compromise on the objective value in going for the BCI solution.

4.5.3 Formulation for Homogeneous Cranes

Many of the cases in the literature deal with homogeneous cranes. Under homogeneous cranes set-up, we propose a simplified formulation to find BCI solutions that avoids the difficulty of enlisting the berthing profiles. In this formulation, we drop the term berthing profiles and use berths instead. This means that the η variable will be dropped completely which will result in a simplified formulation. This is elaborated further before presenting the formulation.

Consider the case where you have $K = 12$ homogeneous cranes and take $\kappa_m = 2$ and $\kappa_M = 4$. In this case, maximum number of berths that one can have is 6 $(= \lfloor K/\kappa_m \rfloor)$.

DRP solution

	1	2	3	4	5	6	7	8	9	10	11	12	13
14			1	1	1	1							
15		Vessel 2	2	2	2	2		Vessel 3					
16		Arrival: 3	3	3	3	3		Arrival: 9	3	3	3		
17		Load: 38.1	4	4				Load: 34.9	4	4	4		
18		Length: 9						Length: 8					
19									6	6	6		
20									7	7	7		
21													
22													
23		1											
24	Vessel 1	2											
25	Arrival: 2	3											
26	Load: 46.6	4			4	4							
27	Length: 8		5	5	5	5							
28			6	6	6	6							
29			7	7	7	7							
30													
31													
32													
33													
34													

Crane	Capacity	Berth Section Range	
1	263.6	1	26
2	263.6	1	26
3	319.0	1	26
4	319.0	1	26
5	212.1	14	34
6	212.1	14	34
7	319.0	14	34

A DRP optimal solution with objective values 26.

BCI solution

	1	2	3	4	5	6	7	8	9	10	11	12	13
14			1	1	1	1	1						
15		Vessel 2	2	2	2	2	2						
16		Arrival: 3	3	3	3	3							
17		Load: 38.1											
18		Length: 9											
19													
20													
21													
22													
23													
24	Vessel 1	4	4	4	4	4		Vessel 3	4	4	4	4	
25	Arrival: 2	5	5	5	5	5		Arrival: 9	5	5	5		
26	Load: 46.6	6	6	6	6			Load: 34.9	6	6	6		
27	Len: 8	7	7	7	7			Length: 8	7	7	7		
28													
29													
30													
31													
32													
33									Objective value =28				
34													

FIGURE 4.4: A comparison of an optimal solution with BCI solution for a problem instance with three ships. The assigned crane numbers are shown in each time period.

Assuming that it is possible to make six berths, each having two cranes, designate these six berths by $b = 1, 2, \ldots, 6$. Next, note that we cannot create 5 berths each having 3 cranes as $K = 12$. Therefore, we can have at most 4 berths each having 3 cranes. Assuming we can make such berths, designate them by $b = 7, \ldots, 10$. Next, consider berths with 4 cranes each. We can have at most 3 such berths. Designate them by $b = 11, 12, 13$. In our formulation below, we shall assume that the berths are designed in this fashion. We drop the use of berthing profiles and use berths instead. The parameter N will now be interpreted as the number of berths. We drop η from all the decision variables and input parameters. Thus, we replace $c_{\eta b}$ and $\pi_{\eta b}$ with c_b and π_b respectively which denote the number of cranes and number of berth sections in berth b respectively. Unlike in the previous formulation, π_bs are decision variables in this formulation. Similarly, we replace the berthing profile selection indicator variable λ_η with berth selection indicator variable λ_b which is 1 if berth b is selected. Similarly, y_{bv} is an indicator variable which is 1 if ship v is assigned to berth b; s_{bv} and x_{bv} stand for the starting time period and service duration of ship v in berth b.

Homogeneous Cranes Model

$$\text{Minimize} \sum_{v=1}^{V} \sum_{b=1}^{N} (s_{bv} + x_{bv}) \tag{4.5.12}$$

subject to

$$\sum_{b=1}^{N} c_b \lambda_b \leq K, \tag{4.5.13}$$

$$\sum_{b=1}^{N} y_{bv} = 1 \quad \forall\ v, \tag{4.5.14}$$

$$s_{bv} \geq a_v y_{bv} \quad \forall \quad b,\ v \tag{4.5.15}$$

$$s_{bv} + x_{bv} \leq T y_{bv} \quad \forall \quad b,\ v \tag{4.5.16}$$

$$\tau_v y_{bv} \leq \pi_b \quad \forall \quad b,\ v, \tag{4.5.17}$$

$$\sum_{b=1}^{N} c_b x_{bv} \geq q_v, \quad \forall\ v, \tag{4.5.18}$$

$$T(2 - y_{bu} - y_{bv}) + s_{bu} \geq 1 + s_{bv} + x_{bv} - T w_{uv} \quad \forall\ b,\ u < v \tag{4.5.19}$$

$$T(2 - y_{bu} - y_{bv}) + s_{bv} \geq 1 + s_{bu} + x_{bu} - T(1 - w_{uv}) \quad \forall\ b,\ u > v \tag{4.5.20}$$

$$\sum_{b=1}^{N} \pi_b \leq B, and \tag{4.5.21}$$

$$\sum_{v=1}^{V} b y_{bv} \geq \lambda_b, \forall \quad b, \tag{4.5.22}$$

$$\sum_{v=1}^{V} b y_{bv} \leq NV \lambda_b, \forall \quad b, \tag{4.5.23}$$

$s_{bv},\ x_{bv},\ \pi_b$ are nonnegative integers,

$\lambda_b,\ w_{uv},\ y_{bv} \in \{0,\ 1\} \quad \forall \quad b,\ v.$

The objective function and constraints (4.5.14) to (4.5.20) can be interpreted as in the previous formulation. Constraint (4.5.13) ensures that the total number of cranes used in the selected berths does not exceed the available number of cranes, and constraint (4.5.21) ensures that the total number of berth sections used in the selected berths does not exceed the number of berth sections in the quay. Constraints (4.5.22) and (4.5.23) together ensure that a berth is created if and only if at least one ship is assigned to it. In Subsection 4.6.2 we will present results for the application of the above formulation.

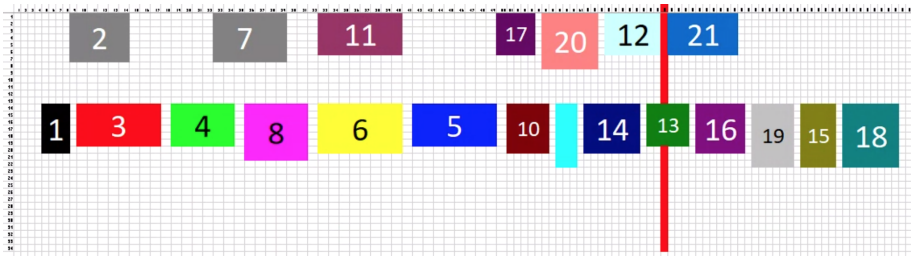


FIGURE 4.5: Solution to first subproblem with 21 ships. The red solid line is time period 73.

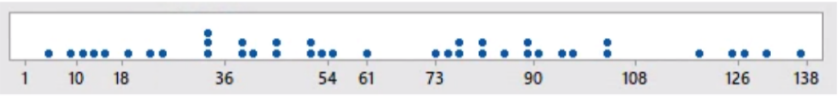
4.5.4 Expanding the BCI Class

While the performance of BCI solutions appears to be satisfactory (see the next section for the results of our numerical experiments), it is desirable to enlarge the class to include better solutions. Towards this, we propose an approach to expand the BCI class in this section. The idea is to break the planning horizon into two or three pieces at appropriate places and use a BCI solution for each piece. The method will be described with an example.

Consider BACASP with 40 ships with 168 one-hour time periods over one week. The data are presented in Table 4.5 along with a picture of arrivals. We start by breaking the

TABLE 4.5: Data for 40-ship instance

v	τ_v	q_v	a_v	v	τ_v	q_v	a_v	v	τ_v	q_v	a_v	v	τ_v	q_v	a_v
1	7	35	5	11	6	58	33	21	6	74	61	31	7	74	90
2	7	45	9	12	6	57	39	22	7	32	73	32	9	67	95
3	6	87	10	13	6	58	39	23	7	51	74	33	8	34	96
4	6	64	13	14	7	83	40	24	7	69	76	34	6	41	102
5	6	84	15	15	9	52	44	25	9	68	76	35	9	46	102
6	7	86	18	16	7	76	45	26	8	84	80	36	7	36	118
7	7	51	23	17	6	36	50	27	6	30	81	37	8	62	125
8	8	61	24	18	9	84	50	28	7	35	85	38	8	83	126
9	9	31	32	19	9	66	53	29	7	35	89	39	7	46	131
10	7	59	32	20	8	57	54	30	6	81	89	40	6	62	138



problem into two subproblems. From the picture of arrivals, we see that there is a large gap between the 21st arrival ($a_{21} = 61$) and 22nd arrival ($a_{22} = 73$). So, we consider two subproblems. In the first subproblem we consider only ships 1 to 21 and solve it. The solution to the first subproblem is shown in Figure 4.5. The earliest arrival time of

	Subproblem 1	Subproblem 2
Time Periods	1 to 65	66 to 168
Berths	Berth 1: Berth sections: 1 to 13 cranes: 1, 2, 3 Vessels: 10, 11, 12, 14, 17, 20 Berth 2: Berth sections: 14 to 26 Cranes: 4, 5, 6, 7 Vessels: 1, 2, 3, 4, 5, 6, 7, 8, 9	Berth 1: Berth sections: 1 to 13 cranes: 1, 2, 3,4 Vessels: 13, 15, 22, 23, 25, 26, 27, 29, 30, 34, 35, 36, 37, 38 Berth 2: Berth sections: 14 to 26 Cranes: 5, 6, 7 Vessels: 16, 18, 19, 21, 24, 28, 31, 32, 33, 40

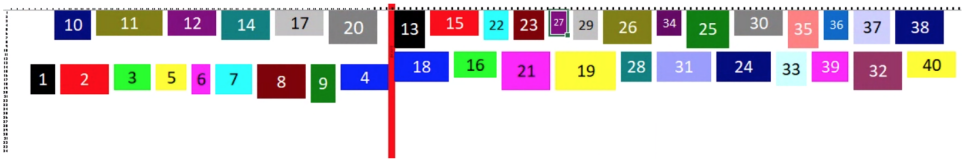


FIGURE 4.6: Final solution for the 40 ship problem

FIGURE 4.7: One-shot solution for the 40 ship problem

the second set of ships is 73. We take the planning horizon for the second subproblem as 74 to 168 (time period 73 is left out for clearance). Since some or all of service periods of ships 13, 15, 16, 18, 19 and 21 are falling into the planning horizon of second subproblem, we delete these ships from the first subproblem and include them into the second subproblem. Before solving the second subproblem, the first problem is solved once again by dropping the six ships just mentioned. The last service time period in the resulting solution turned out to be 64. Leaving one time period for clearance, we revise the planning horizon of the second subproblem as 66 to 168. Further, we change the arrival times of the six ships to 66 for solving the second subproblem.

When the problem is solved as a whole, the solver reported infeasibility. After extending the planning horizon to 200, the solver produced a feasible solution within a minute. Best solution after four hours had objective value 4064 with three ships (38, 40 and 12) assigned beyond time period 168. Optimal solution used two berths - one with cranes 1, 2, 3 and 4, 5, 6, 7. Figure 4.7 presents the solution pictorially.

4.6 Numerical Experiments

In this section we shall find BCI solutions to a number of instances. For all the instances, we consider a planning horizon of 168 time periods and assume that all arrivals occur during the time periods 1 to 150 as in Agra and Oliveira (2018). As the number of ships increase, it would be difficult to fit them in the planning horizon with limited number of cranes and berth sections (limited quay length). It will be necessary to consider the instances with larger number of cranes and berth sections. Wang et al. (2018) consider BACASP along with integration of yard management. They solve instances involving number of ships up to 60. Therefore, for larger number of ships, we have taken number of cranes and berth sections approximately similar to the instances considered in Wang et al. (2018). In Subsection 4.6.1, we present the results of instances with heterogeneous cranes. The results for homogeneous cranes are presented in Subsection 4.6.2. We used LINGO 13.0 commercial solver on a computer with a CPU Intel(R) Core i7, with 16 gigabytes RAM to solve the instances.

4.6.1 Results for Instances with Heterogeneous Cranes

In Agra and Oliveira (2018), a number of instances were simulated for BACASP with number of ships in the range of 7 to 40. For all these instances, simulation inputs are as follows: (i) the number of berth sections is 34 and the number of cranes used is 7 with particulars as in Table 4.3, (ii) ship lengths varied from 6 to 9 berth sections, ship workloads (cargo) varied from 3000 tons to 8800 tons, and (iii) arrival times from 1 to 150. For our numerical experiments of this subsection, we have considered two sets of instances. For the first set, we have used the input parameters exactly as Agra and Oliveira (2018) described in the above three steps. For the second set of instances we have changed only the capacities of the cranes to test BACASP instances with a higher heterogeneity among cranes.

In the first set, we have considered the instances with cases of ship numbers 15, 20, 30 and 40 (to have correspondence with the results of Table 5 in Agra and Oliveira (2018)). With regard to simulation of arrival times, we simulated them completely randomly. In their case, they fixed the initial arrivals selectively (for some arrivals) to test the

performance of their methods. Since we do not have the complete data for instances of Agra and Oliveira (2018), a direct evaluation of BCI solutions is not possible. However, a broad picture can be assessed based on repeated instances. The results for the first set of instances are presented in Table 4.6. Comparing the results for 15, 20 and 30 ship cases (compare the average columns in Table 4.6), the performance is reasonably good. Figure 4.8 presents the time to reach optimal or near optimal solutions.

In case of instances with 40 or more ships, solver exhibited infeasibility or was not producing feasible solutions in many cases. This is expected because we are trying to accommodate 40 ships in 168 time periods with just seven cranes. We had to break the problem into two subproblems to solve the instances. We had terminated the solver to find near optimal solutions as the solver could not find optimal solutions. Optimality percentage is computed using the formula $100 - \frac{100(BFS-LB)}{LB}$ where BFS is the objective value of the best feasible solution obtained by the solver at the time of terminating the solver, LB is the lower bound produced by the solver at termination. For the case of 40-ship instances, the objective values reported in Table 4.6 are the best solutions found within 6 minutes. Comparing with the lower bounds provided by the solver, these solutions are at least 80% optimal (the actual percentages are 87, 95, 81, 80 and 88).

TABLE 4.6: Results for first set of instances

No. of	Instances						Agra (2018)			
ships	1	2	3	4	5	Avg	a	b	c	Avg
15	1264	1303	992	1024	1194	1155	1295	1331	1149	1258
20	1619	1689	1662	1666	1677	1663	1680	1654	1492	1609
30	2630	2764	2634	2555	2143	2454	2408	2313	2343	2355
40	3579	3645	3516	3619	3871	3646	2931	3253	3156	3113

For the second set of instances, we have taken the crane capacities as 219, 219, 263, 263, 263, 319 and 319 in that order. For this case, when the number of ships is 40, the solver exhibited infeasibility consistently. Once again, this is expected as the number of cranes is inadequate and added to this, the crane capacities have also been lowered for two of the cranes compared to the earlier situation. The solver is stopped when the solution is at least 95% optimal. The results for other instances with ship numbers are presented in Table 4.7. Figure 4.9 presents the times to reach at least 95% optimal solutions.

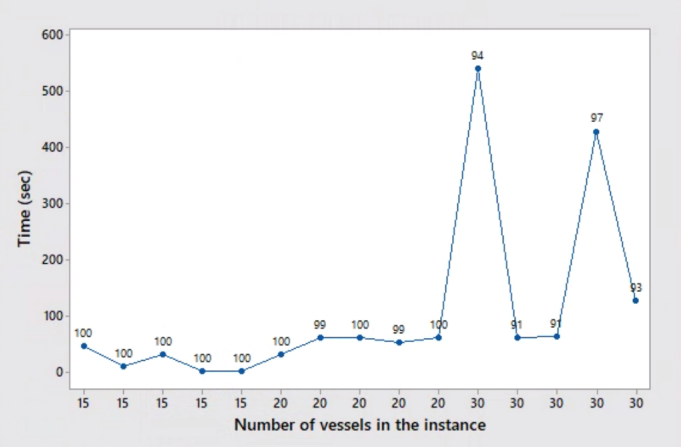


FIGURE 4.8: Time in seconds to reach optimal or near optimal solutions. The data labels in the figure indicate the minimum optimality percentages.

TABLE 4.7: Objective values of best feasible solution with at least 95% optimal

No. of ships	1	2	3	4	5	Avg
15	1354	993	1021	1196	1292	1171
20	1512	1684	1680	1832	1437	1629
25	2230	2063	1943	1917	1933	2017
30	2354	2583	2607	2622	2872	2608

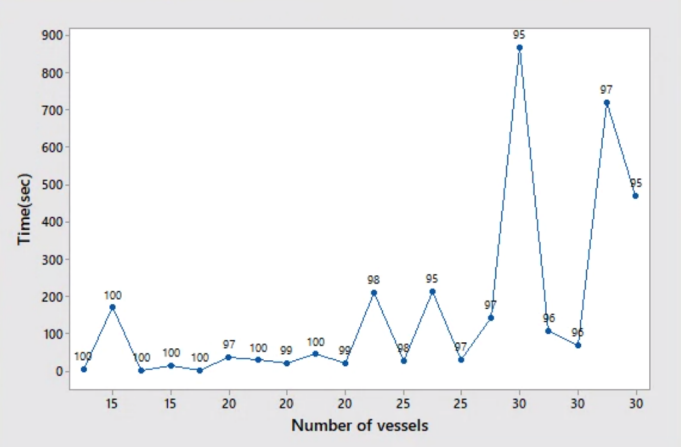


FIGURE 4.9: Results with crane capacities 219, 219, 263, 263, 263, 319 and 319. Time (seconds) is to reach optimal or near optimal solutions. The data labels in the figure indicate the minimum optimality percentages.

4.6.2 Results for Instances with Homogeneous Cranes

In this section also, we present results for two sets of instances with homogeneous cranes. The first set corresponds to results presented in Table 6 of Agra and Oliveira (2018) and the other set corresponds to instances with large number of ships. For all these instances,

the common crane capacity is taken as 263.6 tons per hour. Results for the first instance are presented in the last two columns of Table 4.8. The other columns are extracted from Table 6 of Agra and Oliveira (2018) and presented for broad comparison. As before, since we do not have the data for the instances of Agra and Oliveira (2018) instances, the results cannot be compared directly but give an approximate picture.

TABLE 4.8: Results for first set of instances with homogeneous cranes

			RPF		RHH1		DRPF++		BCI	
Inst	v	Opt	BFS	Time	BFS	Time	BFS	Time	BFS	Time
I1	7	125	125	1571	125	6	115	10	133	30
I2	8	223	224	3600	223	4	207	8	238	1
I3	8	129	135	3600	130	17	108	172	126	20
I4	10	328	347	3600	328	7	307	29	339	16
I5	10	204	259	3600	205	24	174	2	205	15
I6	12	441	459	3600	442	20	416	144	433	3
I7	12	283	414	3600	285	27	251	702	294	6
I8	15	562	576	3600	562	25	531	234	531	14
I9	15	506	542	3600	514	27	465	1555	566	2

Note: Inst - instance code as in Agra (2018); v = number of ships; Opt is optimum objective value; BFS is objective value at the termination the time.

In the second set of instances, we have simulated data for larger number of ships (35,40,45, 50 and 60). As we have observed earlier, when the number of ships is large (40 or above), we require more cranes and berth sections. Therefore, we took the parameters similar to the ones used in Wang et al. (2018). In Wang et al. (2018), discrete BACASP was considered along with integration of yard management. They consider homogeneous cranes with prefixed QC profiles from which selections are made. We have used the formulation for homogeneous cranes presented in Subsection 4.5.3 to find BCI solutions to the BACASP problems. For our purpose, we took the number of cranes and number of berths from their instances for given number of ships. Then, from the number of berths we took number berth sections as 9 times the number of berths or slightly less than that. The arrivals are simulated between 1 and 150 randomly and the workloads between 3000 tons and 9000 tons. The solver was terminated when near optimal solutions were obtained. Optimality percentage is calculated as explained earlier. When the number of ships was 60, the solver was not producing feasible solution. Therefore, for this case we adapted a split technique which is explained below. The number of time periods in the planning horizon for all the instances is 168.

When the number of ships is large, split the problem into subproblems. Consider the case of 60 ships where the quay length is also 60. Split the quay into two parts, first part from 1 to 30 berth sections and the second part consisting of berth sections 31 to 60. Distribute the ships into two groups so that ships in each group have less number of ships wit h closer arrival times. Then solve the two subproblems independently. Finally, club the two solutions to get a solution to the original problem. The problem with 60 ships shown in Table 4.9 is solved in this fashion. Figure 4.10 presents the solution thus obtained.

TABLE 4.9: Results for second set of instances with homogeneous cranes

v	N	K	B	BFS	LB	Opt %
35	5	12	40	3101	2654	83
40		14	45	3629	3486	96
45	6	16	50	4115	3966	96
50	7	18	60	4599	4425	96
60	8	21	60	4887	4551	93

v =number of ships; N = number of berths; K = number of cranes;
BS: berth sections; Opt: optimality %; BFS: objective value of best feasible solution at termination.

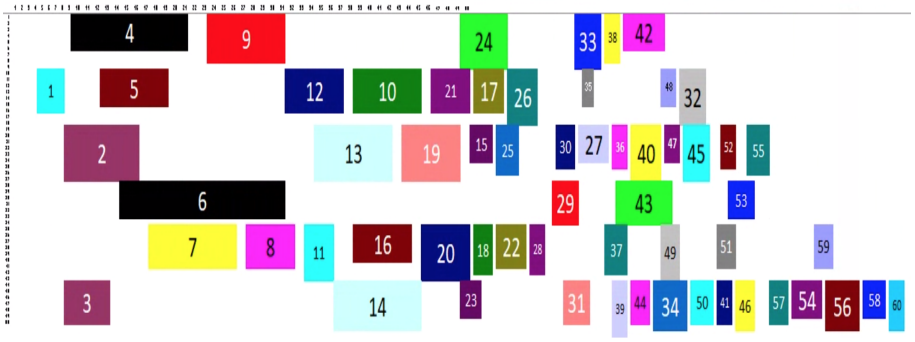


FIGURE 4.10: Solution to the 60-ship problem solved using split technique.

4.7 Summary

Berth allocation, crane assignment and crane scheduling problem is an important problem for ship transportation and has been studied extensively from optimization perspective. A number of variants of the problem have been studied in the literature and solution

methodologies are explored. We have considered the continuous berth allocation with non-crossing constraints. Existing integer programming formulations for solving the problem are complex and are not suitable for commercial solvers to find solutions to practical problems particularly when the number of ships is large. In this chapter, we have introduced a new class of solutions, the BCI class, for the problem and proposed a compact integer linear programming formulation. The size of the problem (in terms of number of variables and constraints) is much smaller compared to the formulations for finding the global optimal solutions. Though BCI solutions are suboptimal, their performance appears to be satisfactory with respect to the objective value of service completion time. We have conducted a number of numerical experiments to evaluate this aspect. Further, the speciality of these solutions is that they are desirable from the view point of ease of implementation.

Chapter 5

Extension of Resource Scheduling Models to Wind Power Scheduling

5.1 Introduction

This study deals with providing methodology for determining strategies for optimal utilization of wind power generation by private power producers under government controlled trading mechanism. It helps in promoting renewable energy generation which is gaining importance at an exponential rate. Several nations have come together in combating the global warming which is posing a serious threat to mankind (Paris Agreement at https://treaties.un.org). Renewable energy generation is a great source for mitigating global warming (see https://www.ucsusa.org/resources/benefits-renewable-energy-use). Besides mitigating global warming, it has several other benefits such as power supplies to rural areas and industries, new employment opportunities, etc. Figure 5.1 exhibits the windmill growth across the world over the past two decades (see https://ourworldindata.org/renewable-energy). As of 2018, the global wind power capacity stood at 591 GW (9.6% higher compared to the previous year) and contributed to 4.8% of total electric power consumption.

Following a spike in industrial energy consumption (at an average annual growth of 3.9% over the period 2010 to 2017 (International Energy Agency, Paris (2019)), India advocated aggressive measures to promote renewable energy generation to reduce its share of carbon emissions, and made changes to its National Electricity Policy. As of 2019, 35% of India's installed electricity generation capacity is from renewable sources, generating 17% of total electricity in the country (International Energy Agency (2019)). The policy provides incentives to independent power producers (IPPs) to produce and supply power from renewable sources (Ministry of Power (2005)). This has lead to establishment of green power industries generating clean power commercially. One of the measures initiated, the inter state transmission system (ISTS), allows industries to produce clean power in any state and utilize power from government grid in that state or other states according to an exchange trade mechanism with incentives of subsidised tariffs. This chapter deals with a decision making problem for a large paper manufacturing company that is involved in this exchange trade mechanism. The company has set up windmills at ideal locations, usually hilltops, which are far from the paper mills of the company.

As per ISTS, the company supplies power from its windmills to the local grid and draws power from grid nearer to its paper mills. According to the trade mechanism, the company has to announce supply schedules to the grid at windmills 48 hours in advance. The extent of power drawn from the grid at the paper mills during the same schedule period is regulated by the schedule announced at the windmills. The decisions to be made on the supply schedule and the power to be drawn from the grid nearer to the paper mills depends on varying power demands of the paper mills and the deviations in the actual supplies from the supply schedules committed at the windmills (the details are described in Section 5.3).

The current literature on green power supply trading deals with schedule preparation (known as unit commitment) and its management only. It has no exchange mechanism business. As wind power largely depends on weather conditions, it fluctuates very much over shorter periods of time. To circumvent this and supply regulated power, power storage systems are deployed. The decision making problems, then, revolve around the power generation control mechanisms and appropriate schedule preparations. The problem studied in this chapter is new. The decision variables in the problem are: (i)

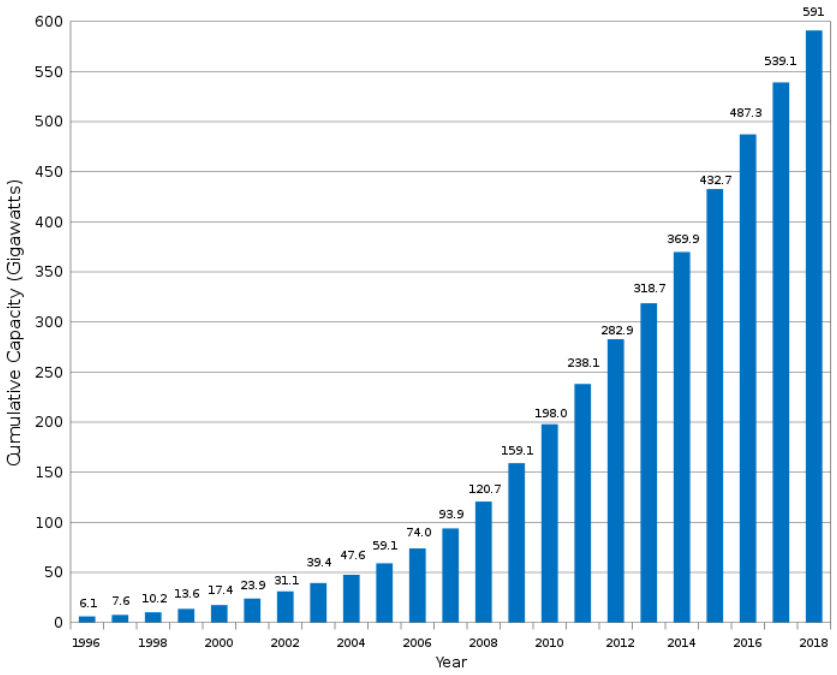


FIGURE 5.1: Historic Development of Global Installation Capacity

the 15-minute power supply schedule over one-day planning horizon and (ii) the draw schedule at the paper mill which involves deciding on splitting the planning horizon into unknown spell lengths and the corresponding draw rates from the grid. The problem turns out to be a combinatorial optimization problem with non-convex quadratic objective function. We tried to formulate the problem using dynamic programming model. But one of the constraints of the problem prevents this approach. The problem in its original form has decision variables that are functions of other decision variables. It is not amenable to standard models available with the professional OR packages.

The main contribution of this chapter is our innovative approach in modelling the problem. Our formulation uses a technique that was developed recently in the context of solving task and staff scheduling problems (Lalita and Murthy (2020a)). We study the properties of the solution to the windmill problem and provide necessary and sufficient conditions for optimality of the solutions. Solutions to a number of real-life instances of the problem are presented. An interesting side benefit of this application is that we can extend this methodology to a problem of scheduling airport check-in counters for departures (Lalita et al. (2020)).

The organization of the chapter is as follows. Section 5.2 presents a literature review

on the windmill power optimization problems. Section 5.3 presents the description and formulation of the problem. Section 5.4 presents some properties of the solutions of the problem. Section 5.5 presents a reformulation of the problem that is amenable for solving it. Section 5.6 presents the results of applying our methodology to real-life data from the company. The chapter is concluded in Section 5.7 with a brief summary.

5.2 Literature Review

Most of the studies on wind power optimization in the existing literature deal with unit commitment (UC) scheduling and handling the fluctuations in the power generation (see Abujarad et al. (2017)). At the receiving end, the grid needs power for its end users' requirements which varies over time. This in turn gives rise to a demand pattern over time. As wind power fluctuates with weather conditions, matching the required demand pattern with wind power alone is not possible. To overcome this problem, the IPPs use power from other sources in combination with generated wind power to match the grid demand pattern. The choice of alternate source for balancing the supply depends upon factors such as installed windmill capacity, capacity of alternate sources, availability, etc. IPPs with high windmill capacities are prone to high fluctuations and therefore should have access to alternate sources with high balancing capacities. Some of the studies in the literature focus on problems of using wind power with alternate sources such as thermal power (Reddy (2017)), solar power (Liang and Liao (2007)), pumped storage units (Khatod et al. (2013)), energy storage devices (Tanabe et al. (2008), Korpaas et al. (2003) or all of the above (Wang et al. (2013)). Wang et al. (2013) have modeled the problem for an IPP with generation portfolio consisting of thermal, hydro and wind power units and applied sample average approximation to obtain optimal biding strategies for price based unit commitment of power. Liang and Liao (2007) propose a fuzzy optimization approach for scheduling wind power generation. Their objective is to minimize the thermal unit fuel cost in the presence of hydro, solar and wind energy sources. Makarov et al. (2011) consider additional sources of uncertainty in the system and therefore take into account the capacity, ramp rate and the ramp duration requirements. Their model calculates the uncertainty ranges of each parameter and ensures a reliable supply of dispatch at different time horizons. In other studies,

optimal reserves to be maintained in fuel cells are computed. Fuel cells are increasingly being used for maintaining energy reserves and in ensuring constant power supply from the wind mills (Tanabe et al. (2008)). Su et al. (2014) schedule microgrid energy in the presence of renewable energy sources, storage devices and plug-in electric vehicles. Korpaas et al. (2003) focus on scheduling and operation of energy storage for wind power plants. Venkatesh et al. (2011) propose a mixed integer linear programming model for frequency based unit commitment for a state utility in the presence of a pumped hydro unit. Possible hourly energy deficit and associated drop in frequency is also modelled. Catalão et al. (2012) provides optimal offering strategy for wind power producers to offer wind energy in the day ahead market. In all the above studies, optimization methods are used to optimally commit wind power with the objective of minimizing variation in wind power supply from a wind power plant. Arlitt et al. (2012) propose methods to achieve a net zero operations at a data center, i.e., total power consumption is less than total renewable energy supply, given storage mechanism at the data center. There are very few studies that attempt to match renewable supply to demand in the absence of storage facilities. Following are some studies related to intermittent power scheduling without storage mechanisms. Goiri et al. (2011), Goiri et al. (2015) propose a job scheduler 'Greenslot' for a datacenter powered by solar energy and grid power. Jobs are scheduled to maximize green energy consumption. The advantage of this type of job scheduling in the absence of energy storage is that green energy wastage is minimized and cost of battery storage and energy losses due to resistance in batteries etc are not present. The authors aim at reducing costs while meeting as many deadlines as possible. Li et al. (2012) propose a supply switching scheduler - iSwitch. The authors discuss a supply optimization scheme and a demand smoothing scheme on load-side. The authors propose load matching with the energy source in operations of data centers. At times of low wind power, power supply source switches to utility draw and when enough wind power is available, the power supply source switches again. This is the first paper to introduce a supply-load cooperative scheme. Moreover, these schedulers match variable demand to intermittent supply.

From our literature study, we have not found any article that deals with the type of problem considered in this chapter. The main distinction of our problem with those considered in the literature is that we have the freedom of defining spells in an optimal

way. This problem does not arise for the problems in the current literature. Our methodology may be useful in the selection of alternate sources by way of determining desirable spells in the planning horizon. We will take up this problem in a separate study.

5.3 Problem Description and Formulation

For ease and convenience of description and presentation of the problem, the basic terminology and notation are summarized in Table 5.1. Refer to this table for the terms used in this chapter. Unless stated otherwise, the terms supply, demand and draw stand for the rate of electrical power measured in megawatts (MW). The decision making problem for the company in question involves two establishments: (i) a windmill that generates power and supplies to the grid at the windmill location, and (ii) a paper mill that draws power from the grid $\mathcal{G}$ at the paper mill location. Decisions are made for one day planning horizon T based on forecasted power generation $\hat{x}_j$s at the windmill. The decisions are to be made two days ahead of time. At the windmill, the company has to declare the power supply $(s_1, s_2, \ldots, s_{96})$ to the distribution company (DISCOM). At the paper mill, the company is allowed to draw power from the local grid $\mathcal{G}$ during the planning horizon. For this, the company divides the planning horizon into a *split* $\widetilde{h} = (\widetilde{h}_1, \widetilde{h}_2, \ldots, \widetilde{h}_k)$. Due to operational constraints, the *spell lengths* must be at least 16 and the number of spells cannot exceed 4 (this essentially means that draw rate cannot be changed more than three times a day). According to the power trade-off rules, the paper mill can draw power from $\mathcal{G}$ and the draw rate α_i during any spell $\widetilde{h}_i$ cannot exceed s_j in time period j.

Operations at the Mills

At windmill, the company has 16 windmill turbines each with a capacity of 2 MW of power generation. Actual power supply rate of all turbines put together, x_j, in time period j depends upon the prevailing wind forces. Using a statistical mechanism, x_js are forecasted (as $\hat{x}_j$s) two days ahead of time. As there is no storage facility at the windmill, the generated power x_j is supplied to the DISCOM. Since the declaration of s_js is to be made 48 hours in advance, the decisions are made based on the assumption

TABLE 5.1: Notation

Notation/term	Description
$\mathcal{B}$ and $\mathcal{G}$	$\mathcal{B}$ paper mill power plant, $\mathcal{G}$ power grid at paper mill
$U_l,\ l = 0, 1, 2, \ldots, m$	Power consuming units at the paper mill; U_0 is static which cannot be connected to grid directly; other U_ls are dynamic which can be connected either to $\mathcal{B}$ or to $\mathcal{G}$.
$T = \{1, 2, \ldots, 96\}$	Planning horizon: 15-minute time periods starting from 00:00 hours; j stands for j^{th} time period, $j = 1, 2, \ldots, 96$
Split $\widetilde{h} = (\widetilde{h}_1, \widetilde{h}_2, \ldots, \widetilde{h}_k)$, spells ($\widetilde{h}_i$s) and spell lengths h_is	A split $\widetilde{h}$ is a break up of planning horizon into blocks of contiguous time periods, $\widetilde{h}_1 = [1, 2, \ldots, h_1]$, $\widetilde{h}_2 = [h_1 + 1, h_1 + 2, \ldots, h_1 + h_2], \ldots, \widetilde{h}_k = [h_1 + \ldots + h_{k-1} + 1, \ldots, 96]$. Each block $\widetilde{h}_i$ is called a spell; h_i is the length of spell $\widetilde{h}_i$. Example: $\widetilde{h} = (\widetilde{h}_1, \widetilde{h}_2, \ldots, \widetilde{h}_4)$, $\widetilde{h}_1 = [1, \ldots, 24]$, $\widetilde{h}_2 = [25, \ldots, 44]$, $\widetilde{h}_3 = [45, \ldots, 70]$, $\widetilde{h}_4 = [71, \ldots, 96]$.
Indices $i,\ j,\ l$	i stands for spell number, j for time period and l for power units at paper mill.
q_{lj}	Power demand (in MW) of unit U_l in time period j.
$Q_j = \sum_{l=0}^{m} q_{lj}$	Total demand of all units in time period j.
$\overline{Q}_j(\alpha_i)$	Effective draw from $\mathcal{G}$ during spell $\widetilde{h}_i$, where α_i is the maximum allowed draw from $\mathcal{G}$ during spell $\widetilde{h}_i$.
α_0	Threshold for connecting $\mathcal{G}$ and $\mathcal{B}$; if $\alpha_i > \alpha_0$, then connect $\mathcal{G}$ to $\mathcal{B}$, else connect $\mathcal{G}$ to selected dynamic units directly.
$\alpha_{01}, \alpha_{02}, \ldots, \alpha_{0g}$	Possible demands from dynamic units; $0 < \alpha_{0b} < \alpha_{0(b+1)}, b = 1, 2, \ldots, g-1,\ \alpha_{0g} = \alpha_0$.
$\eta_{\mathcal{B}}$	Cost of paper mill power in rupees per megawatt hour (MWh).
$\eta_{\mathcal{G}}$	Cost of grid power in rupees per megawatt hour (MWh).
$\widetilde{s} = (s_1, \ldots, s_{96})$	s_j is power supply committed for time period j at windmill.
$\widetilde{\alpha} = (\alpha_1, \ldots, \alpha_k)$	α_i is the maximum power draw allowed from $\mathcal{G}$ during spell i, $i = 1, \ldots, k$.
$\widetilde{x} = (x_1, \ldots, x_{96})$	x_j is the actual power supply during time period j at windmill.
$\widetilde{x}_i,\ \widetilde{s}_i$	Subvectors of $\widetilde{x}$, $\widetilde{s}$ confined to $\widetilde{h}_i$ respectively, $i = 1, \ldots, k$.
$\hat{x}_j,\ j = 1, 2, \ldots, 96$	Forecast of x_j.
$\widetilde{y} = (y_1, \ldots, y_k)$	$y_i = 1$ if $\mathcal{G}$ is connected to $\mathcal{B}$ during spell $\widetilde{h}_i$, $= 0$ otherwise.
$\widetilde{u}_i = (u_{i1}, \ldots, u_{ig})$	$u_{ib} \in \{0,\ 1\},\ \sum_{b=1}^{g} u_{ib} \leq 1$.

that $x_j = \hat{x}_j$ for all j. The deviation $d_j = x_j - s_j$ results in a penalty/gain to the company during each time period j. This penalty/gain is governed by a continuous piecewise linear function given by (5.3.1). Note that gain is treated as negative penalty.

$$C_p(d) = \begin{cases} 10336 - 1105(d + 11.2), & \text{if } -32 \leq d < -11.2, \\ 7072 - 1020(d + 8), & \text{if } -11.2 \leq d < -8, \\ 4080 - 935(d + 4.8), & \text{if } -8 \leq d < -4.8, \\ -850d, & \text{if } -4.8 \leq d < 0, \\ -850d, & \text{if } 0 \leq d < 4.8, \\ -4080 - 765(d - 4.8), & \text{if } 4.8 \leq d < 8, \\ -6528 - 680(d - 8), & \text{if } 8 \leq d < 11.2, \\ -8704 - 595(d - 11.2), & \text{if } 11.2 \leq d \leq 32. \end{cases} \tag{5.3.1}$$

At the paper mill, the company has its own power plant $\mathcal{B}$, known as *boiler plant*, generating thermal power produced from the lignin of wood. In addition, the company has the option to use power from the local grid $\mathcal{G}$ in a limited way according to the trade-off rule (the draw rate in any time period j cannot exceed s_j). Further, due to operational constraints, the power draw α_i during any spell $\widetilde{h}_i$ cannot be changed. The demand for power at the paper mill arises from its units U_l, $l = 0, 1, 2, \ldots, m$. The units are classified into two categories - the *static units* and the *dynamic units*. The static units cannot be connected to $\mathcal{G}$ directly, whereas the dynamic units can be connected to either $\mathcal{B}$ or $\mathcal{G}$ but their connection cannot be interrupted during a spell. For the purpose of this study we may assume that there is only one static unit, U_0. Unit U_l requires power at the rate of q_{lj} MW in time period j. With regard to using grid power, the company has the following options. During any spell $\widetilde{h}_i$ of a split, $\mathcal{G}$ is connected to $\mathcal{B}$ or a selected subset of dynamic units are connected to $\mathcal{G}$ directly. When $\mathcal{G}$ is connected to $\mathcal{B}$, $\mathcal{B}$ incurs an additional consumption of 2 MW to synchronize with the grid frequency. The effective utilization of draw from $\mathcal{G}$ depends on the type of connection. If α_i is the draw from $\mathcal{G}$ during a spell $\widetilde{h}_i$, it is fully utilized when $\mathcal{G}$ is connected to $\mathcal{B}$; otherwise its utilization is limited only to the demand of the dynamic units connected to $\mathcal{G}$ during the spell.

Cost Trade-off and Objective

According to the existing trade-off rules under ISTS, the company can draw power from $\mathcal{G}$ in lieu of the power supply at windmill. In order to avail this facility, the company incurs a cost of $\frac{\eta_{\mathcal{G}}}{4}\sum_{j=1}^{96} s_j$ at the windmill. Since $\eta_{\mathcal{B}} > \eta_{\mathcal{G}}$, by declaring higher s_js, the company can draw more power from $\mathcal{G}$ at the paper mill at a lower cost. But this results in higher payment at the windmill. At windmill, the company has to pay for $\sum_{j=1}^{96} s_j$ plus the penalty for the deviations. The total cost to the company at windmill is given by

$$\phi_W(\widetilde{s}) = \frac{\eta_{\mathcal{G}}}{4}\sum_{j=1}^{96} s_j + \sum_{j=1}^{96} C_p(x_j - s_j). \tag{5.3.2}$$

On the other hand, the cost to the company at paper mill in a spell $\widetilde{h}_i$ depends on the power generated from $\mathcal{B}$ which in turn depends on the power draw from $\mathcal{G}$. Based on the utilization possibilities, there is a threshold α_0 for draw from $\mathcal{G}$. If the draw from $\mathcal{G}$ is less than or equal to α_0, then some selected dynamic units are connected to $\mathcal{G}$ directly to utilise as much of power from $\mathcal{G}$ as possible; other wise $\mathcal{G}$ is connected to $\mathcal{B}$. In the present problem $\alpha_0 = 14.3$.

Consider a spell $\widetilde{h}_i$ of length h_i. According to the trade rules, draw from $\mathcal{G}$ in the spell cannot exceed minimum of s_js in the spell. If this minimum is less than the threshold α_0, then the effective draw from $\mathcal{G}$ is limited to the demand of a selected subset of dynamic units so that their demand is closest to the minimum of s_js in the spell. The demand from paper mill during a time period j in the spell is equal to Q_j. The power generation requirement from $\mathcal{B}$ during time period j is equal to Q_j minus the *effective draw* of the power scheduled from $\mathcal{G}$. The effective draw is a function of the minimum of s_js and the threshold α_0. This function is given by (5.3.3).

$$\overline{Q}_j(\alpha_i) = \begin{cases} \alpha_i - 2, & \text{if } \alpha_i > \alpha_0 \\ \underset{V\subseteq\{1,2,\ldots,m\}}{max}\{\mu = \sum_{l\in V} q_{lj} : \mu \le \alpha_i,\}, & \text{otherwise.} \end{cases} \tag{5.3.3}$$

When $\alpha_i > \alpha_0$, the effective draw is reduced by 2MW (lost for synchronization). For $\alpha_i \le \alpha_0$, the effective draw is a step function of α_i. To see this, let $0 = \alpha_{01} < \alpha_{02} < \ldots < \alpha_{0g} = \alpha_0 - 2$ be the distinct possible values of $\{\mu = \sum_{l\in V} q_{lj} : \mu \le \alpha_i, V \subseteq \{1, 2, \ldots, m\}\}$.

Then, we can rewrite

$$\overline{Q}_j(\alpha_i) = \begin{cases} \alpha_i - 2, & \text{if } \alpha_i > \alpha_0 \\ F(\alpha_i), & \text{otherwise,} \end{cases} \tag{5.3.4}$$

where

$$\begin{aligned} F(\alpha_i) &= \max\{\alpha_{0b} : \alpha_{0b} \leq \alpha_i,\ 1 \leq b \leq g,\} \\ &= \max\left\{\mu : \mu = \sum_{b=1}^{g} \alpha_{0b} u_{ib} \leq \alpha_i,\ \sum_b u_{ib} = 1, u_{ib} \in \{0,1\}\right\} \end{aligned} \tag{5.3.5}$$

Using indicator variable y_i which is 1 if $\alpha_i > \alpha_0$, we can write

$$\overline{Q}_j(\alpha_i) = y_i(\alpha_i - 2) + (1 - y_i)F(\alpha_i) \tag{5.3.6}$$

When $y_i = 1$ we must have $u_{ib} = 0$ for all b. In other words, we must have $y_i + \sum_{b=1}^{g} u_{ib} = 1$ or $\sum_{b=1}^{g} u_{ib} = 1 - y_i$ for each i. Further, at the border case $\alpha_i = \alpha_0$, the values of the two expressions of (5.3.4) are equal. Therefore, the effective draw in this case is same irrespective of whether $y_i = 0$ or $y_i = 1$. Note that $\overline{Q}_j(\alpha_i)$ is the reduction in the power generation demand from $\mathcal{B}$, and hence it should be maximized as grid power is cheaper compared to plant power. The cost to the company at paper mill is given by

$$\phi_P(\widetilde{h},\ \widetilde{s},\ \widetilde{\alpha}) = \frac{\eta_{\mathcal{B}}}{4} \sum_{i=1}^{k} \sum_{j \in \widetilde{h}_i} (Q_j - \overline{Q}_j(\alpha_i)) \tag{5.3.7}$$

The overall cost to the company in the planning horizon is given by

$$\phi_W(\widetilde{s}) + \phi_P(\widetilde{h},\ \widetilde{s},\ \widetilde{\alpha}) \tag{5.3.8}$$

Constraints

Note that a split can be uniquely identified with a positive integer k (the number of spells) and an integer vector $\mathbf{h} = [h_1, h_2, \ldots, h_k]$ where h_i is the length of i^{th} spell $\widetilde{h}_i$. In the current problem the number of spells cannot exceed 4 and hence we must have $k \leq 4$. We must have $h_i \geq 16$ and $\sum_{i=1}^{k} h_i = 96$. From the trade-off rules, we must have $\alpha_i \leq s_j$ for any j in spell $\widetilde{h}_i$. The decision variables in the problem are k, h_is, s_js, α_is and u_{ib}s. In order to discourage large deviations, $|x_j - s_j|$s, a gaming constraint is imposed. The gaming constraint is imposed on the overall deviations through an upper

bound on the sum of absolute penalties as follows.

$$\sum_{j=1}^{96} |C_p(x_j - s_j)| \leq \beta, \tag{5.3.9}$$

where β is specified by the DISCOM. At present $\beta = 33750$. There is another constraint that arises from the power plant $\mathcal{B}$. Power plant cannot be operated to produce power below a certain threshold. This in turn limits the power drawn from $\mathcal{G}$ during any spell to a maximum of 22 MW. The complete formulation of the company's decision making problem is presented in (5.3.10) to (5.3.24).

Constraints (5.3.18), (5.3.19) and (5.3.22) ensure that $y_i = 1$ if, and only if, $\alpha_i \geq \alpha_0$. Under the minimization of the objective function subject to constraints (5.3.18) to (5.3.22), the expression $y_i(\alpha_i - 2) + (1 - y_i)F(\alpha_i)$ will be equal to $\overline{Q}_j(\alpha_i)$. Therefore, the objective function in (5.3.10) is same as the one in (5.3.8). Constraints (5.3.16) and (5.3.17) ensure that draws do not exceed the declared supplies at the windmill during the respective time periods. The gaming constraint (5.3.11) can be converted to linear constraints using the standard trick of writing a real number r as difference of two nonnegative numbers ($r = r^+ - r^-$) and its absolute value as the sum of the numbers ($|r| = r^+ + r^-$). Other constraints in the formulation are self explanatory. It can be observed that the objective function is a non-convex quadratic function.

The windmill problem: $P_{\widetilde{x}}(\widetilde{h}, \widetilde{s}, \widetilde{\alpha}, \widetilde{y}, \widetilde{u})$

$$\texttt{Minimize} \quad \frac{\eta_{\mathcal{G}}}{4} \sum_{j=1}^{96} s_j + \sum_{j=1}^{96} C_p(x_j - s_j) + \frac{\eta_{\mathcal{B}}}{4} \sum_{i=1}^{k} \sum_{j \in \widetilde{h}_i} \{Q_j - [y_i(\alpha_i - 2) + (1 - y_i)F(\alpha_i)]\} \tag{5.3.10}$$

$$\texttt{subject to}$$

$$\sum_{j=1}^{96} |C_p(x_j - s_j)| \leq \beta, \tag{5.3.11}$$

$$h_i \geq 16, \ i = 1, 2, \ldots, k, \tag{5.3.12}$$

$$k \leq 4, \tag{5.3.13}$$

$$\sum_{i=1}^{k} h_i = 96, \tag{5.3.14}$$

$$\alpha_i \leq 22 \text{ for all } j \in \widetilde{h}_i,\ i = 1, 2, \ldots, k, \tag{5.3.15}$$

$$\alpha_i \leq s_j \text{ for all } j \in \widetilde{h}_i,\ i = 1, 2, \ldots, k, \tag{5.3.16}$$

$$\sum_{b=1}^{g} \alpha_{0b} u_{ib} \leq s_j \text{ for all } j \in \widetilde{h}_i,\ i = 1, \ldots, k, \tag{5.3.17}$$

$$\alpha_i \geq \alpha_0 y_i \text{ for all } i = 1, 2, \ldots, k, \tag{5.3.18}$$

$$\alpha_i \leq \alpha_0 (1 - y_i) + 22 y_i, \text{ for all } i = 1, 2, \ldots, k, \tag{5.3.19}$$

$$\sum_{b=1}^{g} \alpha_{0b} u_{ib} \leq \alpha_i \text{ for } i = 1, 2, \ldots, k, \tag{5.3.20}$$

$$\sum_{b=1}^{g} u_{ib} = 1 - y_i \text{ for all } i = 1, \ldots, k, \tag{5.3.21}$$

$$u_{ib},\ y_i \in \{0,\ 1\} \text{ for } i = 1, 2, \ldots, k; b = 1, \ldots, g, \tag{5.3.22}$$

$$h_1, h_2, \ldots, h_k \text{ and } k \text{ are nonnegative integers,} \tag{5.3.23}$$

$$\alpha_i \geq 0,\ i = 1, 2, \ldots, k. \tag{5.3.24}$$

Definition 5.3.1. *If $(\widetilde{h}, \widetilde{s}, \widetilde{\alpha}, \widetilde{y}, \widetilde{u})$ is an optimal solution to $P_{\widetilde{x}}(\widetilde{h}, \widetilde{s}, \widetilde{\alpha}, \widetilde{y}, \widetilde{u})$, then call $\widetilde{h}$ as **optimal split**.*

5.4 Analysis

Consider the windmill problem $P_{\widetilde{x}}(\widetilde{h}, \widetilde{s}, \widetilde{\alpha}, \widetilde{y}, \widetilde{u})$. We have partially reduced the complexity of this problem by converting some of the nonlinear expressions to linear ones. But the major challenge of solving the problem stems from the decision variables h_k and k. The problem here is that the decision variable k appears as an index of another decision variable and it also appears as a limit in summation expression. There are no templates or standard models available in the professional OR packages to solve this type of problems. The novelty of our solution approach is in our solution methodology for this problem. Before we present this (in the next section), we observe some interesting properties of the problem. These are presented in the following propositions.

Proposition 5.4.1. *The problem $P_{\widetilde{x}}(\widetilde{h}, \widetilde{s}, \widetilde{\alpha}, \widetilde{y}, \widetilde{u})$ has an optimal solution.*

Proof. The arguments $\widetilde{h}$ and $\widetilde{y}$ are discrete and belong to a finite set, and $\widetilde{\alpha}$ and $\widetilde{s}$ are continuous and belong to a compact set. Thus, the feasible region is a nonempty

compact set. Since the objective function is continuous, the problem has an optimal solution. □

Proposition 5.4.2. *Suppose* $(\widetilde{h}, \widetilde{s}, \widetilde{\alpha}, \widetilde{y}, \widetilde{u})$ *is an optimal solution for the windmill problem. Then for each* i*,*

(i) $s_j = \alpha_i$ *for all* j *such that* $j \in \widetilde{h}_i$ *and* $x_j \leq \alpha_i$*, and*

(ii) $s_j \leq x_j$ *for all* j *such that* $j \in \widetilde{h}_i$ *and* $x_j \geq \alpha_i$*.*

Proof. Let $\widetilde{s}^\star$ be defined by $s_j^\star = \alpha_i$ if $j \in \widetilde{h}_i$ and $x_j \leq \alpha_i$, $s_j^\star = x_j$ if $j \in \widetilde{h}_i$, $x_j \geq \alpha_i$ and $s_j > x_j$; and $s_j^\star = s_j$ otherwise, $i = 1, \ldots, k$. It can be checked that $(\widetilde{h}, \widetilde{s}^\star, \widetilde{\alpha}, \widetilde{y}, \widetilde{u})$ is a feasible solution and its objective value is strictly less than that of $(\widetilde{h}, \widetilde{s}, \widetilde{\alpha}, \widetilde{y}, \widetilde{u})$ if $\{j : s_j^\star \neq s_j\} \neq \emptyset$. Assertions (i) and (ii) follow from this and the optimality of $(\widetilde{h}, \widetilde{s}, \widetilde{\alpha}, \widetilde{y}, \widetilde{u})$. □

Proposition5.4.3 plays crucial role in the construction of our algorithm for solving the windmill problem. For this, we need the concept of ϵ-optimal solutions where ϵ is any positive number for the windmill problem.

Definition 5.4.1. *Consider the windmill problem with optimum objective value* z_0*. Say that a solution to the problem with objective value* z *is an* ϵ*-optimal solution provided*

(i) it satisfies all the constraints except the gaming constraint and the gaming constraint is violated by at most ϵ*, that is,*

$$\sum_{j=1}^{96} |C_p(x_j - s_j)| \leq \beta + \epsilon, \text{ and}$$

(ii) $z < z_0 + \epsilon$*.*

Proposition 5.4.3. *Suppose* $(\widetilde{h}, \widetilde{s}, \widetilde{\alpha}, \widetilde{y}, \widetilde{u})$ *is an optimal solution for the windmill problem. Then there exist* $\widetilde{s}^\star$ *and* $\widetilde{\alpha}^\star$ *such that* $(\widetilde{h}, \widetilde{s}^\star, \widetilde{\alpha}^\star, \widetilde{y}, \widetilde{u})$ *is an* ϵ*-optimal solution to the windmill problem with* $\alpha_i^\star > 0$ *for all* i*.*

Proof. Suppose $\alpha_i = 0$ for some i. Since $\sum_{j \in \widetilde{h}_i} s_j$ and $\sum_{j \in \widetilde{h}_i} C_p(x_j - s_j)$ are continuous functions of s_js, small increase in s_js will result in small changes in these two functions in

a continuous fashion. Let $\bar{\beta}_i = \sum_{j \in \widetilde{h}_i} |C_p(x_j - s_j)|$ which is the sum of gain plus penalty of the spell $\widetilde{h}_i$. From Proposition 5.4.2, we have $\alpha_i \leq s_j \leq x_j$ for all $j \in \widetilde{h}_i$. This implies that all time periods in $\widetilde{h}_i$ result in gain only and not in penalty. Therefore, if we push s_js towards respective x_js, then the gain is non-increasing and penalty is non-decreasing (if $x_j = 0$ then raising s_j will result in increasing penalty; and if $x_j > 0$, then increasing s_j results in reduction in gain). Note that the third component of the objective function involving $\overline{Q}(\alpha_i)$ is a constant (as $\overline{Q}(\alpha) = 0$ for all α sufficiently small) and will remain the same for small changes in s_js of the spell. Therefore, we can increase all zero s_js to a positive level so that the change in the objective value of $(\widetilde{h}, \widetilde{s}, \widetilde{\alpha}, \widetilde{y}, \widetilde{u})$ is less than ϵ and the change in the penalty plus gain of $(\widetilde{h}, \widetilde{s}, \widetilde{\alpha}, \widetilde{y}, \widetilde{u})$ is less than ϵ. It follows that we can increase α_i to a positive level ensuring that the change in objective value and in the sum of penalty plus gain of $(\widetilde{h}, \widetilde{s}, \widetilde{\alpha}, \widetilde{y}, \widetilde{u})$ is at most ϵ. As this procedure can be applied to all spells with zero α_is, the conclusion of the proposition follows. □

The application of the above proposition is that we can solve for ϵ-optimal solutions. This helps us in getting near optimal solutions to the windmill problem. Note that if $(\widetilde{h}, \widetilde{s}, \widetilde{\alpha}, \widetilde{y}, \widetilde{u})$ is an optimal solution and if $\alpha_i = 0$ for some i, then it is possible to get an ϵ-optimal solution even without violating the gaming constraint provided each spell with $\alpha_i = 0$ has at least one positive x_j. If $x_j > 0$, then there is scope for reducing gain which can be used for off setting the increasing penalty arising out of increasing zero-s_js towards positive side. If $\bar{\beta}_i > 0$, then there must be at least one positive x_j in $\widetilde{h}_i$. Also, if there is no spell of length 16 or above in planning horizon with all $\hat{x}_j$s zero, then we can get ϵ-optimal solutions without violating the gaming constraint.

The following proposition is about the stability of the solution. If $(\widetilde{h}, \widetilde{s}, \widetilde{\alpha}, \widetilde{y}, \widetilde{u})$ is an optimal solution for an input vector $\widetilde{x}$, then it is also optimal for the problem with any other input vector $\widetilde{x}^\star$ which is obtained by permuting elements of $\widetilde{x}$ within the spells.

Proposition 5.4.4. *Suppose $(\widetilde{h}, \widetilde{s}, \widetilde{\alpha}, \widetilde{y}, \widetilde{u})$ is an optimal solution for the windmill problem with input $\widetilde{x}$. Suppose $\widetilde{x}^\star$ is such that $\{x_j : j \in \widetilde{h}_i\} = \{x^\star_j : j \in \widetilde{h}_i\}$ for all i. Then, $(\widetilde{h}, \widetilde{s}, \widetilde{\alpha}, \widetilde{y}, \widetilde{u})$ is optimal for the windmill problem with input $\widetilde{x}^\star$.*

Proof. Follows from the fact that all the three terms of the objective function (5.3.10) are invariant under the said permutations. □

5.4.1 The Spell Subproblems $P_{\widetilde{x}_i}(\widetilde{s}_i, \alpha_i, y_i, \widetilde{u}_i, \bar{\beta}_i)$

For a given a spell $\widetilde{h}_i$, let $\widetilde{x}_i$, $\widetilde{s}_i$ and $\widetilde{u}_i$ denote the subvectors of $\widetilde{x}$, $\widetilde{s}$ and $\widetilde{u}$ confined to $\widetilde{h}_i$ respectively. Let $\bar{\beta}_i$ be a nonnegative number (an input to the subproblem). Define the i^{th} spell subproblem $P_{\widetilde{x}_i}(\widetilde{s}_i, \alpha_i, y_i, \widetilde{u}_i, \bar{\beta}_i)$ as:

$$\texttt{Minimize} \quad \frac{\eta_{\mathcal{G}}}{4} \sum_{j \in \widetilde{h}_i} s_j + \sum_{j \in \widetilde{h}_i} C_p(x_j - s_j) + \frac{\eta_{\mathcal{B}}}{4} \sum_{j \in \widetilde{h}_i} \{Q_j - [y_i(\alpha_i - 2) + (1 - y_i)F(\alpha_i)] \tag{5.4.1}$$

$$\texttt{subject to}$$

$$\sum_{j=1}^{96} |C_p(x_j - s_j)| \leq \bar{\beta}_i, \tag{5.4.2}$$

$$\alpha_i \leq 22 \ \texttt{for all} \ j \in \widetilde{h}_i, \tag{5.4.3}$$

$$\alpha_i \leq s_j \ \texttt{for all} \ j \in \widetilde{h}_i, \tag{5.4.4}$$

$$\sum_{b=1}^{g} \alpha_{0b} u_{ib} \leq s_j \ \texttt{for all} \ j \in \widetilde{h}_i, \tag{5.4.5}$$

$$\alpha_i \geq \alpha_0 y_i, \tag{5.4.6}$$

$$\alpha_i \leq \alpha_0(1 - y_i) + 22 y_i, \tag{5.4.7}$$

$$\sum_{b=1}^{g} u_{ib} = 1 - y_i, \tag{5.4.8}$$

$$u_{ib} \in \{0,\ 1\} \ \texttt{for} \ b = 1, \ldots, g, \tag{5.4.9}$$

$$y_i \in \{0,\ 1\},\ \alpha_i \geq 0 \tag{5.4.10}$$

The speciality of the spell subproblem is that the α_i value determines the rest of the solution. Firstly, y_i is fixed depending upon whether $\alpha_i > \alpha_0$ or not. To see that s_js are also determined once α_i is fixed, observe that the properties stated in Proposition 5.4.1 and Proposition 5.4.2 also apply to the spell subproblems. Therefore, for any solution to be optimal for the spell subproblem, we must have $\alpha_i \leq s_j \leq x_j$ for all j such that $x_j \geq \alpha_i$, and for j such that $x_j \leq \alpha_i$, $s_j = \alpha_i$. So, if α_i is fixed, then all s_js that result in penalty are fixed at α_i, and the other s_js that result in gain are adjusted to maximize the gain subject to constraint (5.4.2). Note that increasing gain results in reducing the first two components of the objective function. The third component of the objective

function is anyway the function of α_i only. Therefore, fixing α_i determines the objective value in a unique way. We shall illustrate these ideas with an example. Take $\bar{\beta} = 6000$. Table 5.2 presents a spell of 17 time periods, 1 to 17, with x_js and two feasible solutions. The first solution has $\alpha_1 = 6$ and the corresponding s_j values in the third row (against the α_1 value 6). The second solution has $\alpha_1 = 6.343$ and the corresponding values of s_js are presented in the against 6.343. For $\alpha_1 = 6$, the penalty is 4250 and the gain is 1750. For $\alpha_1 = 6.343$, the penalty is 6000 and the gain is zero. Therefore, in this case, $s_j = x_j$ for all time periods resulting in gain. For the case $\alpha_1 = 6$, penalty is fixed at 4250. Therefore, we adjust the s_7, s_8 and s_9 to make up for the gain 1750. Thus, for this spell subproblem, a feasible solution must have $\alpha_1 \leq 6.343$.

TABLE 5.2: Example of a spell subproblem: s_j values for two different α_1s

TP	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
x_j	5	5	5	5	5	6	7	7	8.00	9	10	10	11	12	13	13	14
6	α_1	α_1	α_1	α_1	α_1	α_1	6	6	7.94	9	10	10	11	12	13	13	14
6.343	α_1	α_1	α_1	α_1	α_1	α_1	7	7	8.00	9	10	10	11	12	13	13	14

Note: TP = time period; $\bar{\beta} = 6000$ for this problem; the first 6 time periods result in penalty for both cases of α_1.

Given a split, the spell subproblems are independent, and they can be solved using the above formulation under two different cases: $y_i = 0$ and $y_i = 1$. Under these special cases, the formulation is a mixed integer linear programming problem. A plausible solution to the main problem may be obtained by concatenating spell subproblem solutions. However, the feasibility/optimality of solutions thus obtained may not hold good for the main problem. The following proposition provides a sufficient condition for a solution to be optimal for the windmill problem.

Proposition 5.4.5. *Suppose $\widetilde{h}$ is an optimal split for the windmill problem and optimal solutions of the spell subproblem with respect to $\widetilde{h}$ satisfy the condition that $\sum_{i=1}^{k} \beta_i \leq \beta$, then the solution obtained by concatenating spell subproblem solutions is an optimal solution for the main problem.*

Proof. Follows from the fact that the objective function of the main problem is sum of the *nonnegative* objective functions of the spell subproblems. If the concatenated solution is not optimal for the main problem, then it will lead to the contradiction that one of the solutions of the spell subproblems is not optimal. □

The following proposition provides a necessary condition for optimal solutions of the windmill problem. Suppose $(\widetilde{h}, \widetilde{s}, \widetilde{\alpha}, \widetilde{y}, \widetilde{u})$ is a feasible solution to the windmill problem. Let $\bar{\beta}_i$ be the sum of penalty plus gain for the i^{th} spell subproblem corresponding to $(\widetilde{h}, \widetilde{s}, \widetilde{\alpha}, \widetilde{y}, \widetilde{u})$.

Proposition 5.4.6. *If $(\widetilde{h}, \widetilde{s}, \widetilde{\alpha}, \widetilde{y}, \widetilde{u})$ is optimal for the windmill problem, then for each i, $(\widetilde{s}_i, \alpha_i, y_i, \widetilde{u}_i)$ is optimal for $P_{\widetilde{x}_i}(\widetilde{s}_i, \alpha_i, y_i, \widetilde{u}_i, \bar{\beta}_i)$.*

Proof. Follows from the fact that the subproblem are independent for a given split. □

5.5 Solving the Windmill Problem

We shall now present our approach to solve the windmill problem. We use a technique that was developed to solve the integrated task and staff scheduling problem (ISTSP) (see Lalita and Murthy (2020a)). We shall briefly recall the model of the problem and the solution approach. Then, we draw a parallel to the windmill problem. We shall restrict the description to a special case of ISTSP in which tasks scheduling is not a part of the problem. In the problem, we are given the number of workers required in each time period of a planning horizon. Staff (workers) work according to shifts. A Shift is a vector of contiguous time periods in the planning horizon. Shifts may vary in their duration (duration specified as the number of time periods). A shift schedule is a shift starting at a specific time period of the planning horizon. For example, consider a shift with a duration of 20 time periods. If this starts at time period 1, a worker who is assigned this shift will be available for work during the time periods 1 to 20. This is one shift schedule. If the same shift starts at time period 31, it will be considered as another shift schedule. A worker who is assigned this shift schedule will be available for work during the time periods 31 to 50. Once the shift schedules are assigned to a set of workers, we can figure out how many workers are available during each time period. In ISTSP, a cost is associated with each shift schedule, and one of the problems there is to assign shift schedules to workers so that the required number of workers are available in each time period with an objective of minimizing the associated cost. The optimization problem here is to determine the optimal shift schedules and the number of workers assigned to each of them.

We adapt the above model to solve the windmill problem. We identify a shift with a spell. The duration of the shift is the length of the spell. In order to reformulate the problem we need to define new variables. Define indicator variables w_{hj}, where w_{hj} is 1 if a new spell of length h is started in time period j. Notice that the variables w_{hj} are defined for $h = 16$ to 96, and j for 1 to 96 only. This is for convenience of the formulation. Define the binary array of variables $w = (w_{hj})$ for h and j in the above specified range. Any split with minimum spell length 16 can be represented by the binary array of variables w satisfying conditions (5.5.1) to (5.5.4), and any w satisfying the conditions yields a split with spells having length at least 16.

$$\sum_{j=1}^{96} \sum_{h=16}^{96} h w_{hj} = 96, \tag{5.5.1}$$

$$\sum_{j=1}^{96} \sum_{h=16}^{96} w_{hj} \leq k, \tag{5.5.2}$$

$$\sum_{j'=1}^{j} \sum_{h=\max(16,\ j-j'+1)}^{96} w_{hj'} = 1, \ \texttt{for}\ j = 1, 2, \ldots, 96 \tag{5.5.3}$$

$$w_{hj} \in \{0,\ 1\}\ \texttt{for}\ h = 16, 17, \ldots, 96,\ j = 1, 2, \ldots, 96. \tag{5.5.4}$$

Constraint (5.5.1) ensures that the sum of the spell lengths is 96, (5.5.2) ensures that there are at most k spells and (5.5.3) ensures that every time period j is covered by exactly one spell (and hence no overlapping). Further, (5.5.1) and (5.5.3) together ensure that spell lengths are at least 16. The spell starting periods are the positive jw_{hj}s and their lengths are the positive hw_{hj}s. Let the spell starting periods in the ascending order, be $j_1(= 1), j_2, \ldots, j_k$. Let the corresponding lengths be denoted by $h_1, h_2, \ldots, h_k$. Next, we define the draw decision variables. Let λ_{hj} be the power to be drawn from $\mathcal{G}$ during a spell of length h that starts in time period j. Let ϵ be any positive number less than α_{02}. Consider the following constraints.

$$\lambda_{hj} \geq \epsilon w_{hj}, \ \texttt{for all}\ h,\ j, \tag{5.5.5}$$

$$\lambda_{hj} \leq 22 w_{hj}, \ \texttt{for all}\ h,\ j, \tag{5.5.6}$$

The above two constraints ensure that $\lambda_{hj} > 0$ if, and only if, $w_{hj} = 1$. From (5.5.3), $\sum_{j'=1}^{j} \sum_{h=\max(16,\ j-j'+1)}^{96} w_{hj'} = w_{h_i j_i}$ for some i, $1 \leq i \leq k$. Let $\lambda_j =$

$\sum_{j'=1}^{j} \sum_{h=\max(16,\ j-j'+1)}^{96} \lambda_{hj'} = \lambda_{h_i j_i}$ for some i, $1 \le i \le k$. From the definition of λ_{hj}, $\lambda_{j'} = \lambda_j$ for all j' that belong to the spell that starts in j [1]. Thus, draw from $\mathcal{G}$ remains unchanged in each spell. For the purpose of modelling, we need to define connection status between $\mathcal{G}$ and $\mathcal{B}$ (y_is) for each time period. For this, we replace the constraints (5.3.18) and (5.3.19) with $\lambda_j \ge \alpha_0 \bar{y}_j$ and $\lambda_j \le \alpha_0(1-\bar{y}_j) + 22\bar{y}_j$, where $\bar{y}_j$s are the indicator variables which are 1 if $\mathcal{G}$ and $\mathcal{B}$ are connected in the time period j. The windmill problem is reformulated in (5.5.7) to (5.5.20).

The main feature of the revised formulation is that it does not involve the spell variables h_is, $\widetilde{h}_i$s and the number of spells variable k. Constraint (5.5.17) insists that $\lambda_{hj} > 0$. This adds additional constraints to the original problem. We solve the problem by choosing ϵ positive and less than α_{02}. In the light of Proposition 5.4.3, any optimal solution to the revised formulation is an ϵ-optimal for the original problem. Note that the we should have used the term $(1-\bar{y}_j)F(\lambda_j)$ in the objective function, but $1-\bar{y}_j$ is redundant here. This is because when $\bar{y}_j = 1$, $F(\lambda_j) = 0$ as all u_{jb}s are equal to zero under constraint (5.5.13). As λ_j remains unchanged during any spell, constraints (5.5.11) and (5.5.12) force $\bar{y}_j$s to remain unchanged within spells.

The windmill problem (revised formulation): $P_{\widetilde{x}}(\widetilde{h}, \widetilde{s}, \widetilde{\alpha}, \widetilde{y}, \widetilde{u})$

$$\texttt{Minimize} \quad \frac{\eta_{\mathcal{G}}}{4}\sum_{j=1}^{96} s_j + \sum_{j=1}^{96} C_p(x_j - s_j) + \frac{\eta_{\mathcal{B}}}{4}\sum_{j=1}^{96}\{Q_j - [\bar{y}_j(\lambda_j - 2) + F(\lambda_j)]\} \tag{5.5.7}$$

$$\texttt{subject to}$$

$$\sum_{j=1}^{96} |C_p(x_j - s_j)| \le \beta, \tag{5.5.8}$$

$$\lambda_j \le s_j, \ \texttt{for}\ j \in T, \tag{5.5.9}$$

$$\sum_{b=1}^{g} \alpha_{0b} u_{jb} \le \lambda_j, \ \texttt{for}\ j \in T, \tag{5.5.10}$$

$$\lambda_j \ge \alpha_0 \bar{y}_j \ \texttt{for}\ j \in T, \tag{5.5.11}$$

[1] Suppose $j^\star$ is in the spell of length h that starts in j. This means $j^\star \ge j$. We have $\lambda_{j^\star} = \sum_{j'=1}^{j^\star} \sum_{h'=\max(16,\ j^\star - j'+1)}^{96} \lambda_{h'j'} \ge \lambda_{hj}$ (λ_{hj} is one of the terms in summation). From the nonoverlapping constraint, equality follows.

$$\lambda_j \leq \alpha_0(1-\bar{y}_j) + 22\bar{y}_j, \quad \texttt{for } j \in T, \tag{5.5.12}$$

$$\sum_{b=1}^{g} u_{jb} = 1 - \bar{y}_j, \quad \texttt{for } j \in T, \tag{5.5.13}$$

$$\sum_{j=1}^{96}\sum_{h=16}^{96} h w_{hj} = 96, \tag{5.5.14}$$

$$\sum_{j=1}^{96}\sum_{h=16}^{96} w_{hj} \leq 4, \tag{5.5.15}$$

$$\sum_{j'=1}^{j} \sum_{h=\max(16,\ j-j'+1)}^{96} w_{hj'} = 1, \quad \texttt{for } j \in T, \tag{5.5.16}$$

$$\lambda_{hj} \geq \epsilon w_{hj}, \quad \texttt{for all } h,\ j, \tag{5.5.17}$$

$$\lambda_{hj} \leq 22 w_{hj}, \quad \texttt{for all } h,\ j. \tag{5.5.18}$$

$$\lambda_j = \sum_{j'=1}^{j} \sum_{h=\max(16,\ j-j'+1)}^{96} \lambda_{hj'} \tag{5.5.19}$$

$$u_{jb},\ \bar{y}_j,\ w_{hj} \in \{0,\ 1\},\ \lambda_{hj} \geq 0. \tag{5.5.20}$$

5.6 Application

We first describe the details of the units (U_is) at the paper mill and then present the data for the windmill problem. There are 13 units at the paper mill. These units and their demands are presented in Table 5.3. The units require power throughout the planning horizon at the rates of demands specified in the table. Units U_9 to U_{13} are static units and the remaining ones are dynamic units. The number of switches between $\mathcal{G}$ and $\mathcal{B}$ is limited to at most 3 which means that the maximum number of spells is 4. The threshold $\alpha_0 = 14.3$ (equals the sum of demands of dynamic units plus 2 MW), and the distinct possibilities of $\overline{Q}_j(\alpha)$ (see (5.3.3)) is 228 (with $[\alpha_{0,1},\ \alpha_{0,2}, \ldots, \alpha_{0,228}] = [0, 0.19, 0.25, 0.36, 0.44, 0.47, \ldots, 12.3]$). The other inputs to the problem are the forecasts $\hat{x}_j$s. Past data on forecasts and actual supplies are available for several days. In this chapter, we are concerned with the forecasts. The problem is solved for 20 instances (each instance is for a day) and they are from the first 20 days of October. Figure 5.2 depicts the variation in the forecasts for five selected instances. The results are summarized in Table 5.4. First four columns of this table present the summary statistics of the

TABLE 5.3: Demand Load of Units at the Paper Mill

Unit	Code	Demand (MW)
New Fiber Line	U_1	3.74
Paper Machine 1	U_2	3.42
Paper Machines 2 and 3	U_3	3.32
Old Chipper	U_4	0.19
IRU L-B	U_5	0.25
DDR	U_6	0.55
New Chipper	U_7	0.47
Colony	U_8	0.36
Paper Machine 4	U_9	3.50
Paper Machine 5	U_{10}	4.00
Paper Machine 7	U_{11}	4.00
Paper Machine 8	U_{12}	4.31
Pulp Mill	U_{13}	7.00
	Total	35.11

forecasts. The mean stands for the average of the 96 observations of an instance. From the table, it is seen that there are days with very low forecasts (instance number 13 onwards). To see how the model is deciding, we observe the metric $\sum s_j / \sum \hat{x}_j$. When this figure is above 1, we may interpret it as the model taking advantage by declaring s_j values on the higher side (on an average basis) at the windmill. For the instance 16, this ratio is 3.5. Though the forecasts vary from 0 to 1.3 MW for this instance, the model declares 3.5 times higher compared to the average forecast of 0.2 MW. In all the 20 cases, the solver had to be aborted without reaching the optimal solution. The 'optimality' is computed as the ratio of best objective value obtained to the lower bound (as computed by the solver). To obtain

at least 90% optimal solution, it is taking about 8 minutes on an average. The number of spells for the terminal solutions is 4 for all the 20 instances. For the instance 13, the spell α_i values are equal to 0.2 MW for the solution at the termination. The company can treat all the four spells as one single spell with common $\alpha_i = 0.2$ MW and take the advantage (of connecting only the Old Chipper to $\mathcal{G}$ directly throughout the day

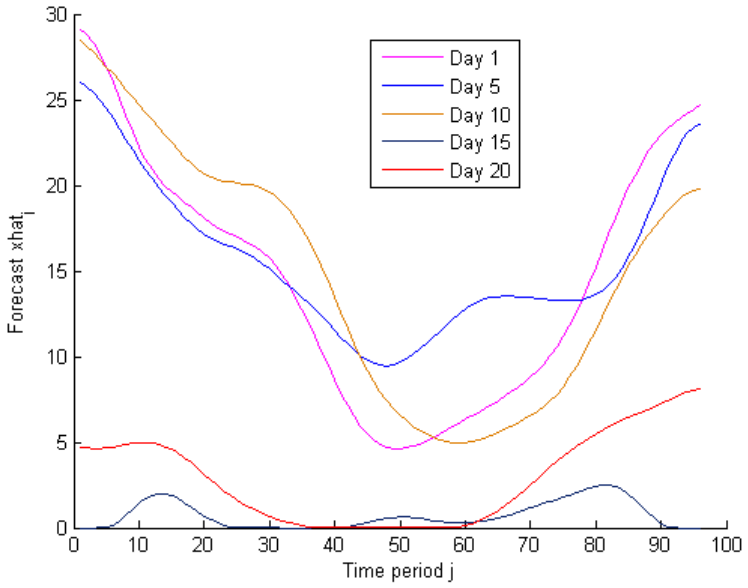


FIGURE 5.2: Forecasts for selected five days

without disturbing any connections). Next, look at α_i values of instance 15. Though it has only two distinct α_is, one can club only spells 2 and 3 to make it a 3-spell solution.

A complete solution to instance 1 is presented graphically in Figure 5.3. The spells are 1-17, 18-40, 41-80 and 81-96. As $\alpha_3 = 4.7$, $\mathcal{G}$ is connected directly to units Paper machines 2 and 3, DDR, New Chipper and Colony for 10 hours. The horizontal lines in the figure (in orange color) correspond to the α_is. The blue line represents the s_js and the grey one represents the $\hat{x}_j$s. Out of 96 time periods, s_js differ from the corresponding x_js in 18 time periods. Out of these 18, two time periods correspond to gain, namely, time periods 42 and 63. From the graph it can be observed that $s_j < \hat{x}_j$ for $j = 42, 63$.

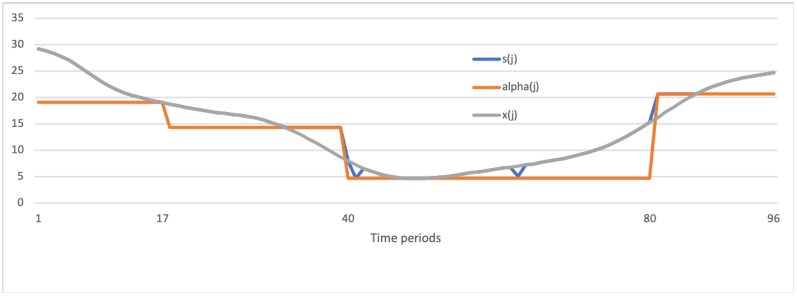


FIGURE 5.3: Data and solution to an instance of the windmill problem.

TABLE 5.4: Results from solving 20 real-life instances of the windmill problem

	Summary statistics of $\hat{x}_j$s						Paper mill draws					Total		Time	
Instance	Mean	sigma	Min	Max	k	h_is	α_1	α_2	α_3	α_4	Gain	Cost	Optimality	(seconds)	$\frac{\sum s_j}{\sum \hat{x}_j}$
1	14.5	7.2	4.6	29.1	4	17,23,40,16	19.0	14.3	4.7	20.6	11	2267	90	762	1.0
2	16.7	5.7	10.7	27.4	4	23,24,29,20	20.6	12.3	11.2	12.3	61	2112	91	676	1.0
3	10.5	5.4	3.4	22.1	4	24,22,31,19	12.3	8.3	3.3	14.3	42	2391	90	1080	1.0
4	10.3	5.9	4.9	23.6	4	16,47,16,17	12.3	5.6	6.2	15.0	46	2338	92	844	1.0
5	15.7	4.4	9.5	26	4	18,15,45,18	19.9	14.4	12.3	13.4	0	2133	93	476	1.0
6	13.7	5.8	6.8	25.8	4	16,19,41,20	19.3	12.3	6.8	12.3	61	2214	91	354	1.0
7	9.5	6.6	3	24.2	4	29,35,16,16	12.3	2.3	4.0	6.9	57	2506	90	1040	1.0
8	17	6.2	10.6	31	4	43,21,16,16	14.3	10.7	12.3	22.0	54	2135	94	795	1.0
9	23.3	4.7	16.8	31.5	4	37,19,23,17	21.4	16.9	18.4	22.0	98	1923	93	117	1.0
10	14.8	7.2	5	28.5	4	27,17,33,19	20.0	12.3	5.2	12.3	74	2180	90	401	1.0
11	8.7	7.5	0.8	24.2	4	16,17,47,16	18.9	12.3	0.8	5.1	58	2444	92	802	1.0
12	2.4	3.2	0	11.1	4	16,45,19,16	7.8	0.2	0.8	0.1	33	2613	92	540	1.1
13	0.8	0.4	0.2	1.7	4	16,16,44,20	0.2	0.2	0.2	0.2	100	2600	96	71	0.5
14	1.1	0.7	0	2.4	4	27,31,20,18	0.1	0.9	1.5	0.2	5	2667	93	56	1.3
15	0.7	0.8	0	2.5	4	16,17,31,32	0.3	0.1	0.1	0.3	77	2626	96	124	0.7
16	0.2	0.4	0	1.3	4	17,21,17,41	0.4	0.4	0.3	0.4	2	2665	97	331	3.5
17	0.5	0.8	0	2.5	4	17,16,38,25	0.1	0.2	0.1	0.2	13	2680	95	138	1.6
18	1.5	2	0	6.1	4	20,21,24,31	0.2	0.2	0.2	0.9	1	2698	90	238	1.3
19	2.4	2.2	0	6.7	4	30,30,20,16	1.8	0.1	0.2	5.2	96	2562	94	217	0.8
20	2.9	2.6	0	8.1	4	16,16,41,23	4.5	0.3	0.1	3.9	95	2550	93	312	0.9

Notes: The solver had to be terminated in all the 20 instances. The 'optimality' column presents the ratio of best objective to the lower bound at termination as percentage. The average of these percentages is 93, and the average of times is 469 seconds (close to 8 minutes). Penalty plus gain is 33750 in all the instances. 'Gain' column shows the gain percentage against 33750. Cost is shown in 1000s.

5.7 Summary

In this chapter, we have addressed a decision making problem for a large paper industry that trades its windmill power with a distribution company under a special incentive scheme of exchanging power promoted by the government. The problem involves declaring schedules for power supply at the windmill on a one-day planning horizon basis to take advantage of drawing power from the grid at the paper mill location. The decisions result in costs at windmill and at paper mill. The problem is a difficult combinatorial optimization problem involving intricate relationships among decision variables with a quadratic cost objective function. We derive several interesting properties of the problem and the solutions. Further, we show that the solutions obtained by method are ϵ-optimal. To the best of our knowledge, this is a new type of problem encountered in windmill power trade-off optimization problems. Solving the problem needed a new solution approach. This chapter provides a novel solution methodology. Both formulation and solution methodology are innovative. The methodology uses a technique developed recently for solving staff and task scheduling problems. A number of real-life instances of the problem are solved and the results are presented. The novelty of the chapter is in adapting a technique that was developed for a completely different type of problem. We believe that the technique will be useful for many other types of job scheduling optimization problems involving multiple tasks which can be processed on multiple processors.

Chapter 6

Conclusions and Future Work

This chapter presents a summary of the contributions of this thesis with some supplementary results and possible problems for future research. The contributions are induced by our attempts to address real-life problems from industry. The inherent characteristic of these problems is that they are scheduling problems with renewable resources. Either these problems are too large in size to handle with the existing solution approaches or need approaches that are computationally faster. We have adopted mathematical programming formulations and developed strategic modelling techniques to provide new solutions to these problems. The research problems and our contributions for these problems are summarized in the following subsections.

6.1 The Integrated Staff and Task Scheduling Problem

Staff and task scheduling problems, stand alone or integrated, are hard problems to solve. There are many applications where flexible shifts (shifts of different durations) are indispensable. Another dimension that adds to the flexibility is the requirement of relief breaks within shifts. Shift's flexibility increases the problem size dramatically in staff scheduling. Due to the complexities involved, introduction of breaks has received cursory attention in the literature on staff scheduling problems. Considering one lunch break in shifts, Brunner and Stolletz (2014) introduced a branch and price algorithm for

a staff scheduling problem at airport check-in counters. To overcome the slow convergence of the column generation, they proposed a stabilization technique using a dynamic parameter updating procedure. With all this, it still takes several hours, as reported in the paper, to produce high quality feasible solutions. The integrated staff and task scheduling problem (ISTSP) adds yet another dimension to the staff scheduling problem, namely, scheduling of tasks. In ISTSP, the man power requirements of each task are specified along with their durations, and are allowed to start within respective time windows. The choice of task start times will dictate the staff requirements. Further, the tasks may have precedence relationships among themselves. Therefore, the problem is much more complex than the staff scheduling problem. The latest contribution to ISTSP is by Volland et al. (2017b). Proposing a master integer programming (MIP) formulation, the authors developed an iterative procedure involving the master problem and two subproblems, the staff scheduling subproblem and the task scheduling subproblem. With a lower bound for MIP obtained through the iterative procedure, the authors solve the MIP imposing the lower bound constraint. As reported in Volland et al. (2017b), there are only few publications on ISTSP. Beliën and Demeulemeester (2008) and Maenhout and Vanhoucke (2016) are the two other articles on ISTSP. Models in these two articles do not consider breaks in shifts. While Volland et al. (2017b) discusses incorporation of breaks into shifts, the formulation presented and the numerical experiments studied in the article do not incorporate breaks.

In this thesis, we considered both staff scheduling problem and ISTSP with most versatile features. The features include (i) variable shift durations, (ii) multiple relief breaks with minimum time gap restrictions between successive breaks within shift, (iii) minimum time gap between successive shifts, (iv) consecutive days-off constraints, and (v) maximum number of hours per worker in the planning horizon.

We solved the problems using a decomposition principle. According to this, we split the problem into two stages and in each stage we solve an assignment problem. Introducing the concept of shift patterns and shift schedules, the first stage solution determines the shift schedules (a shift schedule is a shift pattern that starts at a particular time period in the planning horizon). If the problem is ISTSP, then the first stage determines shift schedules along with task starting times. The objective function for the first stage is the cost of each shift schedule. Taking costs as 1, the objective function reduces to

number of shift schedules to meet all the demands. In the second stage, the shift schedules determined in first stage are assigned to workers satisfying work assignment rules. The objective of the second stage is to make the assignments with minimum number of workers. The formulation of the second stage assignment is a crucial step of this approach. If the costs in the objective function of the original problem depend only on the shift schedules, then our two stage approach will actually produce an optimal solution. If the objective is to minimize the number of workers, then the solution is near optimal. To prove optimality or near optimality, we provide a useful and very effective lower bound for the number of workers as a function of the lower bound for the first stage objective value.

Another important contribution of our work is the development of the split technique. With this technique, we are able to solve large problems in approximately the same time that small problems take. We believe that this is a significant contribution to staff scheduling and ISTSP.

The efficacy of our methodology is evaluated through a number of numerical experiments. The test instances are created for staff scheduling as well as for ISTSP. The test cases for ISTSP are specially designed so that the results can be compared to those of Volland et al. (2017b). Also, we tested with instances with very huge demands compared those reported in the published papers. Results are compared with respect to optimality gap and processing time. With respect to optimality gap, out of 41 instances tested, 23 are optimal, 6 are at least 99% optimal, 37 are at least 95% optimal. Of the 41 instances, 19 are under ISTSP. Of these 18, 19 are 100% optimal. With respect to time, we find huge reductions compared to the existing results.

Our split technique to solve large scale problems in approximately same time that is required for problems with small demands is possibly a lead for a research problem on the complexity analysis. In the existing literature, solution methods are assessed at different demand sizes such as small, medium and large. Through the split technique introduced in this chapter, we are able to handle problems with large demands efficiently. This raises a question that whether demand size has any influence on the complexity of the problem. This point needs to be explored theoretically. Another direction for future research is extending the methods introduced in this chapter to multi-skill personnel

staff scheduling problems.

6.2 Check-in Counters Problem

We encountered this problem at a large airport in India. The departures from the airport are tactically planned based on two seasons. The airport operator receives applications from different airlines. Each airline submits an application for a number of departures each season. The nature of departures in a season is a weekly schedule rostered over the season. Due to limited resources, all requested departures may not be accommodated. In the problem addressed to us, the resource is the limited number of check-in counters in the airport. The research questions addressed are:

Problem A What is the number of counters required for a departure or a group of departures under variable counter assignment under the standard rational assumptions of seat capacities, service norms and first-in-first-out queue discipline?

Problem B Determine a check-in counter assignment for a given set of departures under adjacent variable counter allocation with minimum number of counters?

Problem A

Problem A has a direct impact on the customer waiting times. The initial solutions provided in the literature addressed the problem using queueing models and simulation approaches. The first exact formulation for this problem was proposed in Bruno and Genovese (2010). This was later enhanced by Araujo and Repolho (2015) to control number of passengers waiting in the queue. There was no mathematical model in the literature to control waiting time directly or to implement the first-in-first-out queue discipline. We are the first to propose an integer linear programming formulation to control the waiting time and implement the first-in-first-out queue discipline. The *uv*-model proposed for this purpose has been compared with the model proposed by Araujo and Repolho (2015) and the airport's solution. A theoretical comparison of these models is as follows. In our *uv*-formulation, the maximum waiting time of any passenger is equal to the duration of two time periods. That is, if the planning horizon is divided into 15-minute time periods, then the maximum waiting time under *uv*-model is 30 minutes. If

one wants to ensure, theoretically, that the maximum waiting time is 20 minutes (which is the standard norm according to IATA), then we can model the problem using planning horizon with 10-minute time periods. Thus, the *uv*-model controls waiting time directly. The model by Araujo and Repolho (2015) also controls waiting time but implicitly. In their model, they impose a constraint that a certain proportion of the passengers present at the beginning of any time period is cleared (served) during that period. This model clearly does not control the waiting time directly. The solution by the airport operator is not backed by any mathematical model. Through the following counterexample, it has been verified that the airport solution is not even feasible. When the three models were applied to a particular instance of the problem to minimize the number of counters, the optimum objective value was 57 whereas the airport solution reported 42 counters. The performance of the models were compared using a number of simulated instances. The *uv*-model used smaller number of counters compared to the other two models (the airport solution with 42 counters is infeasible and hence not comparable). However, with regard to waiting time of passengers, the Araujo and Repolho (2015) model had a marginal edge over *uv*-model but this is at the cost of using higher number of counters.

In the *uv*-formulation, passengers arriving in a time period i are served either in time period i or $i+1$. This can be generalized to u^k-*formulation* by distributing the passengers over k time periods. That is, serve the passengers arriving in time period i in the time periods $i,\ i+1,\ldots,p$, where $p = \min(i+k-1, q)$ and q is the last check-in time period for the departure. In the new notation, *uv*-formulation is u^2-formulation. The drawback with u^k-formulation for $k > 2$ is that it looses the ability to maintain the first-in-first-out queue discipline. But for this, the model enhances its capability to better optimization and better planning. Application of u^k-formulation can be envisaged in priority queues. For example, consider a case where the first class passengers should be served within one time period, the executive class passengers within two time periods and the rest in at most three time periods. In such cases, we may use u^3-formulation. One should be cautious when applying u^k models for the case of check-in counters where common counters are being assigned to a group of departures. We intend to explore the properties of u^k model and the areas where it will be useful.

Problem B

Problem B comes under the adjacent resource scheduling problems. This fact was

pointed out by Duin and Sluis (2006) and its complexity was analysed. Dijk and Sluis (2006) proposed integer linear programming formulations for both dynamic and variable counter allocation. This formulation was the one that was used in all subsequent publications on the problem. Dijk and Sluis (2006) observed that the formulation was suitable for small scale problems and reported that it took more than 30 minutes on an average for instances with 48 flights and 24 periods. To address large size problems, we came up with a new integer linear programming formulation by introducing assignment structures that are pragmatic and desirable. The new formulation is capable of solving large scale problems. A problem instance with 783 departures over 229 time periods was solved to near optimality (at least 90% optimal) in two minutes. The size (number of variables plus constraints) of our formulation is much smaller compared to that of Dijk and Sluis (2006). For the instance just mentioned, the size of our formulation (number of variables and constraints) is 82566 whereas it is 191062 for the Dijk and Sluis (2006) formulation. The main problem meant for seasonal planning involved one week planning horizon with 687 time periods of 15-minutes duration. An instance of this problem had 2539 departures. To solve problems of large size like this instance, we developed two special strategies. The first strategy is to fix the counters for few airlines' departures with common counter requirement that contribute to a major chunk of all departures, and then use the remaining counter time to accommodate other departures. With this we could solve the instance with 2539 departures over 687 time periods to near optimality (at least 95% optimal) in 42 minutes. To assess the optimality percentages mentioned above, we used a crude lower bound which is less likely to be attained by an optimal solution. The second strategy is to divide the planning horizon into two or more parts and then solve the problem of each part with respective departures, and finally combine the solutions of the parts to derive a solution for the original problem. For this large instace, the size of our formulation is 540,416 where as the size of the formulation by Dijk and Sluis (2006) is 1,780,000.

There is yet another strategy that one may explore to handle large size problems. To describe this we need to first look at the derivation of the crude lower bound for the minimum number of counters mentioned in the previous paragraph. For Problem B, the number of counters required for each departure is taken as input (this number is obtained from the first stage optimization). From this, we can find, for each time period

t, the sum of number of counters of all departures that require counters in time period t. Suppose this number is $n(t)$. Then, the lower bound mentioned above is taken as $\max_t n(t)$. This bound depends on which departures are under consideration. For the instance with 2539 departures, $\max_t n(t) = 171$. Now suppose we have 171 counters in the airport. Then, divide the 2539 departures into three groups so that for the first group $\max_t n(t)$ is less than or equal to 50, for the second $\max_t n(t) \leq 60$ and for the third, $\max_t n(t) \leq 61$. Now solve the three problems separately and combine the solution. Note that dividing the departures into three groups is itself an optimization problem which can be formulated as another integer linear programming problem.

An important and useful supplementary contribution of this work is our Excel program that draws the time-counter space diagram for the solution of Problem B on Excel spread sheet. With the help of excel tools to magnify and compress the spread sheet, the user can carry out the sensitivity analysis of the solution. For instance, the user can find out where new departures can be accommodated, how to modify the assignment manually using cut and paste tools, when to schedule the maintenance activities of check-in counters, etc. A detailed description of this is included in section 3.7. We also compare our model for adjacent resource allocation with the formulation by Dijk and Sluis (2006) and theoretically demonstrate that significant gains in performance result from our model (see Theorem 3.7.1 in section 3.7).

6.3 The Berth and Crane Assignment (Specific) Problem

Berth allocation problems integrated with crane assignments are known to be NP-hard problems. Exact methods for these problems involve huge number of variables and constraints which arise from the non-overlapping of ships constraints, positioning of ships in the time-berth space diagram and assignment of non-crossing cranes to ships. The case of continuous berth allocation is even worse as it requires assignment of contiguous berth sections to each ship. For instance, the problem of one-week planning horizon with 168 one-hour time periods, 7 cranes and 30 ships, the DRPF formulation of Agra and Oliveira (2018) has 44,138 variables and 15,691,192 constraints. These formulations are not suitable for solving the problems on commercial solvers. Therefore, one tries to use

methods such as column generation (Wang et al. (2018)). Similarly, Agra and Oliveira (2018) proposed an enhanced formulation and a heuristic to generate upper bound so that the enhanced formulation with the upper bound produces solutions faster. Thus, finding optimal solutions to these problems is difficult. Another issue with the optimal solutions is that some of them may be undesirable from an operational point of view. For instance, frequent shifting of cranes from one ship to another will be operationally inconvenient and time consuming. For this reason, some models impose the invariant crane restriction which does not allow any crane serving a ship to serve another ship until the service of the ship it is serving is completed (see Imai et al. (2008) and Türkoğulları et al. (2014)). In the survey article Bierwirth and Meisel (2015), the authors report that exact methods are applied in only one fourth of the approaches, ranging from MILP formulations combined with standard solvers to highly sophisticated branching based algorithms. Against this backdrop, we looked for solutions with simple formulations which can be effectively solved using commercial solvers.

In this thesis, we considered the continuous berth layout and dynamic arrivals and introduced a new class of solutions, the BCI (berth and crane invariant) solutions, for the berth allocation and crane assignment (specific) problem (BACASP). We proposed a formulation for finding an exact BCI solution. BACASP model was introduced in Türkoğulları et al. (2014). Agra and Oliveira (2018) proposed an enhanced integer linear programming formulation for an exact optimal solution for BACASP. From the operational point of view, the BCI solutions are very convenient. Further, the formulation sizes are very small and the commercial solvers can handle these formulations with ease. As a comparison, for the problem instance mentioned in the previous paragraph with 30 ships over 168 time periods and 7 cranes, the number of variables and constraints in our formulation for BCI solution are 3426 and 26146 respectively. From the numerical experiments, we found that the BCI solutions are satisfactory in terms of being close to optimum objective values. Besides, one can use the optimum objective values of BCI solutions as useful upper bounds in the formulations for exact optimal solutions as it was done in Agra and Oliveira (2018) using rolling horizon heuristic. When the number of ships become too large, finding even a BCI solution gets tougher. To address such instances, we have proposed decomposition techniques. Finally, we propose a formulation to expand the BCI class of solutions to a larger class of solutions by breaking invariance

restriction for the entire planning horizon. In this case, the planning horizon is divided into two or three equal (approximately) parts and we insist invariance of berths and cranes in each part.

We have not performed a formal analysis on the extent of optimality gap between a BCI solution and a global optimal solution. This needs proper basis for comparison. That is, one has take into account the benefits derived from a BCI solution such as time savings rendered by avoiding frequent shifting of cranes, to compare with a global optimal solution. This is a possible opportunity for future research in this direction.

6.4 Windmill problem

In this work we developed a solution procedure for a complex scheduling problem under a power exchange scheme. This is an important problem as it promotes green power generation. A paper mill that also owns a windmill, exchanges power produced at the windmill with the power drawn from grid at the paper mill. Under the exchange scheme, a supply schedule is to be declared at the windmill and consumption schedule is to be planned at the paper mill. The problem is to determine the two schedules based on a planning horizon of a day divided into ninety-six 15-minute time periods. According to the structure of the scheme, the shortage/excess from the schedule in each time period attracts penalty/reward according to a piece-wise linear function. Further, the scheme stipulates that in each time period, the power drawn from grid at the paper mill cannot exceed the corresponding supply declared in the supply schedule at the windmill. The company uses the grid power as complimentary to its own self generated power. The company's self generated power is relatively more expensive compared to the grid power. The scheduling decisions result in a cost to the company which is nonlinear function due to the nature of restrictions on using the power from the grid. Due to the restriction on the connectivity issues at the paper mill, the power draw from the grid has to be a piece-wise linear function with at most four pieces and in each piece the function has to be nonnegative constant (that is, the power draw in each piece should be at a constant rate). To prevent the company from declaring greedy supply schedules at the windmill, the exchange scheme imposes a ceiling on the overall penalty/reward resulting from the

schedule. This is known as gaming constraint, and according to this, the total penalty plus gain cannot exceed the specified ceiling.

We developed a mathematical formulation for the problem. The formulation involves real and integer variables with a quadratic objective function and nonlinear constraints. One of the challenges of this problem is determining the piece lengths of the piece-wise linear function for the power draw at the paper mill. The formulation involves decision variables that are functions of other decision variables. As a result the mathematical model, in its original form is not suitable for solving the problem with standard models using any of the commercial solvers. We study various properties of the problem. The main contribution of this work is our formulation and the development of a solution procedure to solve it. We introduce a parameter ϵ, an arbitrary small positive number, and with that we reformulate the problem as a mixed integer program with linear constraints. We prove that by choosing ϵ sufficiently small, the solution of the formulation can be made arbitrarily close to the optimal solution to the windmill problem. We believe that for ϵ sufficiently small, the formulation actually yields an optimal solution to the windmill problem. It will be interesting to see if this is true. The computational complexity for each of the problems discussed in this thesis is a possible direction for future research.

Appendix A

The check-in counter allocation problem: A Detailed literature survey

A.1 Introduction

Air-travel has become the most preferred mode of travel. The total number of air travellers was 4.1 billion in 2017, which is 7.2 per cent higher than the year 2016, while the number of flight departures reached 36.7 million in 2017, a 3.1 per cent increase compared to 2016 (International Civil Aviation Organization (2017)). The increase in economic growth and the average household income has contributed to the growth of the aviation industry. The present trends in air transport suggest that passenger numbers could double to 8.2 billion in 2037 (IATA (2018)). The ever increasing demand for airport resources has made careful planning and optimal use of resources a necessity. Delay due to inadequacy or inefficient management of airport facilities may result in penalty for the airline companies (Hsu and Chao (2005)). It has been found that 80% of the passenger delay at airports is due to waiting for check-in (Takakuwa and Oyama (2003), Parlar et al. (2013)). To overcome this problem, kiosks were introduced in airports. Abdelaziz et al. (2010) study outcomes of introducing kiosks at Cairo airport. Passengers with baggage were asked to use luggage-drop counters after check-in at kiosks. This reduced

waiting time at kiosks for check-in, but resulted in waiting at luggage drop areas. Though airlines rely on kiosks for management of queue and congestion at airports, their use mainly depends on the convenience for check-in as well as luggage drop. With many passengers checking into their flights online, luggage drop areas are also expected to become congested.

Therefore, efficient usage of check-in counters will improve service levels at airports, result in smaller queue lengths for airport operators and faster check-ins for passengers. As most of the passenger waiting time is spent in the check-in area at an airport, a reduction impacts public perception of the level of service and indirectly enhances the airport revenue due to the increased stress free time in the commercial areas of the airport (Hsu and Chao (2005), Lin and Chen (2013), Parlar et al. (2013)).

To achieve all the above objectives in counter allocation, different optimization techniques have been used in the literature. This supplementary material reviews all the methods proposed for counter allocation. The techniques discussed in here for allocating adjacent counters can be applied to other adjacent resource allocation problems, such as: warehouse space optimization where warehouse area has to be assigned to customers for storage for a certain time, berthing problem at ship yards, where berthing space has to be allocated to ships given the ship size and time for which it docks at the shipyard (see Bierwirth and Meisel (2010)), for assignment of computer hard disk memory (if contiguous allocation of memory is required (see Duin and Sluis (2006))) and other resource scheduling problems where jobs cannot be shifted in time but resource requirements may be satisfied by any set of adjacent resources and may vary with time.

Efforts at reviewing resource usage at airport terminals have not explicitly focused on check-in counter utilization. Cheng et al. (2012) and Wu and Mengersen (2013) have reviewed models presented in literature for resource scheduling at airport terminals. Cheng et al. (2012) reviews theory of allocating and scheduling resources by grouping existing literature into three parts, viz., methods of integrating airport operation data, methods of predicting passenger flow at airport terminals, and optimization of allocating and scheduling passenger service resources at airport terminals. Wu and Mengersen (2013) review airport passenger terminal models by classifying them based on usage scenarios.

Wu and Mengersen (2013) review work related to capacity planning, operational planning and design, security policy and airport performance. Cheng et al. (2012) and Wu and Mengersen (2013) do not study the check-in counter allocation problem in airport terminals. This supplementary material reviews published literature related to planning and modelling the counter allocation problem for flight departures at an airport, it does not consider studies concerning with real-time management of sudden/unforeseen crisis/circumstances at the airport.

This appendix has four sections. Section A.2 describes the check-in counter allocation problem. Section A.6 classifies different methods in the literature by the problem type addressed and its limitations. Section A.7 discusses different approaches for modelling the check-in counter allocation problem.

A.2 The Check-In Counter Allocation Problem

Airports are divided into airside and landside areas. Airside operations consist of scheduling flights on the runways and other related on-air flight operations. Landside operations consist of scheduling and organization of the processes that passengers need to undergo before boarding a flight. It is imperative that airport landside operations focus on timely boarding of passengers and flight departures. For ease of operation, an airport has designated areas for different pre-boarding operations for passengers, such as check-in, security check, immigration, etc. In most airports, the check-in area consists of multiple structures of counters arranged around a conveyor belt in a u-shape. This structure is referred to as an island. Many such islands make up the check-in area (see Fig: A.1).

Though all the counters are not physically adjacent to one another, this is supposed to be the case for simplicity in modelling the problem. Moreover, counters can be rearranged according to physical layout at later stages for implementation of the solution. In most of the literature discussed here, a two dimensional space is used as a representation of the counters in the planning horizon (see Fig.A.2). The planning horizon is the time period for which counter assignments are computed. To model the problem, the planning horizon is divided into smaller Time Intervals (also Time Windows(TWs)). The

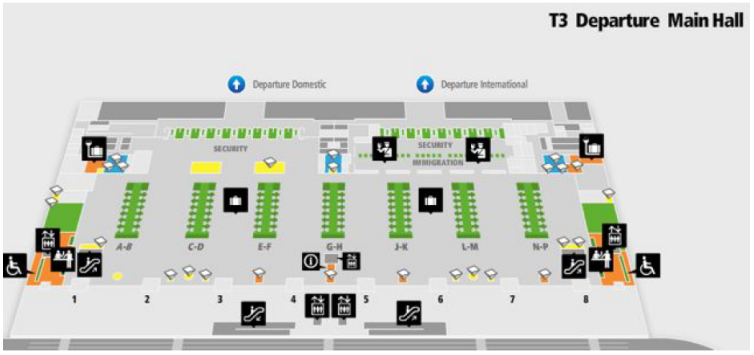


FIGURE A.1: Departure Hall in an Airport

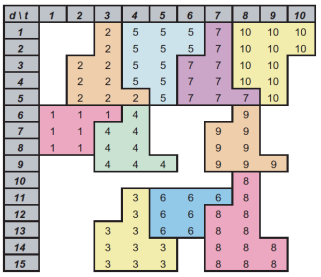


FIGURE A.2: Polyominoes in counter allocation. The x-axis stands for time periods and the y-axis for counter numbers.

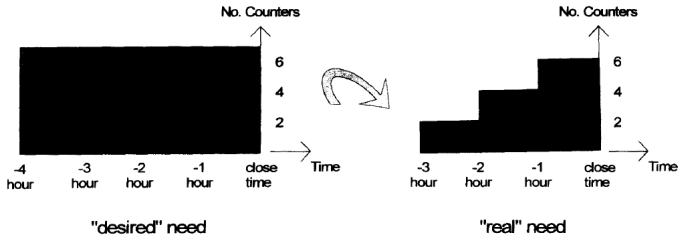


FIGURE A.3: Desired need for airlines as compared to real need

length of the TW has to be chosen while modelling a problem. Too small TWs mean less time for staff to handle operations and large TWs can consist of large variations in passenger arrivals. Fig: A.2 shows counter allocation by Dijk and Sluis (2006) with a TW length of 60 min, planning horizon of 10 hours with resource availability of 15 counters. Here, the numbers represent the flights, 'd' represents the desk number or counter number and 't' represents TWs. Since resource requirement times are fixed, it is not possible to shift items horizontally, only vertical movement is allowed.

These counters are owned by the airport operator and suitably leased to airlines. The airport operator has to issue the minimum number of adjacent counters required for each flight/ group of flights. Adjacent counters are important for passenger and

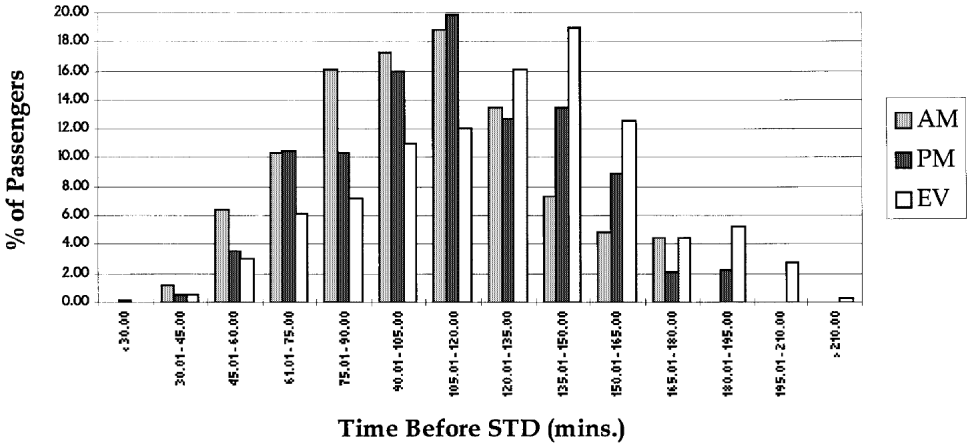
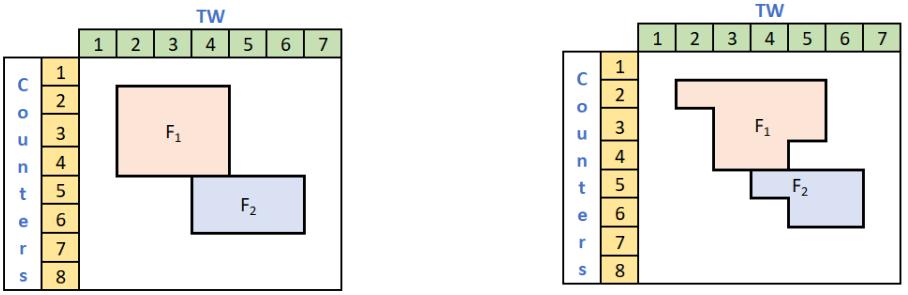


FIGURE A.4: Arrival Distribution Observed at Kai Tak Airport, Hong Kong

airline convenience, airline visibility and operations, and for ease in baggage management and sorting, since baggage from adjacent counters of an island is collected at one baggage collection center. Typically, airlines demand more counters from the airport operator than required, for ease in providing service and visibility at the airport (see Fig. A.3 from Chun (1996)). This creates problems for the airport operator, who has to now simultaneously satisfy the real need for counters, ensure optimal allocation to all airlines and minimize changeover operations. A changeover operation consists of staff of one airline closing the counters at the end of a TW and shifting to other counters for continuing the check-in process in subsequent TW(s). To minimize the changeover operations, counters were traditionally allocated to a flight for its entire check-in time. Since this type of allocation would result in a waste of counter time due to time-varying arrival distribution, time-varying counter requirements are proposed (Dijk and Sluis (2006), Bruno and Genovese (2010), Araujo and Repolho (2015) , Lalita et al. (2020) etc). This results in structures known as polyominoes (see Fig.: A.2 for each flight (or a group of flights). Actual counter requirement (for illustration see Fig. A.5, from Dijk and Sluis (2006)) is computed using the arrival distribution of passengers (for illustration see Fig.: A.4 from Chun and Mak (1999), where arrival distribution is computed for flights in the morning, afternoon and evening), and an appropriate load factor. Load factor is defined as the percentage of passengers of an airplane seat capacity expected to finally board the flight. The counter allocation problem is reduced to determining counter requirement of flights and then placing the resulting polyominoes, that can be moved only vertically, in the counter-TW area in an optimal way.

Flight		1	2	3	4	5	6	7	8	9	10
Number of passengers		150	210	240	180	270	150	210	300	180	270
Starting period		1	2	3	3	4	5	6	7	7	8
Constant		3	4	4	3	5	3	4	5	3	5
Variable	1st period	3	3	3	3	4	3	3	4	3	4
	2nd period	3	5	5	4	5	3	5	6	4	5
	3rd period	1	1	2	1	2	1	1	2	1	2

FIGURE A.5: Minimum Counters Required in constant and variable case



(A) Static Counter Allocation

(B) Dynamic Counter Allocation

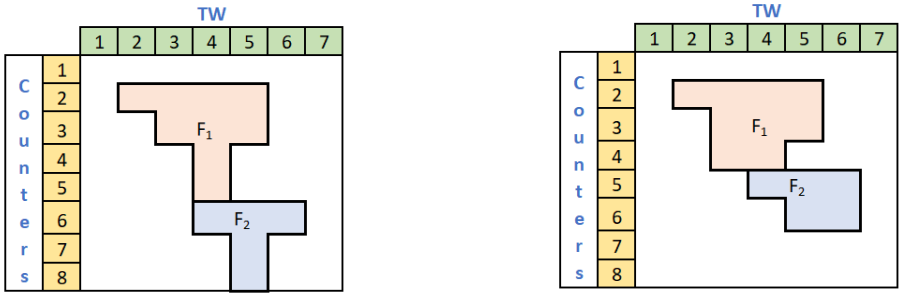
FIGURE A.6: Dynamic Counter Allocation saves counter time

It has been observed that time-varying or dynamic counter allocation (Fig.A.6) provides on-time service and consequently results in significant savings in terms of counter time, operation costs and reduced queue size (Joustra and Van Dijk (2001), Dijk and Sluis (2006)). In contrast, static counter allocation assumes constant counters in the check-in period. In both types of allocation, two or more flights of an airline can be grouped together for assignment if the demand for counters overlaps in some TWs, i.e., if flights are scheduled for departure a few hours apart. It is general practice by airlines to assign common counters to a set of flights with simultaneous demand in at least one TW. The demand for counters is treated as demand for one departure. This enables passengers of an airline to use the same counters irrespective of the flight boarded. Dedicated counters on the other hand exclusively serve passengers of a single flight. Both allocations are used in airports based on opportunities available to group flights. This allows the airlines to minimize the counter operating cost and changeover operations also. In most of the models proposed for counter determination, minimizing the counter operating cost is the objective. Counter operating cost includes the cost of operating baggage belts, the cost of staff operating the check-in counters, queue cost and the cost of delay. Hsu and Chao (2005) construct various cost functions related to facility management.

Some of the challenges to this problem, as discussed by Snowdon et al. (1998), are estimating the passenger arrival distribution, the complexity of determining the passenger mix (number of passengers using different services), changes to the resources (mainly check-in facilities) available depending on the arrival distribution, ensuring passenger service levels are attained and evaluating flight data in order to group flights which can benefit from common counters vis-a-vis dedicated counters. Initial attempts to solve the problem through simulation improved the prevailing methods for counter allocation. Atkins et al. (2003) propose simulation to compare operational strategies and to determine the optimal staff levels required. Chun (1996) and Chun and Mak (1999) use simulation for counter determination and counter allocation. Real life modelling of the check-in process at airports has been presented by Joustra and Van Dijk (2001) and Dijk and Sluis (2006) for check-in at Schiphol airport in Amsterdam, Atkins et al. (2003) at Vancouver airport, Bruno and Genovese (2010) at Naples International Airport, Lous (2011) at the Copenhagen airport, Al-Sultan (2016) at an airport in Kuwait and Felix and Reis (2017) at the airport of Lisbon and Chun (1996), Chun and Mak (1999) at the Hong Kong airport etc. The counter allocation problem with the adjacency restriction is NP-complete (Dijk and Sluis (2006), Duin and Sluis (2006)) and cannot be solved in polynomial time. The complexity of the problem has been studied in detail by Duin and Sluis (2006). All the models proposed for counter allocation including the above real-world applications have been classified on the basis of the problem solved. These methods are discussed in detail below.

A.3 Determining Optimal Number of Check-in Counters

This section discusses the problem of determining counter requirement for a flight or a set of consecutive flights of an airline. The number of counters allocated to a flight (or group of flights) depends on the arrival pattern of passengers, the queueing area available, queue length in an interval and restrictions on waiting time. Different authors have modelled the problem with different constraints, different objectives and different facilities (eg: counters and kiosks). Before studying these models, we present a basic model for counter determination:



(A) Allocation using Basic Formulation

(B) Improved Allocation

FIGURE A.7

$$\texttt{Min} \quad \sum_{ij} x_{ij} \tag{A.1}$$

$$\texttt{subject to} \tag{A.2}$$

$$\sum_{j} a_{ij} s_i \leq \; t \sum_{j} x_{ij} \tag{A.3}$$

$$x_{ij} \geq 0 \;\; \texttt{are nonnegative integers.} \tag{A.4}$$

In this formulation, x_{ij} is the number of counters assigned to the i^{th} flight in the j^{th} TW, a_{ij} is the average number of arrivals for i^{th} flight in j^{th} TW (this is obtained from passenger surveys conducted at the airport), s_i is the average service time per passenger for the i^{th} flight, 't' is the length of each TW. This formulation provides resources exactly in proportion to airline requirement. This results in counter allocation with peaks and troughs exactly like the passenger arrival distribution. Since airport operators mandate airlines to limit waiting times and counter queue lengths, the only way to improve the solution to this model and get more rectangular polyominoes is to postpone the check-in of some passengers while respecting the service level requirements. Note that with even a slightly more rectangular structure of the counters allocated (see Fig.(A.7b)), changeover operations are reduced, improving the basic solution (see Fig.(A.7a)).

Mathematical Models for Counter Determination

Published work on check-in counter determination is described briefly in the following paragraphs. Park and Ahn (2003) published a paper for optimal assignment of check-in counters. Their paper aims to assign counters based on passenger arrival distribution at

the airport. Other factors considered are aircraft type (standard or chartered), aircraft size, time allowed for check-in, passenger arrival distribution, ticket status (economy, business, first class etc), processing time of staff at the check-in counters and load factor assumed. A passenger survey at the airport (in Park and Ahn (2003)) determines the variation in load factors during peak and non-peak hours and the resulting arrival patterns. The airport arrival patterns are then used as input to a regression model to determine the cumulative arrival pattern based on time before departure. Counters are allocated to airlines directly in proportion to the estimated passenger arrivals. This kind of allocation may not result in an optimal assignment of counters to airlines since the overall cost to the airport operator or queue lengths among other things are not considered.

Bruno and Genovese (2010) propose the following static model to determine the optimal number of counters for flights with the objective of reducing counter cost and queue length.

$$\texttt{Minimize} \quad z = \sum_j \sum_t (h_j.I_{jt} + s_j.x_{jt}) \tag{A.5}$$

subject to

$$I_{jt} = I_{j(t-1)} + d_{jt} - q_{jt},\ j = 1, 2, ..., J,\ t = 1, 2, ..., N, \tag{A.6}$$

$$p_j q_{jt} = C_t x_{jt},\ j = 1, 2, ..., J,\ t = 1, 2, ..., N, \tag{A.7}$$

$$\sum_j p_j q_{jt} \leq C_t,\ t = 1, 2, ..., N, \tag{A.8}$$

$$I_{jt} = 0,\ t \in\ T_j \tag{A.9}$$

$$q_{jt},\ I_{jt} \geq 0,\ j = 1, 2, ..., J,\ t = 1, 2, ..., N, \tag{A.10}$$

$$x_{jt} \in 0, 1, j = 1, 2, ..., J, t = 1, 2, ..., N, \tag{A.11}$$

The following notations are used in the above model. h_j is the cost associated with queue related to flight j, s_j is the desk opening cost for flight j, T is the planning horizon (usually one day), l is the length of the TWs considered, N is the number of TWs, J is the number of flights scheduled in T, p_j is the average desk service time for flight j,

d_{jt} is the service demand for flight j in TW t, C_t is the available check-in time based on counters operating in TW t, I_{j0} is the number of passengers of flight j waiting before counters open for flight j, T_j is the set of TWs in which counters for flight j do not operate. Decision variables are: I_{jt}, the number of passengers in queue for flight j at the end of TW t, q_{jt}, the number of passengers of flight j to be accepted for service in TW t. Even though x_{jt}, the binary variable representing the possibility of checking-in passengers for flight j in TW t, is defined as a decision variable, it is not, since this is fixed in advance and cannot be restructured. Constraints (A.6) represent the change in queue length between two successive TWs. Constraints (A.7) ensure that enough counter time is available for passenger check-in in TWs where check-in is possible. Constraints (A.8) ensure the overall service capacity is as required, constraints (A.9) forces all passengers of flight j to be accepted by the closing time of check-in service for flight j.

Bruno and Genovese (2010) propose models for both static (see model (A.5)-(A.11)) and dynamic counter allocation. Both the models define the number of passengers in queue for a flight and the passengers accepted for service in each TW as decision variables. In the static model presented above, the total cost of counter operation and the cost of queue is minimised (see objective function (A.5)). In the dynamic model, counters operating in each TW and the cost associated with queue are minimized. The authors derive mathematical formulations from the Capacitated Lot Sizing problem in literature (see Bitran and Yanasse (1982)) and Florian et al. (1980)). The authors also present a real-life airport management study at the Naples airport.

The model presented by Araujo and Repolho (2015) is an extension of the model by Bruno and Genovese (2010). Araujo and Repolho (2015) present two models for determining counter requirement for flights and aim to determine the optimal number of counters for flights operating at an airport. ILPs are presented for dedicated and common counter check-in. Following is the ILP for dedicated counter allocation.

$$\texttt{Minimize} \quad z = \sum_{j} \sum_{t} (h_j . I_{jt} + s_j . x_{jt}) \tag{A.12}$$

$$\texttt{subject to}$$

$$I_{jt} = I_{j(t-1)} + d_{jt} - q_{jt},\ j = 1, 2, ..., J,\ t = 1, 2, ..., N, \tag{A.13}$$

$$\sum_{j} p_j q_{jt} \leq C_t,\ t = 1, 2, ..., N, \tag{A.14}$$

$$I_{jt} = 0,\ t \in T_j \tag{A.15}$$

$$I_{jt} \leq \alpha.(d_{jt} + I0_{jt}),\ j = 1, 2, ..., J,\ t = 1, 2, ..., N, \tag{A.16}$$

$$q_{jt},\ I_{jt} \geq 0,\ j = 1, 2, ..., J,\ t = 1, 2, ..., N, \tag{A.17}$$

An additional service level constraint (such as constraint (A.16) for dedicated counter allocation as above) is added to both models to ensure that only a small percentage of passengers, α, remain in queue at the end of a TW. The service level is chosen by the airport operator. The models predict the counter requirement for flights (or for a group of flights). The model solutions are analysed using simulation for model validation in different scenarios that may occur at the airport. Once the models are validated, the corresponding solution is implemented by allocating adjacent counters to each flight using the ILP by Dijk and Sluis (2006). Though a service level constraint is added to the model, the waiting time of passengers and the maximum queue length allowed are not considered. It can be easily proved that model solution does not follow FIFO and due to this counter usage may be different than expected.

Lalita et al. (2020) propose a model ((A.18)-(A.22)) for determining the optimal number of counters with postponement of service times. The model considers the passenger arrival distribution, where d_{ji} is the arrivals in TW i for flight j, u_{ji} is the number of arrivals (in TW i), served in TW i and v_{ji} is the number of passengers arrivals for flight j served in TW $i+1$. P denotes the time horizon for these departures. Let $i_o(1)$ denote the counter opening time of the first departure and $i_c(1)$ denote its closing time, then, $P = \{i_o(1), i_o(1)+1, \ldots, i_c(\hat{D})\}$, $P_j = i_o(j), i_o(j)+1, \ldots, i_c(j)$ and $d_{ji} = 0$ for all (j, i) such that $i \notin P_j$.

Constraints (A.19) and (A.20) ensure that all passengers are served, constraint (A.21) ensures that enough counter time is available for serving all the passengers of all the flights. The formulation makes the resulting counter allocation more rectangular by postponing service to the next TW for some passengers. It also limits the waiting time by length of two TWs and follows the FIFO queue discipline. Adding a service level

constraint to this model may improve counter allocation. Lalita et al. (2020) also propose a model for adjacent check-in counter assignment, discussed in section A.4.

$$\texttt{Minimize} \quad z \tag{A.18}$$

$$\texttt{subject to}$$

$$u_{ji} + v_{ji} = d_{ji}, \forall\ i \in P_j,\ j = 1, 2, \ldots, \hat{D}, \tag{A.19}$$

$$v_{ji_c(j)} = 0\ \forall\ j = 1, 2, \ldots, \hat{D}, \tag{A.20}$$

$$\sum_j s_j(u_{ji} + v_{j(i-1)}) \leq \omega c_i\ \forall\ i \in P, \tag{A.21}$$

$$c_i \leq z\ \forall\ i \in P, \tag{A.22}$$

$$u_{ji},\ v_{ji} \texttt{ and } c_i\ \forall\ i \in P,\ \texttt{are nonnegative integers.} \tag{A.23}$$

Hsu et al. (2012) also propose an ILP to determine the number of check-in facilities required for a departure from the airport. Check-in facilities considered are the counters and kiosks, online check-in, and barcode check-in. Check-in services offered by these facilities are : ticket purchase, check-in, boarding pass and checking baggage, each offering one or more services. Each facility may offer different check-in services. The model proposed aims to minimize the passenger waiting time, operation costs and counter requirements. The authors explore dynamic allocation of different check-in facilities and passengers at the airport. For modelling the problem only two types of check-in facilities, counters and kiosks are considered. Passengers are assigned different facilities based on service requirement. The model proposed is based on the dynamic model by Nikolaev et al. (2007). The services required by passengers and their arrival times are predicted and are an important input to the model. The assignment of the n^{th} passenger by the model depends on the assignment of the $(n-1)^{th}$ passenger. Therefore, for dynamic assignment of passengers, various possible scenarios are analysed. Due to this, the model takes more than 3 hours to solve for 15 passenger arrivals. In view of this the authors have used clustering algorithms (in heuristics) for solving the model. A real life scenario has been presented in the study to show improved counter utilization rates on using the model at an airport in Taiwan. A brief analysis of the modelling of the problem suggests that the success of the model is entirely dependent on the willingness of the

passengers to use the check-in facility assigned to them. The model may fail to generate an effective solution at airports where passengers prefer to use other facilities, it may cause the check-in facilities operating at the airport at certain TWs to be insufficient.

Determining the counter requirement at an airport is necessary but sometimes, the capacity of check-in counters at an airport also needs to be assessed. It is essential in decision making for airport expansion. Airport terminal capacity is assessed by Brunetta et al. (1999), Fayez et al. (2008) and Pacheco and Fernandes (2003). Kıyıldı and Karasahin (2008) present a fuzzy logic method to determine check-in capacity at Antalya airport in Turkey.

Capacity of a check-in counter is computed as the number of passengers and luggage that can be checked-in in one hour. In the model, possible passenger and luggage combinations are determined. Number of passengers travelling together and the number of bags (luggage carried by a person/group) is randomly generated and fuzzy logic is used to predict the processing time. It was observed that the processing time is highly correlated with volume of luggage (high R^2). Fuzzy models are used to calculate the capacity of each check-in counter in terms of the number of persons served and the number of luggage items checked-in. This is used to calculate the check-in capacity of the airport which is compared with the current passenger arrivals and growth rate at the airport for adding new resources to the airport.

Hwang et al. (2012) present a mathematical model for optimizing check-in counters, kiosks, part-time staff and full-time staff required during each shift in a week at airports. Their model assumes static counter allocation and defines the number of passengers using counters and kiosks in a shift as parameters. The study conducted gathers information on counter requirements and cost of operation in the presence of various factors including the day of the week and varying load factors. The authors compute the ratio of counters to kiosks that would be best suited for serving a flight. Model solutions are verified for optimizing cost in different scenarios using simulation. The authors conclude that usage of kiosks reduces the operational costs and that services can be improved by installing additional kiosks. Regarding the average waiting time, the study concludes that check-in counter cost can be reduced by limiting passenger service time. Though the observation is fact-based, it is not clear as to how this may be achieved. This recommendation

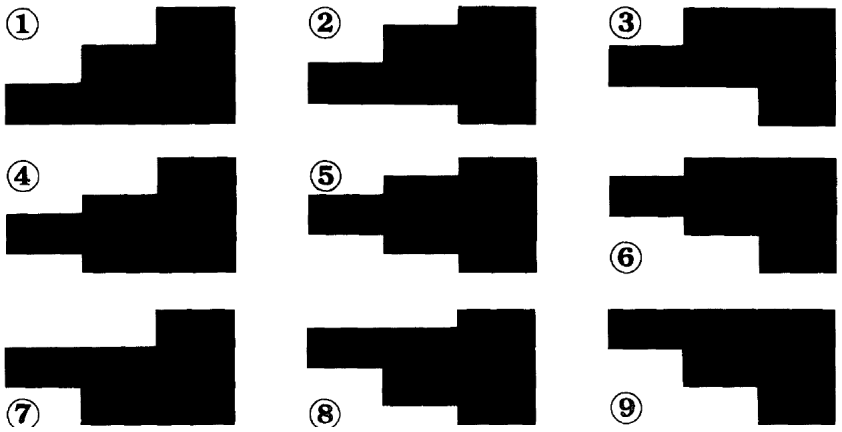


FIGURE A.8: Possible variations of the 2-4-6 Counter Profile

also contradicts the existing counter planning strategies of using average service time to determine counter allocation instead of fixing service time per passenger. Reducing service time per passenger would definitely make the counter allocation problem easier, but it may not be always possible to limit passenger processing time.

Simulation has been used for counter determination by Chun (1996), Chun and Mak (1999) and Dijk and Sluis (2006).

A.4 Adjacent Counter Allocation

The most challenging problem at an airport is adjacent counter allocation for a flight while simultaneously maximizing counter utilization. Chun (1996) first addressed this problem by defining structures called counter profiles.

A counter profile is a two dimensional shape that defines the check-in counter resource requirement for a flight. In the algorithm proposed by Chun (1996), different operators are presented to change the shape of the counter profile to check for a convenient fit for allocation in the counter-TW rectangle of fixed dimensions. All possible shapes are considered for every counter profile (see Fig.A.8). Placing a counter profile in a two dimensional counter TW space ensures adjacency of counters. Algorithm by Chun (1996) starts with selecting one possible counter profile for a flight, it is assigned a location in the counter TW space (also Gantt Chart), all the relevant constraints are checked, if some of the constraints are violated (i.e. if the specific shape of the counter

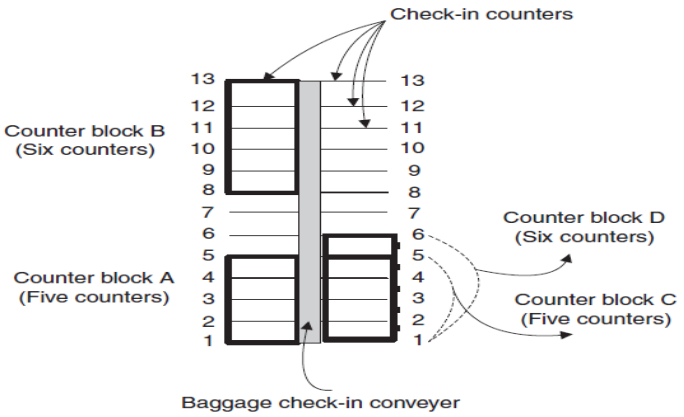


FIGURE A.9: Blocking by Yan et al. (2004)

profile cannot fit in the Gantt Chart), the algorithm backtracks by removing the previous allocation to a flight and freeing up space. The counter profile shapes may need to be changed in order to accommodate the remaining flights. Chun (1996) simulated different possible shapes of the counter profile, similar to fitting pieces of a puzzle randomly until an acceptable solution was reached. If the order in which flight counter profiles are allocated does not have a feasible solution, the algorithm involves backtracking to remove flight allocations. This results in a time consuming hit and trial process for counter allocation. An ILP is presented by Dijk and Sluis (2006) for the same problem. The ILP considers all possible arrangements of a counter profile in the counter-TW area, till an optimal solution is obtained. Due to this, the ILP takes a long time to converge to a feasible solution (especially for large flight departures). The formulation is effective for small size problems compared to the real-world problems of the day. In the static counter allocation problem considered by Yan et al. (2004) and Yan et al. (2005), each counter has one or two service lines to provide check-in service to passengers. To allocate adjacent counters, the authors define each block as a set of service lines. Tang (2010) defines each block as a set of adjacent counters. Two blocks may overlap as a service line/ counter may belong to both (see Fig. A.9).

Service lines are demanded by airlines in accordance with the passenger arrival pattern and the number of passengers on the flight. Yan et al. (2004) propose a static model to assign blocks to flights. The objectives of the study are to allocate counters to minimize passenger walking distance and reduce inconsistency in counter location. Allocation to different blocks of counters on different days of the week for the same

flight is defined as an inconsistency. Inconsistency values are introduced as the airlines prefer to have the same block of counters allocated for flights in a week.

Passenger walking distance is calculated for possible flight assignments to a block and all possible blocks to which a flight can be assigned are considered in the model. For a given flight and block combination, it is the average distance a passenger walks after check-in from the block allocated till boarding. An ILP is proposed for minimizing the passenger walking distance subject to allowable inconsistency. The constraints of the ILP need preprocessing to exclude redundant constraints. Since the solution method lacks scalability, three heuristic models are proposed to solve the problem. The first model provides the assignment for a day. Based on this, inconsistency values are set and the second model is solved for a minimum inconsistency value. Then, the third model is solved for the final assignment of flights to blocks. A heuristic is thus provided for solving the model for a single day. This heuristic has been used in real-life modelling at an airport in Taiwan. The models are used for allocating 140 counters to 70 flights departing from the airport. The number of variables exceed 80000 and constraints exceed 70000. Results from the case study conducted for Taiwan Airport show reduction in the passenger walking distance by 4%.

Yan et al. (2005) study the dynamic counter allocation problem. Their objective is to allocate adjacent counters to flights by minimizing the total inconsistency in blocks allocated to a flight. Blocking plans are similar to blocking by Yan et al. (2004). Each counter may have many service lines (as also defined by Yan et al. (2004)) and the service line requirement for each flight in each TW is known. Since the number of service lines differ from one block to another, additional lines need to be opened or closed between successive TWs. This adjustment of service lines between two consecutive TWs is considered an inconsistency. From the inconsistency values attached to blocks pairwise, the inconsistency value in flight allocation on a day is computed. The problem was solved by an ILP, allocating a block (possibly different) to each flight in each TW. Due to the complexity involved in computing the inconsistency values, the problem becomes increasingly complex with increase in number of flights, and the authors propose a heuristic algorithm. The heuristic for the problem divides it into subproblems, each of which is solved. Each subproblem comprises of equal counters and flight departures. An exchange mechanism between the two parts is proposed to proceed further (similar to

the evolutionary algorithm by Mota (2015)). The exchange of two flights, one selected from each group, (with overlapping departure times), and then re-solving for a solution is continued till a reduction in inconsistency is achieved. The allowable inconsistency value is chosen as appropriate. The solution with the least inconsistency value (according to the stopping criterion) is chosen. The order of exchanges effects the final solution, hence, further improvements to the heuristic algorithm are suggested. For large problems, dividing flights into two groups may be insufficient. Due to this number of groups is increased and the solution algorithm is improved. A disadvantage is the increase in complexity of the problem with additional subgroups, as a result of increase in the number of flights.

Mathematical Models to ensure Adjacency

Dijk and Sluis (2006) model the problem combining both simulation and integer programming. The objectives of the study are to determine the minimal counter requirement for each flight and then to allocate counters at the airport with adjacency maintained. The problem is solved in two stages. In the first stage terminating simulations are run to determine counter requirement till the solutions satisfy service level requirements. In the second stage, an ILP is solved for adjacent allocation of counters. The study presents ILPs for both variable and constant counter allocation. These models ensure adjacency of the counters allocated to each flight and solve to an optimal solution. The ILP for variable counter allocation is given below:

$$\texttt{Minimize} \quad D \tag{A.24}$$

$$\texttt{subject to}$$

$$n_{ft} \leq d_{ft} \leq D, \ \forall f \ \texttt{and} \ t = a_f, \tag{A.25}$$

$$d_{ft} + n_{gt} \leq d_{gt} \ \texttt{or} \ d_{gt} + n_{ft} \leq d_{ft}, \ \forall f, g \ \texttt{and} \ t \in I_f \cap I_g, \tag{A.26}$$

$$d_{ft} - d_{ft-1} \leq \texttt{max}\{0, n_{ft} - n_{ft-1}\}, \ \forall f \ \texttt{and} \ t \in (a_f, b_f], \tag{A.27}$$

$$d_{ft-1} - d_{ft} \leq \texttt{max}\{0, n_{ft-1} - n_{ft}\}, \ \forall f \ \texttt{and} \ t \in (a_f, b_f], \tag{A.28}$$

where, D is the total number of counters required, I_f is the check-in interval of flight f (or counter operating time), n_{ft} is the number of desks required for the check-in

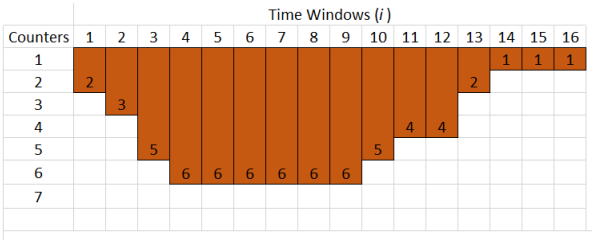


FIGURE A.10: Task Structure for a given Counter Allocation

process of flight f in period t ($t \in I_f$), and d_{ft} is the largest desk number assigned to flight f in period t ($t \in I_f$). Constraint (A.25) ensures that counters assigned to flights do not exceed D, constraint (A.26) ensures that two flights are not assigned the same counter in a TW and also avoids overlap, constraints (A.27) and (A.28) ensure that required counters are opened or closed at the beginning of a TW. Model inputs are the counter requirements (n_{ft}) for flights determined using simulation. Queuing theory and simulation methods are both evaluated, it is observed that queueing theory cannot be applied since the arrival distribution of passengers for different flights is not homogeneous and cannot reach a steady state. Simulation and the model ((A.24)-(A.28)) were applied to data from a Dutch airport and it was observed that increased problem size (increase in number of flights and planning horizon) resulted in an increase in computation time. A similar mathematical model is presented by Duin and Sluis (2006). The model is adapted from RPSP (see Pinedo and Chao (1998)).

Lalita et al. (2020) also present an ILP for allocating adjacent counters. The ILP ((A.31)-(A.35)) has been shown to solve problems with a large number of departures in less time compared with formulations by Araujo and Repolho (2015) and Bruno and Genovese (2010). Tasks are defined as time-varying counter requirement for one or more departures (see Fig.A.10). This definition is similar to that of counter profiles by Chun and Mak (1999). The difference being that there cannot be multiple task structures whereas the shape of a counter profile can be changed using different operators. A counter-TW pair, (k, i) is fixed and variables w_{ki} and m_t are defined by

$$w_{ki} = \sum_{t \in \mathcal{T}_i} \sum_{h=1}^{c_i(t) \wedge k} y_{t(k-h+1)}, \tag{A.29}$$

$$m_t = \max\{c_i(t) : i_o(t) \leq i \leq i_c(t)\}, \tag{A.30}$$

where $i_o(t)$ is the counter opening time and $i_c(t)$ is the counter closing time for task t, $\mathcal{T}_i$ is the set of all tasks t such that $i_o(t) \leq i \leq i_c(t)$, w_{ki} is the number of tasks that use counter-TW combination $(k,\ i)$, y_{tk} is a binary variable, equal to 1 if task t starts at counter k, and $\sum_k ky_{tk} + m_t - 1$ is the largest counter number used by task t.

$$\texttt{Minimize} \quad s \tag{A.31}$$

$$\texttt{subject to}$$

$$\sum_{t\in\mathcal{T}_i} \sum_{h=1}^{c_i(t)\wedge k} y_{t(k-h+1)} \leq 1 \ \texttt{for all}\ (i,k), \tag{A.32}$$

$$\sum_k y_{tk} = 1 \ \texttt{for all}\ t, \tag{A.33}$$

$$\sum_k ky_{tk} + m_t - 1 \leq s, \ \texttt{for all}\ t, \tag{A.34}$$

$$y_{tk} \in \{0,\ 1\} \ \texttt{for all}\ t,\ k, \tag{A.35}$$

Constraint (A.32) ensures that no counter is allocated to more than one task in any TW. Constraint (A.33) ensures that each flight is allocated counters and constraint (A.34) limits the total number of counters used to s.

A.5 Some Real-world Airport Applications

This section discusses models proposed for solving real-life airport problems. We focus on the airport problem considered by the authors, inputs required for the models considered, and prominent disadvantages and advantages to using these models. Joustra and Van Dijk (2001) present a simulation model and Dijk and Sluis (2006) present ILPs in addition to simulation for check-in counter allocation, Atkins et al. (2003) present a simulation model with input data on pre-board screening, shift scheduling, passenger arrivals, and level of service for determining staff scheduling and resource requirements. Simulations were run till the staff schedule obtained satisfied the level of service at the airport. Operations in airports vary in type of check-in, type of queue permitted, flexibility of counter usage (for business or economy class), passenger behaviour, common

or dedicated counters, check-in periods, baggage collection centers and sorting of baggage, queuing area etc. Consequently, simulation models consider different features and requirements of the airport under consideration. Bruno and Genovese (2010) present two models for counter determination. The model by Lous (2011) considers the baggage belt direction, amount of baggage allowed per baggage area, the maximum number of people allowed to queue at a counter, adjacency of counters allocated, counter location preferences by an airline, but aiming to create a flexible model results in a complicated model, and involves a large number of additional computations, especially in allocating preferred counters to airlines. In the study by Al-Sultan (2016), some counters are always kept unused in each zone of the airport to cushion the airport operator against sudden increase in traffic. The author overlooks the model objective function which is constant, the model is thus defective and consequently, the objective of minimizing the counters allocated may not be achieved. Felix and Reis (2017) developed a hybrid discrete-event and agent based simulation model to assess the performance of check-in process at the airport of Lisbon. Passenger behaviour, the sequence of tasks performed in the check-in area and the physical layout of the check-in area are incorporated in the model. The variations in the check-in process are attributed to variation in these aspects. The model by Felix and Reis (2017) is comparable to the model by Lous (2011) in that it considers almost the same set of factors effecting passenger check-in.

In Felix and Reis (2017), the simulation model proposed aims to explore various scenarios at check-in area and enables airport operators to choose the best position and assignment of different types of check-in facilities (counters, kiosks, etc) to airlines (counters and kiosks differ in the services offered). However, in Lous (2011), baggage collection centers with limited capacity and limited queueing area are incorporated in the ILP proposed. Trakoonsanti (2016) also models the problem for an airport in Thailand. The excel based software SimQuick is used to build the model for simulation. The arrival distribution of passengers, check-in service time (or its distribution), type of queue, are input to the software. Different structures at the airport such as the check-in counters, entrances, exits, queue capacity, and other processes can be defined and average time for passenger flow through these processes including the check-in counters can be observed. The paper concludes with different results on the efficiency of counter allocation in terms of passenger waiting times and queue length.

A.6 Different Approaches to Counter Allocation

This section discusses different approaches for check-in counter assignment. The approaches used to model the problem are classified below based on the problem solved and procedures used.

A.6.1 Simulation for Counter Allocation

Initial attempts to solve the counter allocation problem were made by simulation of resource requirement at airports. Constraint satisfaction algorithms were presented by Chun (1996) and Chun and Mak (1999). Subsequently, simulation of passenger flow at the airport terminal was proposed by Wong and Liu (1998) and passenger flow from terminal entrance to boarding was simulated by Kiran et al. (2000). Wong and Liu (1998) focus on passenger traffic characteristics and their impact on terminal operations. Simulation of resources may also be done to identify delays at the check-in system and create scenarios that will improve the efficiency (Appelt et al. (2007)). Real-time system data was used by Ros Prat (2017) to simulate system behaviour and test different conditions and scenarios with the objective of optimizing the check-in procedure at Brisbane Airport. The author presents a dynamic check-in procedure where rapid changes in the check-in procedure are possible in real-time. Simulation was used for counter determination by Dijk and Sluis (2006). For analysis of counter allocation and waiting time of passengers, Bevilacqua and Ciarapica (2010) compared both waiting times and other parameters estimated by queuing theory and simulation. Chun (1996) presents algorithms by modelling the problem as a multidimensional placement problem. Chun (1996) proposes a two-dimensional approximation for counter scheduling where space (counters) and time are assumed to be part of a Gantt chart. The constraint satisfaction algorithms presented consider all possible counter profiles (A.8) for each flight. The final version of the algorithm aims to further improve the solution by considering different shapes per counter profile and by imposing constraints on the minimum number of counters allocated to a flight based on the queue length restrictions. The main drawback of this method is that final allocation depends on the order of flights chosen by the algorithm for allocation. For the final algorithm, counter profile globs are defined as multidimensional objects which include additional dimensions such as queue lengths,

waiting time and baggage restrictions. Also, in case all the flights cannot be allocated to counters, despite re-shaping the counter profile globs and minimizing the counters allocated, the algorithm backtracks, de-assigns the most recent allocation and proceeds to allocate for another flight. This is a major drawback for airports with a large number of departures. There may be too many backtracks in deriving the final solution. Also, the final allocation depends on the order of the flights chosen. Choosing the best counter profile has not been addressed in this paper, which is very much needed for arriving at a good decision.

Chun and Mak (1999) present an intelligent resource simulation system (IRSS). IRSS predicts the check-in counter requirement at an airport. This is a software which takes the airport flight data, passenger turn up data, check-in counter data, and other model parameters as input to generate a simulation model. The IRSS proposed also has a graphic user interface to simulate and animate the check-in counter queueing for a single flight. Simulation parameters such as tolerable passenger waiting time, queue lengths, service rates, check-in times for flights, the number of check-in counters, number of passengers or time of departure can be changed to match the reality at the airport and to observe the change in counter allocation. The objective is to find a counter profile (a counter assignment solution) which saves the maximum counter time compared to the current counter allocation and maintains the desired service level quality. The authors also account for the stochastic processes such as arrival rates. The Check-in Counter Allocation System (CCAS) presented by Chun (1996) together with the IRSS are capable of addressing check-in counter allocation problem at an airport and were used together in the Kai Tak International Airport. The solutions obtained are evaluated by analysing the waiting times of passengers and queue lengths. Though the algorithm tries to achieve the best possible solution, a major disadvantage is that not all possible counter profiles can be simulated. The authors examine some counter profiles to arrive at the best counter allocation (Chun (1996)). Due to this we cannot ensure that the service quality level provided is the best possible. Also, for a large number of flights and time windows, algorithms with simulation will take a large amount of time.

Bevilacqua and Ciarapica (2010) present a case study and an analysis of counter allocation through simulation. Their objective is to calculate minimum counter requirement

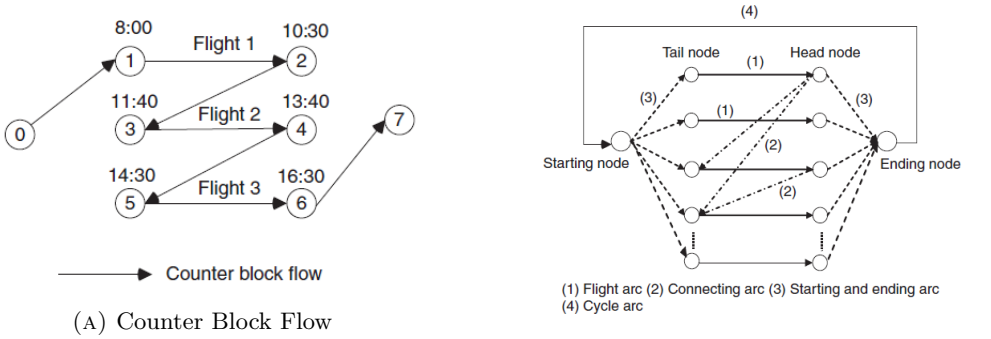
of each flight such that average waiting time does not exceed the maximum allowable limit. Passenger arrivals and counter operations are simulated for a single flight or a group of flights. Poisson distribution was used to generate arrivals. The model assumes steady state of the system. In the case study presented, counter operations are modelled as a queueing theory system and stability conditions for common check-in were calculated. Simulation analysis shows that common check-in is better than dedicated check-in. Simulation results are compared with standard results from queuing theory. The effect of varying arrival time distribution and number of counters allocated to a flight on the average waiting time is computed. It is observed that queueing theory is more applicable for common check-in and simulation is more suited for dedicated check-in.

Using simulation for decision making process has the main disadvantage that it explores only a small subset of the whole possible scenarios that can be reached by the system under study, thus reducing its optimization potential (Mota (2015)). To overcome this problem many authors have proposed mathematical models for determining counter requirements and for ensuring adjacency in counter allocation. Some of the models are discussed in the sections A.3 and A.4.

A.6.2 Network Model for Counter Allocation

Tang (2010) developed a network flow model for allocating blocks of counters to airlines. The network flow model aims to optimize counter usage at a Taiwan airport. The model uses predefined blocks of counters at the airport and opening and closing counter times of flights. Each flight is defined using an arc, the opening time and the closing time form connecting nodes. Arcs are used to denote sequential flight allocations in a counter block (Fig. A.11a).

The opening time, closing time and counter requirements for a flight are inputs to the model, as determined by airlines. An ILP meeting all the constraints at the airport is proposed. Flights, set of nodes and arcs that can be assigned to a block are all inputs to the model resulting in a counter block flow network for each block. A network can also be used to represent all the possible flight assignments to a block in



(A) Counter Block Flow

(B) Multiple Counter Block Flow Networks

FIGURE A.11: Networks for Counter Allocation by Tang (2010)

a day (Fig. A.11b). The main advantages of the network flow method are allocation of adjacent counters and convenient representation of the flight sequence allocated to a counter block. The model involves preprocessing to eliminate redundant constraints (in the example presented by Tang (2010), about 30% of the constraints are redundant) and computing parameters before constructing the model. Due to this preprocessing, running time varies exponentially with input size (the number of flights scheduled and the number of counter blocks allowed). The model is used to find near-optimal solutions at the Taiwan Taoyuan International Airport, though implementation becomes extremely complicated for assigning time-varying counters to flights. The main drawback of their model is that the counters are predivided into blocks for allocating to airlines which results in a lot of preprocessing. Since multiple blocks can be built with the same counter, the number of possible blocks can be very large complicating the process of counter allocation.

A.6.3 Evolutionary Algorithms and Counter Allocation

Some authors have proposed genetic and evolutionary algorithms for the check-in counter allocation problem (Yeung and Chun (1995), Mota and Alcaraz (2015), Mota (2015), Mota and Zuniga (2013)). Evolutionary techniques are a group of methods inspired by common evolutionary processes. These techniques are expected to provide good solutions, i.e. solutions that are close to optimal but may not be optimal(see Goldberg (1989) and Mota (2015)). The efficiency of these techniques relies highly on parameters that drive the selection procedure (see Mota (2015) and Affenzeller et al. (2009)). Genetic

algorithms are a part of evolutionary techniques. Genetic algorithms (GAs) are defined as efficient, adaptive and robust search and optimization processes that are applied in large and complex search spaces. GAs are modelled on the principles of natural genetic systems where the genetic information of each individual or potential solution is encoded in structures called chromosomes. GAs compute a fitness function for directing search in more promising areas. Each individual has an associated fitness value, which indicates its degree of goodness with respect to the solution it represents. GAs search from a set of points called a population and various biologically inspired operators like selection, crossover and mutation are applied to obtain better solutions (Bandyopadhyay and Pal (2007)).

Yeung and Chun (1995) use fitness directed scheduling to develop an airport check-in counter allocation system based on genetic algorithms. Populations of individuals are genetically bred according to Darwinian principles, i.e., reproduction of the fittest and crossover operations. Each individual represents a check-in counter allocation plan for one day (it is the allocation on a Gantt Chart). Each individual also has an associated fitness measure. Fitness measure is in terms of the number of overlaps found in an allocation plan. Lesser the overlaps, fitter the allocation plan. The fittest individual is the best allocation plan for that day. A population of individuals is randomly created and fittest individuals are selected for the crossover operation. The crossover operation is to create offspring counter allocation plans from the selected individuals in the population. By recombining randomly chosen assignments of the fittest allocation plans, we produce new allocation plans. This new population replaces the previous population and the entire process is repeated to create new generations. The best allocation plan that appeared in any generation is the best plan for check-in counter allocation problem.

Mota (2015) presents a methodology that combines evolutionary techniques and simulation and aims to provide a solution which is better than the solution obtained by applying these techniques independently. The algorithm uses a brute-force approach. Flights are allocated sequentially, taking into account all the constraints in flight allocation such as no overlap, counters opened three hours before departure etc. After all the flights are allocated an initial solution is obtained. This is similar to the algorithm first-fit. In order to get varying solutions, the flight order is changed before each allocation. A population of Counter allocation plans is thus obtained. Next, the solutions

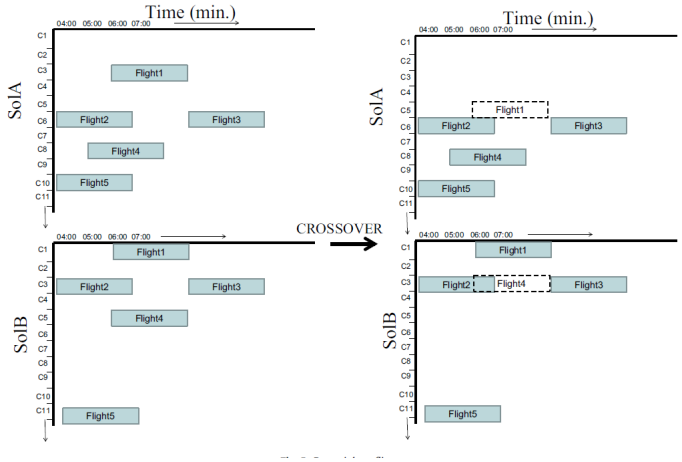


FIGURE A.12: Crossover of two solutions

are converted into vectors (chromosomes) with information that will be used by the evolutionary algorithm. Crossover operations are performed (see Fig.A.12) to improve the existing solutions such that feasibility of the generated solution is maintained. To perform a crossover between any two initial solutions, flights are selected randomly from each of the two solutions and compared. For a pre-decided percentage of flights, with matching counter opening and closing times, counter locations are interchanged. The crossover procedure is performed on solutions with higher fitness values. The resulting solutions with a high measure of fitness are then retained and again crossed over. Though feasibility of the resulting solution is ensured, the computation time to determine feasibility for crossover is very large and increases with the size of the input. For instance, for an airport with about 2500 departures in a week, to perform a crossover, 2500*2500 flight combinations need to be checked for possibility of a crossover. The cost function is computed for each generation and checked for improvement. Since the problem is multi-objective, an objective function is computed as a fitness measure. In the algorithm proposed, the solutions are improved till a stop condition is reached. The stop condition is arbitrarily determined and feasible solutions obtained this way are analysed in real-life conditions using simulation.

A major problem with genetic and evolutionary techniques is that the solution(s) with best fitness may not be anywhere near optimal, but is only relatively better among the allocations generated. Larger the initial population of counter assignments higher the possibility of improving the initial solution. For n flights, the algorithm by Yeung

and Chun (1995) needs $(1 + \binom{2^n}{2}) * (2^n)$ steps, where 2^n is the number of possible recombinations of two counter allocations. This results in unnecessarily prolonged and time consuming calculations for eliminating poor solutions, thus, the number of steps for the algorithm increase exponentially with n.

A.6.4 Queuing Theory and Counter Allocation

Parlar and Sharafali (2008) propose a dynamic programming technique based on queuing theory for determining the counter requirement of a flight. Queueing theory results are used to model the problem. A pure death process is used to model the arrivals. An exponential service time distribution is assumed (Erlang distribution is also explored). The rate of service is observed as proportional to the queue size, hence, service rate is assumed to be state-dependent. The exact forms of distributions of the arrival rate and the service rate are found. These are used to calculate the expected number of passengers in the system and the cost of passengers waiting to be served. Since the arrival rate is time dependent and the arrival process is non-stationary, arrivals are observed and counter operating time is divided into smaller subintervals with constant arrival rate. The arrival rate is then estimated. A dynamic programming model is then used to determine counters to be opened. Counter opening or closing decisions are expected to be made every 20 minutes to minimize the total expected cost. Parlar and Sharafali (2008) propose a model to determine optimal time varying counter allocation for a single flight. Parlar et al. (2013) propose static counter allocation policy for a single flight. Their objective is to minimize the expected total cost of waiting, counter operation, and passenger delay which the authors show to be convex in the number of counters allocated. The authors also introduce a service level constraint to ensure that a certain percentage of passengers are served in each TW.

A.7 Related Scheduling Problems

Two problems related to ACAP, the two dimensional strip packing problem and resource constrained project scheduling (RPSP), have been studied extensively (see Lodi et al. (2002), Amoura et al. (2002), Blazewicz et al. (1986), Duin and Sluis (2006) for further

details). The two dimensional strip packing problem comprises of allocating rectangular items to a larger standard size rectangle with the objective of minimizing waste. The problem, $P|fix_j|C_{max}$ in multiprocessor scheduling, after swapping time and place is equivalent to the adjacent resource allocation problem with rectangular units (for further details see Duin and Sluis (2006) and Amoura et al. (1997)). The problem with irregularly shaped units, such as polyominoes (see Fig.A.2), is similar to RPSP. Hence, a well known integer programming formulation of RPSP (see Pinedo and Chao (1998), Duin and Sluis (2006)) can be modified to solve ACAP (Duin and Sluis (2006)).

Duin and Sluis (2006) discusses similarities of the counter allocation problem with other resource allocation problems in literature such as the two-dimensional strip packing problem (see Lodi et al. (2002)) and the resource constrained project scheduling problem (see Blazewicz et al. (1986) and Du and Leung (1989)).

Staff rostering/human resource management problems at the airport check-in counters are discussed by Lin et al. (2015), Zamorano et al. (2018), Rodič and Baggia (2017) and Xin et al. (2014). Xin et al. (2014) discusses both counter allocation and staff rostering.

Bruno et al. (2018) propose a model to optimize shift scheduling decisions of desk operators and service level measured in terms of passenger waiting times at the counters. A real-life case study has been presented for two airports in Italy.

Hsu and Chao (2005) study optimal facility purchase and replacement. Brunetta et al. (1999) evaluates an airport terminal and estimate delays due to facilities such as the check-in counters. Fayez et al. (2008) estimates the passenger flow through the airport. Efficient use of airport capacity is discussed by Pacheco and Fernandes (2003).

Kim et al. (2017) propose re-assignment of space reserved for airlines on the main conveyor belt to achieve a balanced throughput for check-in counters of different airlines in a check-in area. This method of baggage transfer also results in lowering the maximum check-in waiting time across all counters.

Yan et al. (2008) and Yan et al. (2014) have worked on reassignments of counter allocations in case of sudden unexpected events at the airport such as change in flight schedule, baggage belt malfunction, airport closure and other disturbances to the planned counter allocation. A mathematical formulation has been presented by Yan et al. (2014) with the objective of reducing the impact of unforeseen circumstances at the airport. An inconsistency for a flight is defined as the deviation between original and reassigned counters. The model aims to reduce the inconsistencies in assignment. A heuristic is proposed for solving the model. The model is modified into two relaxations to obtain an upper bound and lower bound. The two relaxations of the model are repeatedly solved till the difference between the two is lower than a predefined limit. The main advantage of the formulation is that it helps the airport authorities with reassignment of counters to restore normalcy and contain the impact of a disturbance to as few flights as possible. Some numerical validation to the model is given. A major disadvantage is that all flights may not be reassigned adjacent counters. Different counters in different TWs may result in confusion for the customers, as it is not possible to shift passengers in a queue from one counter to another.

Various methods of solving the counter allocation problem at airports are presented in here. It is observed that determining counter requirements for flights and then allocating adjacent counter space is most suitable to obtain a practical solution to the adjacent counter allocation problem.

Appendix B

Datasets and Computer Programs associated with the Thesis

1. For the staff scheduling problem presented in Chapter 2, the dataset used has been published (Lalita and Murthy (2021b)) and is available at https://data.mendeley.com/datasets/bk33vpzvkd/1. The code used for simulating instances of ISTSP in Chapter 2 is presented in Github Repository available at https://github.com/trlalita/ISTSP.

2. The airline data used in Chapter 3 for developing methods for check-in counter allocation is given as supplementary data to our paper, Lalita et al. (2020) on check-in counter allocation at https://www.sciencedirect.com/science/article/abs/pii/S0969699719300444.

3. The wind power forecast data used in Chapter 5 has been published (Lalita and Murthy (2021a)) and is available at http://dx.doi.org/10.17632/3t5ns4x58r.1.

4. The code used for simulating instances of the Berth Allocation Problem is available at https://github.com/trlalita/BAP/tree/main.

Bibliography

Abdelaziz, S. G., A. A. Hegazy, and A. Elabbassy (2010): "Study of Airport Self-service Technology within Experimental Research of Check-in Techniques Case Study and Concept," *International Journal of Computer Science Issues (IJCSI)*, 7, 30.

Abujarad, S. Y., M. Mustafa, and J. Jamian (2017): "Recent approaches of unit commitment in the presence of intermittent renewable energy resources: A review," *Renewable and Sustainable Energy Reviews*, 70, 215–223.

Affenzeller, M., S. Wagner, S. Winkler, and A. Beham (2009): *Genetic algorithms and genetic programming: modern concepts and practical applications*, Chapman and Hall/CRC.

Agra, A. and M. Oliveira (2018): "MIP approaches for the integrated berth allocation and quay crane assignment and scheduling problem," *European Journal of Operational Research*, 264, 138–148.

Ak, A. (2008): "Berth and quay crane scheduling: problems, models and solution methods," Ph.D. thesis, Georgia Institute of Technology.

Al-Dhaheri, N. and A. Diabat (2015): "The quay crane scheduling problem," *Journal of Manufacturing Systems*, 36, 87–94.

Al-Sultan, A. T. (2016): "Optimization of Airport Check-In Scheduling at Passenger Terminal," *International Journal of Applied Business and Economic Research*, 14, 3233–3245.

Alfares, H. K. (2004): "Survey, categorization, and comparison of recent tour scheduling literature," *Annals of Operations Research*, 127, 145–175.

Alfares, H. K. and J. E. Bailey (1997): "Integrated project task and manpower scheduling," *IIE transactions*, 29, 711–717.

Alvarez-Valdes, R., E. Crespo, and J. Tamarit (1999): "Labour scheduling at an airport refuelling installation," *Journal of the Operational Research Society*, 50, 211–218.

Amoura, A. K., E. Bampis, C. Kenyon, and Y. Manoussakis (1997): "Scheduling independent multiprocessor tasks," in *European Symposium On Algorithms*, Springer, 1–12.

——— (2002): "Scheduling independent multiprocessor tasks," *Algorithmica*, 32, 247–261.

Appelgren, L. H. (1969): "A column generation algorithm for a ship scheduling problem," *Transportation Science*, 3, 53–68.

Appelt, S., R. Batta, L. Lin, and C. Drury (2007): "Simulation of passenger check-in at a medium-sized US airport," in *Simulation Conference, 2007 Winter*, IEEE, 1252–1260.

Araujo, G. E. and H. M. Repolho (2015): "Optimizing the Airport Check-In Counter Allocation Problem," *Journal of Transport Literature*, 9, 15–19.

Arlitt, M., C. Bash, S. Blagodurov, Y. Chen, T. Christian, D. Gmach, C. Hyser, N. Kumari, Z. Liu, M. Marwah, et al. (2012): "Towards the design and operation of net-zero energy data centers," in *13th InterSociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems*, IEEE, 552–561.

Atkins, D., M. Begen, B. Luczny, A. Parkinson, and M. Puterman (2003): "Right on queue," *OR/MS Today*, 30.

Avramidis, A. N., W. Chan, M. Gendreau, P. L'ecuyer, and O. Pisacane (2010): "Optimizing daily agent scheduling in a multiskill call center," *European Journal of Operational Research*, 200, 822–832.

Aykin, T. (1996): "Optimal shift scheduling with multiple break windows," *Management Science*, 42, 591–602.

Bailey, J., H. Alfares, and W. Y. Lin (1995): "Optimization and heuristic models to integrate project task and manpower scheduling," *Computers & Industrial Engineering*, 29, 473–476.

Bandyopadhyay, S. and S. K. Pal (2007): *Classification and learning using genetic algorithms: applications in bioinformatics and web intelligence*, Springer Science & Business Media.

Barnhart, C., E. L. Johnson, G. L. Nemhauser, M. W. Savelsbergh, and P. H. Vance (1998): "Branch-and-price: Column generation for solving huge integer programs," *Operations research*, 46, 316–329.

Bassett, M. (2000): "Assigning projects to optimize the utilization of employees' time and expertise," *Computers & Chemical Engineering*, 24, 1013–1021.

Beliën, J. and E. Demeulemeester (2008): "A branch-and-price approach for integrating nurse and surgery scheduling," *European journal of operational research*, 189, 652–668.

Beliën, J., E. Demeulemeester, P. De Bruecker, J. Van den Bergh, and B. Cardoen (2013): "Integrated staffing and scheduling for an aircraft line maintenance problem," *Computers & Operations Research*, 40, 1023–1033.

Bellenguez-Morineau, O. and E. Néron (2007): "A branch-and-bound method for solving multi-skill project scheduling problem," *RAIRO-operations Research*, 41, 155–170.

Bevilacqua, M. and F. Ciarapica (2010): "Analysis of check-in procedure using simulation: a case study," in *Industrial Engineering and Engineering Management (IEEM), 2010 IEEE International Conference on*, IEEE, 1621–1625.

Bhulai, S., G. Koole, and A. Pot (2008): "Simple methods for shift scheduling in multiskill call centers," *Manufacturing & Service Operations Management*, 10, 411–420.

Bierwirth, C. and F. Meisel (2010): "A survey of berth allocation and quay crane scheduling problems in container terminals," *European Journal of Operational Research*, 202, 615–627.

——— (2015): "A follow-up survey of berth allocation and quay crane scheduling problems in container terminals," *European Journal of Operational Research*, 244, 675–689.

Bitran, G. R. and H. H. Yanasse (1982): "Computational complexity of the capacitated lot size problem," *Management Science*, 28, 1174–1186.

Blazewicz, J., M. Drabowski, and J. Weglarz (1986): "Scheduling multiprocessor tasks to minimize schedule length," *IEEE Transactions on Computers*, 389–393.

Blazewicz, J., K. H. Ecker, E. Pesch, G. Schmidt, and J. Weglarz (2013): *Scheduling computer and manufacturing processes*, springer science & Business media.

Blazewicz, J., J. K. Lenstra, and A. R. Kan (1983): "Scheduling subject to resource constraints: classification and complexity," *Discrete applied mathematics*, 5, 11–24.

Brown, G. G., S. Lawphongpanich, and K. P. Thurman (1994): "Optimizing ship berthing," *Naval Research Logistics (NRL)*, 41, 1–15.

Brucker, P. and S. Knust (2000): "Resource-constrained project scheduling and timetabling," in *International Conference on the Practice and Theory of Automated Timetabling*, Springer, 277–293.

Brucker, P., R. Qu, and E. Burke (2011): "Personnel scheduling: Models and complexity," *European Journal of Operational Research*, 210, 467–473.

Brunetta, L., L. Righi, and G. Andreatta (1999): "An operations research model for the evaluation of an airport terminal: SLAM (simple landside aggregate model)," *Journal of Air Transport Management*, 5, 161–175.

Brunner, J. O., J. F. Bard, and R. Kolisch (2009): "Flexible shift scheduling of physicians," *Health care management science*, 12, 285–305.

——— (2010): "Midterm scheduling of physicians with flexible shifts using branch and price," *Iie Transactions*, 43, 84–109.

Brunner, J. O. and G. M. Edenharter (2011): "Long term staff scheduling of physicians with different experience levels in hospitals using column generation," *Health care management science*, 14, 189–202.

Brunner, J. O. and R. Stolletz (2014): "Stabilized branch and price with dynamic parameter updating for discontinuous tour scheduling," *Computers & operations research*, 44, 137–145.

Bruno, G., A. Diglio, A. Genovese, and C. Piccolo (2018): "A decision support system to improve performances of airport check-in services," *Soft Computing*, 1–10.

Bruno, G. and A. Genovese (2010): "A mathematical model for the optimization of the airport check-in service problem," *Electronic Notes in Discrete Mathematics*, 36, 703–710.

Castillo-Salazar, J. A., D. Landa-Silva, and R. Qu (2016): "Workforce scheduling and routing problems: literature survey and computational study," *Annals of Operations Research*, 239, 39–67.

Catalão, J. P., H. M. Pousinho, and V. M. Mendes (2012): "Optimal offering strategies for wind power producers considering uncertainty and risk," *IEEE Systems Journal*, 6, 270–277.

Cheng, S. W., Y. P. Zhang, and Y. Y. Guo (2012): "Theory of Allocating and Scheduling Resources at Airport Passenger Terminals: A Review," in *Advanced Engineering Forum*, Trans Tech Publ, vol. 5, 66–70.

Chun, H. W. (1996): "Scheduling as a multi-dimensional placement problem," *Engineering Applications of Artificial Intelligence*, 9, 261–273.

Chun, H. W. and R. W. T. Mak (1999): "Intelligent resource simulation for an airport check-in counter allocation system," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 29, 325–335.

Cordeau, J.-F., M. Gaudioso, G. Laporte, and L. Moccia (2007): "The service allocation problem at the Gioia Tauro maritime terminal," *European Journal of Operational Research*, 176, 1167–1184.

Cordeau, J.-F., G. Laporte, P. Legato, and L. Moccia (2005): "Models and tabu search heuristics for the berth-allocation problem," *Transportation science*, 39, 526–538.

Correcher, J. F. and R. Alvarez-Valdes (2017): "A biased random-key genetic algorithm for the time-invariant berth allocation and quay crane assignment problem," *Expert Systems with Applications*, 89, 112–128.

Correcher, J. F., R. Alvarez-Valdes, and J. M. Tamarit (2019): "New exact methods for the time-invariant berth allocation and quay crane assignment problem," *European Journal of Operational Research*, 275, 80–92.

Daganzo, C. F. (1989): "The crane scheduling problem," *Transportation Research Part B: Methodological*, 23, 159–175.

Dantzig, G. B. (1954): "Letter to the editor—A comment on Edie's "Traffic delays at toll booths"," *Journal of the Operations Research Society of America*, 2, 339–341.

Dantzig, G. B. and P. Wolfe (1960): "Decomposition principle for linear programs," *Operations research*, 8, 101–111.

——— (1961): "The decomposition algorithm for linear programs," *Econometrica: Journal of the Econometric Society*, 767–778.

Desrochers, M. and F. Soumis (1989): "A column generation approach to the urban transit crew scheduling problem," *Transportation Science*, 23, 1–13.

Di Martinelly, C., P. Baptiste, and M. Maknoon (2014): "An assessment of the integration of nurse timetable changes with operating room planning and scheduling," *International Journal of Production Research*, 52, 7239–7250.

Diabat, A. and E. Theodorou (2014): "An integrated quay crane assignment and scheduling problem," *Computers & Industrial Engineering*, 73, 115–123.

Dijk, N. M. v. and E. v. d. Sluis (2006): "Check-in computation and optimization by simulation and IP in combination," *European Journal of Operational Research*, 171, 1152–1168.

Du, J. and J. Y.-T. Leung (1989): "Complexity of scheduling parallel task systems," *SIAM Journal on Discrete Mathematics*, 2, 473–487.

Du Merle, O., D. Villeneuve, J. Desrosiers, and P. Hansen (1999): "Stabilized column generation," *Discrete Mathematics*, 194, 229–237.

Duin, C. W. and E. v. d. Sluis (2006): "On the Complexity of Adjacent Resource Scheduling," *Journal of Scheduling*, 9, 49–62.

Erhard, Melale, C. and M. Vanhoucke (2018): "State of the art in physician scheduling," *Health Systems*, 4, 1–18.

Ernst, A. T., H. Jiang, M. Krishnamoorthy, B. Owens, and D. Sier (2004a): "An annotated bibliography of personnel scheduling and rostering," *Annals of Operations Research*, 127, 21–144.

Ernst, A. T., H. Jiang, M. Krishnamoorthy, and D. Sier (2004b): "Staff scheduling and rostering: A review of applications, methods and models," *European Journal of Operational Research*, 153, 3–27.

——— (2004c): "Staff scheduling and rostering: A review of applications, methods and models," *European journal of operational research*, 153, 3–27.

Fayez, M., A. Kaylani, D. Cope, N. Rychlik, and M. Mollaghasemi (2008): "Managing airport operations using simulation," *Journal of Simulation*, 2, 41–52.

Felix, M. and V. Reis (2017): "A hybrid discrete-event and an agent-based simulation model for assessing the performance of the check-in areas in airports," *EUROPEAN TRANSPORT-TRASPORTI EUROPEI*.

Florian, M., J. K. Lenstra, and A. Rinnooy Kan (1980): "Deterministic production planning: Algorithms and complexity," *Management science*, 26, 669–679.

Geibinger, T., F. Mischek, and N. Musliu (2019): "Investigating constraint programming for real world industrial test laboratory scheduling," in *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, Springer, 304–319.

Giallombardo, G., L. Moccia, M. Salani, and I. Vacca (2010): "Modeling and solving the tactical berth allocation problem," *Transportation Research Part B: Methodological*, 44, 232–245.

Gielen, D., F. Boshell, D. Saygin, M. D. Bazilian, N. Wagner, and R. Gorini (2019): "The role of renewable energy in the global energy transformation," *Energy Strategy Reviews*, 24, 38–50.

Goiri, Í., M. E. Haque, K. Le, R. Beauchea, T. D. Nguyen, J. Guitart, J. Torres, and R. Bianchini (2015): "Matching renewable energy supply and demand in green datacenters," *Ad Hoc Networks*, 25, 520–534.

Goiri, Í., K. Le, M. E. Haque, R. Beauchea, T. D. Nguyen, J. Guitart, J. Torres, and R. Bianchini (2011): "Greenslot: scheduling energy consumption in green datacenters," in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, 1–11.

Goldberg, D. E. (1989): *Genetic Algorithms in Search, Optimization and Machine Learning*, Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1st ed.

Guan, Y. and R. K. Cheung (2004): "The berth allocation problem: models and solution methods," *OR spectrum*, 26, 75–92.

Guan, Y., K.-H. Yang, and Z. Zhou (2013): "The crane scheduling problem: models and solution approaches," *Annals of Operations Research*, 203, 119–139.

Hartmann, S. and D. Briskorn (2010): "A survey of variants and extensions of the resource-constrained project scheduling problem," *European Journal of operational research*, 207, 1–14.

Hsu, C.-I. and C.-C. Chao (2005): "Scheduling Purchase and Renewal of International Airport Departure Facilities," *Journal of the Eastern Asia Society for Transportation Studies*, 6, 736–751.

Hsu, C.-I., C.-C. Chao, and K.-Y. Shih (2012): "Dynamic allocation of check-in facilities and dynamic assignment of passengers at air terminals," *Computers & Industrial Engineering*, 63, 410–417.

Huele, V. (2015): "Decomposition-based heuristics for the integrated physician rostering and surgery scheduling problem," 159–175.

Hwang, T.-L., C.-R. Jeng, and S.-S. Wang (2012): "Airport check-in counter assignment: a proposed solution," *International Journal of Aviation Management*, 1, 257–270.

IATA (2018): "IATA Forecast Predicts 8.2 Billion Air Travellers in 2037," https://www.iata.org/pressroom/pr/Pages/2018-10-24-02.aspx, accessed: 2018-11-1.

IATA-ADRM (2014): *Airport Development Reference Manual, 10th Edition*, International Air Transport Association.

Icmeli, O., S. S. Erenguc, and C. J. Zappe (1993): "Project scheduling problems: a survey," *International Journal of Operations & Production Management*.

Imai, A., H. C. Chen, E. Nishimura, and S. Papadimitriou (2008): "The simultaneous berth and quay crane allocation problem," *Transportation Research Part E: Logistics and Transportation Review*, 44, 900–920.

Imai, A., K. Nagaiwa, and C. W. Tat (1997): "Efficient planning of berth allocation for container terminals in Asia," *Journal of Advanced transportation*, 31, 75–94.

Imai, A., E. Nishimura, M. Hattori, and S. Papadimitriou (2007a): "Berth allocation at indented berths for mega-containerships," *European Journal of Operational Research*, 179, 579–593.

Imai, A., E. Nishimura, and S. Papadimitriou (2001): "The dynamic berth allocation problem for a container port," *Transportation Research Part B: Methodological*, 35, 401–417.

——— (2003): "Berth allocation with service priority," *Transportation Research Part B: Methodological*, 37, 437–457.

Imai, A., X. Sun, E. Nishimura, and S. Papadimitriou (2005): "Berth allocation in a container port: using a continuous location space approach," *Transportation Research Part B: Methodological*, 39, 199–221.

Imai, A., J.-T. Zhang, E. Nishimura, and S. Papadimitriou (2007b): "The berth allocation problem with service time and delay time objectives," *Maritime Economics & Logistics*, 9, 269–290.

International Chamber of Shipping (2020): "Shipping and World Trade," https://www.ics-shipping.org/shipping-facts/shipping-and-world-trade, accessed on 2020-7-1.

International Civil Aviation Organization (2017): "The World of Air Transport in 2017, Annual Report 2017, institution = International Civil Aviation Organization, A UN Specialized Agency, howpublished = https://www.icao.int/annual-report-2017/Pages/the-world-of-air-transport-in-2017.aspx,," .

International Energy Agency (2019): "India 2020, Energy Policy Review," https://niti.gov.in/sites/default/files/2020-01/IEA-India%202020-In-depth-EnergyPolicy_0.pdf, accessed On: 2020-4-10.

International Energy Agency, Paris (2019): "Tracking Industry, May 2019," https://www.iea.org/reports/tracking-industry, accessed On: 2020-4-10.

Iris, Ç., D. Pacino, and S. Ropke (2017): "Improved formulations and an adaptive large neighborhood search heuristic for the integrated berth allocation and quay crane assignment problem," *Transportation Research Part E: Logistics and Transportation Review*, 105, 123–147.

Iris, Ç., D. Pacino, S. Ropke, and A. Larsen (2015): "Integrated berth allocation and quay crane assignment problem: Set partitioning models and computational results," *Transportation Research Part E: Logistics and Transportation Review*, 81, 75–97.

Jacobs, L. W. and S. E. Bechtold (1993): "Labor utilization effects of labor scheduling flexibility alternatives in a tour scheduling environment," *Decision Sciences*, 24, 148–166.

Jacobs, L. W. and M. J. Brusco (1996): "Overlapping start-time bands in implicit tour scheduling," *Management Science*, 42, 1247–1259.

Jarrah, A. I., J. F. Bard, and A. H. deSilva (1994): "Solving large-scale tour scheduling problems," *Management Science*, 40, 1124–1144.

Joustra, P. E. and N. M. Van Dijk (2001): "Simulation of check-in at airports," in *Proceedings of the 33nd conference on Winter simulation*, IEEE Computer Society, 1023–1028.

Khatod, D. K., V. Pant, and J. Sharma (2013): "Evolutionary programming based optimal placement of renewable distributed generators," *IEEE Transactions on Power Systems*, 28, 683–695.

Kim, G., J. Kim, and J. Chae (2017): "Balancing the baggage handling performance of a check-in area shared by multiple airlines," *Journal of Air Transport Management*, 58, 31–49.

Kim, K. and S. Mehrotra (2015): "A two-stage stochastic integer programming approach to integrated staffing and scheduling with application to nurse management," *Operations Research*, 63, 1431–1451.

Kim, K. H. and K. C. Moon (2003): "Berth scheduling by simulated annealing," *Transportation Research Part B: Methodological*, 37, 541–560.

Kim, K. H. and Y.-M. Park (2004): "A crane scheduling method for port container terminals," *European Journal of operational research*, 156, 752–768.

Kiran, A. S., T. Cetinkaya, and S. Og (2000): "Simulation modeling and analysis of a new international terminal," in *Proceedings of the 32nd conference on Winter simulation*, Society for Computer Simulation International, 1168–1172.

Kiyildi, R. K. and M. Karasahin (2008): "The capacity analysis of the check-in unit of Antalya Airport using the fuzzy logic method," *Transportation Research Part A: Policy and Practice*, 42, 610–619.

Kolen, A. W., J. K. Lenstra, C. H. Papadimitriou, and F. C. Spieksma (2007): "Interval scheduling: A survey," *Naval Research Logistics (NRL)*, 54, 530–543.

Kolisch, R. and R. Padman (2001): "An integrated survey of deterministic project scheduling," *Omega*, 29, 249–272.

Korpaas, M., A. T. Holen, and R. Hildrum (2003): "Operation and sizing of energy storage for wind power plants in a market system," *International Journal of Electrical Power & Energy Systems*, 25, 599–606.

Lai, K. and K. Shih (1992): "A study of container berth allocation," *Journal of advanced transportation*, 26, 45–60.

Lalita, T. R., D. K. Manna, and G. S. R. Murthy (2020): "Mathematical formulations for large scale check-in counter allocation problem," *Journal of Air Transport Management*, 85, 101796.

Lalita, T. R. and G. S. R. Murthy (2020a): "An Efficient Algorithm to the Integrated Shift and Task Scheduling Problem," *Technical Report, SQC and OR Division.*

——— (2020b): "The Wind Power Scheduling Problem," *Technical Report-SQCOR-2020-05.*

——— (2021a): "Day Ahead Wind Power Forecast Data," http://dx.doi.org/10.17632/3t5ns4x58r.11.

——— (2021b): "Staff Requirement Data for Staff Scheduling," http://dx.doi.org/10.17632/bk33vpzvkd.1.

Lau, H. C. (1996): "On the complexity of manpower shift scheduling," *Computers & Operations Research*, 23, 93–102.

Li, C., A. Qouneh, and T. Li (2012): "iSwitch: coordinating and optimizing renewable energy powered server clusters," *ACM SIGARCH Computer Architecture News*, 40, 512–523.

Liang, R.-H. and J.-H. Liao (2007): "A fuzzy-optimization approach for generation scheduling with wind and solar energy systems," *IEEE Transactions on Power Systems*, 22, 1665–1674.

Liberatore, F., G. Righini, and M. Salani (2011): "A column generation algorithm for the vehicle routing problem with soft time windows," *4OR*, 9, 49–82.

Lim, A. (1998): "The berth planning problem," *Operations research letters*, 22, 105–110.

Lim, A., B. Rodrigues, F. Xiao, and Y. Zhu (2004): "Crane scheduling with spatial constraints," *Naval Research Logistics (NRL)*, 51, 386–406.

Lin, D., Z. Xin, and Y. Huang (2015): "Ground crew rostering for the airport check-in counter," in *Industrial Engineering and Engineering Management (IEEM), 2015 IEEE International Conference on*, IEEE, 1462–1466.

Lin, Y.-H. and C.-F. Chen (2013): "Passengers' shopping motivations and commercial activities at airports–The moderating effects of time pressure and impulse buying tendency," *Tourism Management*, 36, 426–434.

Liu, J., Y.-w. Wan, and L. Wang (2006): "Quay crane scheduling at container terminals to minimize the maximum relative tardiness of vessel departures," *Naval Research Logistics (NRL)*, 53, 60–74.

Liu, Y., J. Wang, and S. Shahbazzade (2019): "The improved AFSA algorithm for the berth allocation and quay crane assignment problem," *Cluster Computing*, 22, 3665–3672.

Lodi, A., S. Martello, and M. Monaci (2002): "Two-dimensional packing problems: A survey," *European journal of operational research*, 141, 241–252.

Lous, C. (2011): "Modelling and optimization of allocation of check-in counters in an airport," B.S. thesis, Technical University of Denmark, DTU, DK-2800 Kgs. Lyngby, Denmark.

Lübbecke, M. E. and J. Desrosiers (2005): "Selected topics in column generation," *Operations research*, 53, 1007–1023.

Maenhout, B. and M. Vanhoucke (2013): "An integrated nurse staffing and scheduling analysis for longer-term nursing staff allocation problems," *Omega*, 41, 485–499.

——— (2016): "An exact algorithm for an integrated project staffing problem with a homogeneous workforce," *Journal of Scheduling*, 19, 107–133.

——— (2017): "A resource type analysis of the integrated project scheduling and personnel staffing problem," *Annals of Operations Research*, 252, 407–433.

Makarov, Y. V., P. V. Etingov, J. Ma, Z. Huang, and K. Subbarao (2011): "Incorporating uncertainty of wind power generation forecast into power system operation, dispatch, and unit commitment procedures," *IEEE Transactions on Sustainable Energy*, 2, 433–442.

Mauri, G. R., G. M. Ribeiro, L. A. N. Lorena, and G. Laporte (2016): "An adaptive large neighborhood search for the discrete and continuous berth allocation problem," *Computers & Operations Research*, 70, 140–154.

Meisel, F. and C. Bierwirth (2009): "Heuristics for the integration of crane productivity in the berth allocation problem," *Transportation Research Part E: Logistics and Transportation Review*, 45, 196–209.

——— (2013): "A framework for integrated berth allocation and crane operations planning in seaport container terminals," *Transportation Science*, 47, 131–147.

Ministry of Power, G. o. I. (2005): "National Electricity Policy," https://powermin.nic.in/en/content/national-electricity-policy, accessed On: 2018-05-07.

Moccia, L., J.-F. Cordeau, M. Gaudioso, and G. Laporte (2006): "A branch-and-cut algorithm for the quay crane scheduling problem in a container terminal," *Naval Research Logistics (NRL)*, 53, 45–59.

Monaco, M. F. and M. Sammarra (2007): "The berth allocation problem: a strong formulation solved by a Lagrangean approach," *Transportation Science*, 41, 265–280.

Moorthy, R. and C.-P. Teo (2007): "Berth management in container terminal: the template design problem," in *Container Terminals and Cargo Systems*, Springer, 63–86.

Mota, M. M. (2015): "Check-in allocation improvements through the use of a simulation–optimization approach," *Transportation Research Part A: Policy and Practice*, 77, 320–335.

Mota, M. M. and C. Z. Alcaraz (2015): "Allocation of Airport Check-in Counters Using a Simulation-Optimization Approach," in *Applied Simulation and Optimization*, Springer, 203–229.

Mota, M. M. and C. Zuniga (2013): "A simulation-evolutionary approach for the allocation of check-in desks in airport terminals." *ATOS 2013, 4th International Air Transport and Operations Symposium.*

Murthy, G. (2016): *Applications of Operations Research and Management Science*, Springer.

Neumann, K., C. Schwindt, and J. Zimmermann (2012): *Project scheduling with time windows and scarce resources: temporal and resource-constrained project scheduling with regular and nonregular objective functions*, Springer Science & Business Media.

Nikolaev, A. G., S. H. Jacobson, and L. A. McLay (2007): "A sequential stochastic security system design problem for aviation security," *Transportation Science*, 41, 182–194.

Nishimura, E., A. Imai, and S. Papadimitriou (2001): "Berth allocation planning in the public berth system by genetic algorithms," *European Journal of Operational Research*, 131, 282–292.

Oliveira, A. S., S. Urrutia, and J. Oppen (2016): "A decomposition approach to solve the quay crane scheduling problem," *arXiv*, 1604–00527.

Oukil, A., H. B. Amor, J. Desrosiers, and H. El Gueddari (2007): "Stabilized column generation for highly degenerate multiple-depot vehicle scheduling problems," *Computers & Operations Research*, 34, 817–834.

Pacheco, R. and E. Fernandes (2003): "Managerial efficiency of Brazilian airports," *Transportation Research Part A: Policy and Practice*, 37, 667–680.

Papadimitriou, C. H. and K. Steiglitz (1998): *Combinatorial optimization: algorithms and complexity*, Courier Corporation.

Parisio, A. and C. N. Jones (2015): "A two-stage stochastic programming approach to employee scheduling in retail outlets with uncertain demand," *Omega*, 53, 97–103.

Park, Y. and S. B. Ahn (2003): "Optimal assignment for check-in counters based on passenger arrival behaviour at an airport," *Transportation Planning and Technology*, 26, 397–416.

Parlar, M., B. Rodrigues, and M. Sharafali (2013): "On the allocation of exclusive-use counters for airport check-in queues: static vs. dynamic policies," *Opsearch*, 50, 433–453.

Parlar, M. and M. Sharafali (2008): "Dynamic allocation of airline check-in counters: a queueing optimization approach," *Management Science*, 54, 1410–1424.

Pinedo, M. (2002): *Scheduling: Theory, Algorithms and Systems, 2nd edn, 2002*, Prentice-Hall: Englewood Cliffs, NJ.

Pinedo, M. and X. Chao (1998): "Operations scheduling with applications in manufacturing and services, 1999," .

Raghavan, S. and D. Stanojević (2011): "Branch and price for WDM optical networks with no bifurcation of flow," *INFORMS Journal on Computing*, 23, 56–74.

Reddy, S. S. (2017): "Optimal scheduling of wind-thermal power system using clustered adaptive teaching learning based optimization," *Electrical Engineering*, 99, 535–550.

Rodič, B. and A. Baggia (2017): "Airport Ground Crew Scheduling Using Heuristics and Simulation," in *Applied Simulation and Optimization 2*, Springer, 131–160.

Ros Prat, I. (2017): "Improving check-in processing at Brisbane airport," Master's thesis, Universitat Politècnica de Catalunya.

Snowdon, J. L., S. El-Taji, M. Montevecchi, E. MacNair, C. A. Callery, and S. Miller (1998): "Avoiding the blues for airline travelers," in *Proceedings of the 30th conference on Winter simulation*, IEEE Computer Society Press, 1105–1112.

Snowdon, J. L., E. MacNair, M. Montevecchi, C. Callery, S. El-Taji, and S. Miller (2000): "IBM journey management library: an arena system for airport simulations," *Journal of the Operational Research Society*, 51, 449–456.

Soukour, A. A., L. Devendeville, C. Lucet, and A. Moukrim (2013): "A memetic algorithm for staff scheduling problem in airport security service," *Expert Systems with Applications*, 40, 7504–7512.

Statista (2020): "Container Shipping- Statistics and Facts," https://www.statista.com/topics/1367/container-shipping/), accessed on 2020-7-1.

Stolletz, R. (2010): "Operational workforce planning for check-in counters at airports," *Transportation Research Part E: Logistics and Transportation Review*, 46, 414–425.

Stolletz, R. and E. Zamorano (2014): "A rolling planning horizon heuristic for scheduling agents with different qualifications," *Transportation Research Part E: Logistics and Transportation Review*, 68, 39–52.

Su, W., J. Wang, and J. Roh (2014): "Stochastic energy scheduling in microgrids with intermittent renewable energy resources," *IEEE Transactions on Smart Grid*, 5, 1876–1883.

Sungur, B., C. Özgüven, and Y. Kariper (2017): "Shift scheduling with break windows, ideal break periods, and ideal waiting times," *Flexible Services and Manufacturing Journal*, 29, 203–222.

Takakuwa, S. and T. Oyama (2003): "Modeling people flow: simulation analysis of international-departure passenger flows in an airport terminal," in *Proceedings of the 35th conference on Winter simulation: driving innovation*, Winter Simulation Conference, 1627–1634.

Tanabe, T., T. Sato, R. Tanikawa, I. Aoki, T. Funabashi, and R. Yokoyama (2008): "Generation scheduling for wind power generation by storage battery system and meteorological forecast," in *Power and Energy Society General Meeting-Conversion and Delivery of Electrical Energy in the 21st Century, 2008 IEEE*, IEEE, 1–7.

Tang, C.-H. (2010): "A network model for airport common use check-in counter assignments," *Journal of the Operational Research Society*, 61, 1607–1618.

Theodorou, E. and A. Diabat (2015): "A joint quay crane assignment and scheduling problem: formulation, solution algorithm and computational results," *Optimization Letters*, 9, 799–817.

Thompson, G. M. (1995): "Improved implicit optimal modeling of the labor shift scheduling problem," *Management Science*, 41, 595–607.

Thompson, G. M. and M. E. Pullman (2007): "Scheduling workforce relief breaks in advance versus in real-time," *European Journal of Operational Research*, 181, 139–155.

Trakoonsanti, L. (2016): "A Process Simulation Model of Airline Passenger Check – In," .

Türkoğullari, Y. B., Z. C. Taşkin, N. Aras, and İ. K. Altinel (2014): "Optimal berth allocation and time-invariant quay crane assignment in container terminals," *European Journal of Operational Research*, 235, 88–101.

Ursavas, E. and S. X. Zhu (2016): "Optimal policies for the berth allocation problem under stochastic nature," *European Journal of Operational Research*, 255, 380–387.

Van den Bergh, J., J. Beliën, P. De Bruecker, E. Demeulemeester, and L. De Boeck (2013): "Personnel scheduling: A literature review," *European journal of operational research*, 226, 367–385.

Vanderbeck, F. (2000): "On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm," *Operations Research*, 48, 111–128.

——— (2011): "Branching in branch-and-price: a generic scheme," *Mathematical Programming*, 130, 249–294.

Vanderbeck, F. and L. A. Wolsey (1996): "An exact algorithm for IP column generation," *Operations research letters*, 19, 151–159.

Vanhoucke, M., E. Demeulemeester, and W. Herroelen (2002): "Discrete time/cost trade-offs in project scheduling with time-switch constraints," *Journal of the Operational Research Society*, 53, 741–751.

Venkatesh, B., T. Geetha, and V. Jayashankar (2011): "Frequency sensitive unit commitment with availability-based tariff: an Indian example," *IET generation, transmission & distribution*, 5, 798–805.

Volland, J., A. Fügener, and J. O. Brunner (2017a): "A column generation approach for the integrated shift and task scheduling problem of logistics assistants in hospitals," *European Journal of Operational Research*, 260, 316–334.

Volland, J., A. Fügener, J. Schoenfelder, and J. O. Brunner (2017b): "Material logistics in hospitals: a literature review," *Omega*, 69, 82–101.

Wang, F. and A. Lim (2007): "A stochastic beam search for the berth allocation problem," *Decision support systems*, 42, 2186–2196.

Wang, K., L. Zhen, S. Wang, and G. Laporte (2018): "Column generation for the integrated berth allocation, quay crane assignment, and yard assignment problem," *Transportation Science*, 52, 812–834.

Wang, Q., J. Wang, and Y. Guan (2013): "Price-based unit commitment with wind power utilization constraints," *IEEE Transactions on Power Systems*, 28, 2718–2726.

Wibowo, S. S. and S. R. Fadilah (2018): "Queuing analysis using Viswalk for check-in counter: Case study of Lombok Praya International Airport," in *MATEC Web of Conferences*, EDP Sciences, vol. 181, 02006.

Wilhelm, W. E. (2001): "A technical review of column generation in integer programming," *Optimization and Engineering*, 2, 159–200.

Wong, J.-T. and T. Liu (1998): "Development and application of an airport terminal simulation model—a case study of CKS airport," *Transportation Planning and Technology*, 22, 73–86.

Wu, P. P.-Y. and K. Mengersen (2013): "A review of models and model usage scenarios for an airport complex system," *Transportation Research Part A: Policy and Practice*, 47, 124–140.

Xie, F., T. Wu, and C. Zhang (2019): "A Branch-and-Price Algorithm for the Integrated Berth Allocation and Quay Crane Assignment Problem," *Transportation Science*, 53, 1427–1454.

Xin, Z., D. Lin, Y. Huang, W. Cheng, and C. Chong Teo (2014): "Design of service capacity for the ground crew at the airport check-in counters," *International Journal of Quality and Service Sciences*, 6, 43–59.

Yan, S., K.-C. Chang, and C.-H. Tang (2005): "Minimizing inconsistencies in airport common-use checking counter assignments with a variable number of counters," *Journal of Air Transport Management*, 11, 107–116.

Yan, S., C. Tang, and C. Chen (2008): "Reassignments of common-use check-in counters following airport incidents," *Journal of the Operational Research Society*, 59, 1100–1108.

Yan, S., C.-H. Tang, and J.-H. Chen (2014): "Common-use check-in counter reassignments with a variable number of service lines and variable length of time window," *Journal of the Chinese Institute of Engineers*, 37, 643–658.

Yan, S., C.-H. Tang, and M. Chen (2004): "A model and a solution algorithm for airport common use check-in counter assignments," *Transportation Research Part A: Policy and Practice*, 38, 101–125.

Yeung, B. K. W. and H. W. Chun (1995): "Check-in counter allocation using genetic algorithm," .

Zamorano, E., A. Becker, and R. Stolletz (2018): "Task assignment with start time-dependent processing times for personnel at check-in counters," *Journal of Scheduling*, 21, 93–109.

Zhen, L. (2015): "Tactical berth allocation under uncertainty," *European Journal of Operational Research*, 247, 928–944.

Zhen, L., L. H. Lee, and E. P. Chew (2011): "A decision model for berth allocation under uncertainty," *European Journal of Operational Research*, 212, 54–68.