University of Thessaly
Department of Mechanical Engineering
Systems Optimization Laboratory

Set-partition versus set-cover formulations for column generation based solution methodologies for optimal crew scheduling in the airline industry

Author
Odysseas C.M. Moschopoulos

Supervisor
Prof. George Kozanidis

Submitted in partial fulfillment of
the requirements for the Diploma
of Mechanical Engineering

Volos 2018

Thesis Committee

1st Member (Supervisor):

Assistant Professor George Kozanidis
Department of Mechanical Engineering, University of Thessaly

2nd Member:

Associate Professor Dimitris Pandelis
Department of Mechanical Engineering, University of Thessaly

3rd Member:

Assistant Professor Georgios Saharidis
Department of Mechanical Engineering, University of Thessaly

Ευχαριστίες/**Acknowledgements**

Θα ήθελα καταρχάς να ευχαριστήσω θερμά τον καθηγητή κ. Κοζανίδη για την υποστήριξη και την καθοδήγησή του κατά τη συγγραφή της παρούσας διπλωματικής εργασίας, αλλά και για τις ευκαιρίες που μου προσέφερε σε ακαδημαικό επίπεδο. Μου έδειξε εμπιστοσύνη και μου μετέδωσε πολύτιμες γνώσεις, τόσο κατά τη δημιουργία της διπλωματικής μου εργασίας, όσο και τα προηγούμενα χρόνια κατά τη διδασκαλία των μαθημάτων του.

Θα ήθελα επίσης να ευχαριστήσω τους καθηγητές κ. Παντελή και κ. Σαχαρίδη για την παρουσία τους στην εξεταστική επιτροπή της διπλωματικής μου αλλά κυρίως για τις γνώσεις που μου έδωσαν κατά τη διάρκεια της φοίτησής μου στο τμήμα. Ο ζήλος και το ενδιαφέρον που επιδείκνυαν για το επιστημονικό τους αντικείμενο και το διδακτικό τους έργο με ενέπνευσαν να επενδύσω πραγματικά στις σπουδές μου.

Θα ήθελα επιπλέον να ευχαριστήσω τους κύριους καθηγητές για την πολύτιμη υποστήριξη που μου παρείχαν για το επόμενο βήμα στην ακαδημαική μου ζωή, καθώς έδωσαν με προθυμία τις συστάσεις τους, αλλά και για τις άψογες σχέσεις μεταξύ μας, που έκαναν την καθημερινότητά μου στην σχολή, εκτός από εποικοδομητική, και ευχάριστη.

Στην συνέχεια, θα ήθελα να ευχαριστήσω πολύ τους φίλους μου, για όλες τις όμορφες στιγμές που περάσαμε κατά τα χρόνια των σπουδών μας αλλά και για την υποστήριξη που μου παρείχαν στις δύσκολες στιγμές. Αισθάνομαι πολύ τυχερός που τους γνώρισα και πιστεύω πως η φιλία μας θα αντέξει στον χρόνο και στην απόσταση. Η συντροφιά τους έδωσε χρώμα στα φοιτητικά μου χρόνια και η συναναστροφή μου μαζί τους με έκανε καλύτερο άνθρωπο.

Θα ήθελα τέλος να ευχαριστήσω τους δικούς μου ανθρώπους, χωρίς τους οποίους δεν θα μπορούσα να φτάσω στην ολοκλήρωση των σπουδών μου. Η υποστήριξη των παππούδων και των γιαγιάδων μου ήταν πολύ σημαντική και μου επέτρεψε να αφοσιωθώ πλήρως στις σπουδές μου. Μου πρόσφεραν πολλά από τα λίγα που είχαν και αυτό είναι κάτι που δεν θα ξεχάσω ποτέ.

Θα ήθελα επίσης να ευχαριστήσω πολύ την αδερφή μου, Ιφιγένεια, για την υπομονή της κατά τη συγκατοίκησή μας όλα αυτά τα χρόνια και την υποστήριξή της σε περιόδους πίεσης. Από την στιγμή που ήρθε στον Βόλο, η παρέα της βελτίωσε πολύ την διάθεσή μου και είχε θετικό αντίκτυπο στην καθημερινότητά μου. Η παρουσία της θα μου λείψει.

Τέλος, θα ήθελα να ευχαριστήσω τους γονείς μου, Χρήστο και Μαρία, για τις θυσίες τους, που μου έδωσαν τη δυνατότητα να σπουδάσω, να αποκτήσω τα εφόδια για μια αξιοπρεπή ζωή. Στερήθηκαν πολλά για να τα προσφέρουν σε εμένα και την αδερφή μου και για αυτό θα τους είμαι για πάντα ευγνώμων. Τους ευχαριστώ για τις αξίες που μου δίδαξαν, οι οποίες με συντρόφευαν κατά τα χρόνια των σπουδών μου. Πιστεύω πως αυτές είχαν σπουδαίο ρόλο στις επιτυχίες μου. Όμως, σημαντικότερο είναι το γεγονός πως με κράτησαν όρθιο στις αποτυχίες και μου επέτρεψαν να συνεχίσω. Χωρίς τους γονείς μου δεν θα κατάφερνα πολλά.

Η παρούσα διπλωματική εργασία είναι αφιερωμένη στην μητέρα μου, Μαρία, και στον πατέρα μου, Χρήστο.

Abstract

In the present thesis, we consider column generation based solution methodologies utilized widely in the context of airline management for crew scheduling optimization. Such methodologies employ a master problem typically formulated as a set-partition model, which optimizes the total schedule cost comprised of the cost of uncovered duties and of the crew roster quality cost. We propose an alternative formulation of the master problem as a set-cover optimization model that intentionally ignores the quality cost of the crew-schedules. This could help crew planners discover the optimal flight duty coverage much faster and subject to smaller margin of error, which is of great importance in the airline industry. We transform the (possibly over-covering) set-cover solution into an equivalent set-partition one using an optimization model that suitably removes pairings from the existing crew rosters, so as to minimize the final quality cost while also keeping the optimal coverage. We present computational results from tests conducted in realistic environments that show the comparative performance of the alternative formulations and we present the conditions under which the utilization of each formulation is more preferable.

Περίληψη

Στην παρούσα διπλωματική εργασία, εξετάζουμε μεθοδολογίες column generation οι οποίες εφαρμόζονται ευρέως στο χώρο των αερομεταφορών για βέλτιστο χρονοπρογραμματισμό ιπτάμενου προσωπικού. Χρησιμοποιώντας ένα master πρόβλημα που μορφοποιείται ως ένα μοντέλο βελτιστοποίησης set-partition, οι μεθοδολογίες αυτές στοχεύουν στην ελαχιστοποίηση του συνολικού κόστους προγραμματισμού, το οποίο εκφράζεται ως το άθροισμα του κόστους των ακάλυπτων πτητικών διαδρομών και του κόστους της ποιότητας των προγραμμάτων του προσωπικού. Προτείνουμε μια εναλλακτική μορφοποίηση του master προβλήματος ως ένα μοντέλο βελτιστοποίησης set-cover, το οποίο σκόπιμα αγνοεί το κόστος ποιότητας των προγραμμάτων του προσωπικού. Αυτό θα μπορούσε να βοηθήσει τον χρήστη να εκτιμήσει τη μέγιστη δυνατή κάλυψη πτητικών διαδρομών που μπορεί να επιτευχθεί με συγκεκριμένο σύνολο ιπταμένων πολύ ταχύτερα και με μικρότερο περιθώριο σφάλματος, κάτι που είναι υψίστης σημασίας για τη συγκεκριμένη εφαρμογή. Μετατρέπουμε τη set-cover λύση, στην οποία πιθανόν υπάρχει υπέρ-κάλυψη, σε μια αντίστοιχη λύση set-partition με τη χρήση ενός μοντέλου βελτιστοποίησης που αφαιρεί, με κατάλληλο τρόπο, πτητικές διαδρομές από τα υφιστάμενα προγράμματα του προσωπικού έτσι ώστε να ελαχιστοποιείται το τελικό κόστος ποιότητας διατηρώντας τη βέλτιστη κάλυψη. Παρουσιάζουμε πειραματικά αποτελέσματα από δοκιμές που διεξήχθησαν σε ρεαλιστικά προβλήματα, τα οποία καταδεικνύουν τη συγκριτική απόδοση των δύο εναλλακτικών μορφοποιήσεων και καταγράφουμε τις προϋποθέσεις κάτω από τις οποίες κάθε μία από τις δύο είναι καταλληλότερη.

Contents

List of Figures

Chapter 1

Introduction

1.1 General

The field of operations research has had a tremendous impact on the management of today's airline industry. Driven by enormous demand from management to gain a competitive advantage in the market, airlines are turning to advanced optimization techniques to develop elaborate decision support systems for management and control of their operations.

The product offered by airline firms is flights that carry passengers or cargo from one place to another. In general, their success is based on the timeliness, accuracy, quality and price of their services. These criteria translate into flexible schedules, on-time flights, safety, satisfactory in-flight services and convenient ticket purchases for air transportation customers. The basic resources that are needed to operate an airline are aircraft, both ground and air crew and airport facilities such as runways and gates.

The air transportation market is very competitive. The competition is not restricted only within the airline industry, but also includes ground transportation means such as trains, buses and ships.

In order to provide a product with high quality and low cost, which is imperative for their survival and growth, airlines spend a tremendous amount of resources and effort, channeling it towards scientific management, to generate profitable and cost-effective fare classes, flight schedules, aircraft routes and maintenance schedules, crew pairings and rosters, gate assignments and others.[25]

1.2 The History of the Airline Industry

To better illustrate the indispensability of scientific management in the airline industry, a presentation of the industry's history and of the most significant events that led to its present role and size is necessary.

The history of the airline industry can be divided in two parts. In the first part, beginning in 1903 with the first powered flight of a heavier-than-air machine, the shaping force of the industry was technological advancement. In the second part, starting in the 70s, with the Airline Deregulation Act in 1978 being its highlight, the factor that determined the course of the industry globally was regulation, or rather the withdrawal of it, in the absence of a major technological breakthrough. These two periods led us to the present, when air transportation of people and goods is of paramount importance and a major part of our lives. The two following sections are based on [22].

1.2.1 Era of Technological Breakthroughs (1903-1970)

On December 17, 1903, the Wright brothers, after four years of relentless research and design efforts, flew their aircraft for 12 seconds, covering a distance of 37 meters. At that historic moment they conducted the first powered flight of a heavier-than-air machine.

With the outbreak of World War I, the military value of aircraft was quickly recognized and production increased to meet the rising demand for fighting aircraft. A significant advancement probed by WWI was the development of more powerful engines, which enabled aircraft to reach speeds of up to 210 kilometers per hour, more than twice the speed of pre-war aircraft. Increased power made the construction of larger aircraft feasible.

Following the World War I, people tried to utilize airplanes for civilian purposes. Some European countries, such as Great Britain and France, supported commercial aviation by starting air service over the English Channel. In 1917, the U.S. government commenced the transport of mail by air. By the mid-1920s, the Post Office mail fleet was flying around 4 million kilometers and delivering 14 million letters annually.

However, an event in 1927 brought unprecedented public attention to aviation in America and helped secure the industry's future as a major mode of transportation. A pilot named Charles Lindbergh conducted a historic flight across the Atlantic Ocean, from New York to Paris. It was the first solo, nonstop transatlantic flight in an airplane. Nevertheless, others had already crossed the Atlantic, though not alone. During 14–15 June 1919, the British pilots Brown and Alcock had made the first nonstop transatlantic flight. Accomplishments like these had an enormous effect on aviation, since it became a more established industry, attracting great amounts of private investment, as well as the support of millions of people.

Radio was also a development of enormous importance to aviation. Airlines used it to transmit weather information from the ground to their pilots so they could avoid storms. The first air traffic control tower, utilizing radio technology, was built in 1935.

Another breakthrough was the cabin pressurization, introduced in 1940. Aircraft that featured this technology allowed airlines to fly higher, in order to get above the air turbulence and storms common at lower altitudes. Prior to its development, airplanes could fly no higher than 3000 meters because people became dizzy and even fainted due to low levels of oxygen at higher altitudes. But with this novel technology, air was pumped into the plane's body as it gained altitude to maintain an atmosphere inside the cabin similar to the atmosphere that occurs naturally at sea level. With a regulated air compressor, airplanes could fly as high as 6000 meters.

Aviation had an enormous impact on the course of World War II and the war had just as significant an impact on aviation. During the years of the WWII, what is considered the most important technological achievement for the aviation industry was invented. The jet engine. This innovative system enabled the development of larger and faster aircraft and is incorporated predominantly in the modern aircraft up until today. Another great invention of this time that facilitated the operation of the growing airline industry of the coming years was the radar.

1958 marked the beginning of the Jet Age, as the first commercially successful passenger jet, the Boeing 707, was manufactured. With a length of 125 feet and four engines, this iconic aircraft could carry up to 181 passengers and travel at speeds of 880 kilometers per hour. Its engines proved more reliable than piston engines – producing less vibration, putting less stress on the plane's frame, thus reducing maintenance expenses substantially. They also burned kerosene, which cost half as much as the high-octane gasoline used in other planes of that time.

Finally, in the year 1969 another revolutionary aircraft, the Boeing 747, made its debut. It was the first wide-body jet, with four engines, two aisles in its main deck cabin and a separate upper deck over the front section of the aircraft's fuselage. Able to transfer as many as 450 passengers, it was 80 percent bigger than the largest jet up until that time, enabling the realization of economies of scale, which was a major disruption for the airline industry at that time but also largely shaped its future. Although airplane systems have improved since then, the wide-body aircraft was the last game-changing technological innovation in the industry.

1.2.2 Era of Regulatory Changes (1970-2000)

At that point, airplanes were larger, faster and safer than ever before, able to transport hundreds of people in a single flight from one continent to the other. Although technology was advanced enough, there were other hindrances to the further growth of the airline industry.

All over the world, the industry was in the hands of the governments. Government agencies determined the routes each airline flew and the fares that it could charge and thus the industry resembled a public utility.

However the situation was about to change. In America, pressure for airline deregulation had been building for many years. In particular there were many economists who pointed out that the liberalization of the air transport market would largely benefit passengers. Nevertheless, it was a series of developments in the early 70s that intensified the pressure and resulted in changes. One of those developments was the introduction of wide-body aircraft, which significantly increased airline capacity on many flight routes, making it very hard for airlines to cover the cost of extra seats in the market without having the ability to adjust fare prices. Another was the embargo on Arab oil in 1973, which led to a surge of fuel costs. These events put great pressure on the airlines as passenger demand fell sharply while capacity and fuel prices increased.

Finally, after years of effort, in 1978, the Airline Deregulation Act certified that route and rate restrictions would be steadily eliminated. The same occurred in Europe in a decade-long process starting in 1986. Several sets of EU regulations gradually turned, the once protected national aviation markets into a competitive single market for air transport.[24]

The effects of deregulation were huge and changed the industry to its core. A major development was the widespread utilization of hub-and-spoke networks. Hubs are strategically located airports. Airlines schedule batches of flights into and out of these transfer points several times a day. Each batch includes many planes arriving at the airport within minutes of each other. Once on the ground, the arriving passengers and cargo from those flights are transferred to other planes, which are about to depart shortly, that will take them to their final destinations. Airlines developed hub-and-spoke networks in order to efficiently serve substantially more markets with a given fleet size than they could if they only offered direct flights. Air carriers also found that hub-and-spoke systems allowed them to achieve higher load factors (i.e. percentage of seats filled) on flights to and from small cities, in which they could not operate profitably previously.

Furthermore, deregulation opened the airline business to new firms. It is remarkable that, in 1978, there were roughly 40 carriers with large aircraft in the U.S.A.. In 2005, this number increased to around 140. Since the early 90s, there has been a wave of new airlines operating different business models with the most notable being the low-cost model. The appearance of new firms, combined with the strategic expansion into new markets by many of the established airlines, resulted in unprecedented competition in the industry. Today, the majority of airline passengers have many options for their journey, whereas in the past the options were often limited to one operator.

Increased competition resulted in discounts, which is the greatest benefit experienced by customers caused by airline deregulation. Ticket prices fell steadily over the years, finally becoming so low that, in some cases, even bus and rail services have to compete with airlines.

1.2.3 Aviation: A Major Contributor to Global Prosperity (2000-Present)

With greater competition on the vast majority of flight routes, extensive discounting and more available flights, air travel has grown rapidly since its deregulation (see Figures 1.2 and 1.1).

Its tremendous growth benefited the world. Aviation provides the only rapid world-wide transportation system, which makes it imperative for global business. It generates economic growth, creates jobs and supports international trade and tourism.

According to recent estimates by the Air Transport Action Group (ATAG), the total economic impact of the global aviation industry reached USD 2.7 trillion, which is roughly 3.5 per cent of world's gross domestic product (GDP) in 2014. The air transport industry also supported millions jobs globally, providing around 10 million jobs.

One of the industries that relies most heavily on aviation is tourism. By facilitating tourism, air transport helps generate economic growth and alleviate poverty. Currently, approximately 1.2 billion tourists are traveling from one country to another every year. Over half of them use airplanes to travel to their destinations.

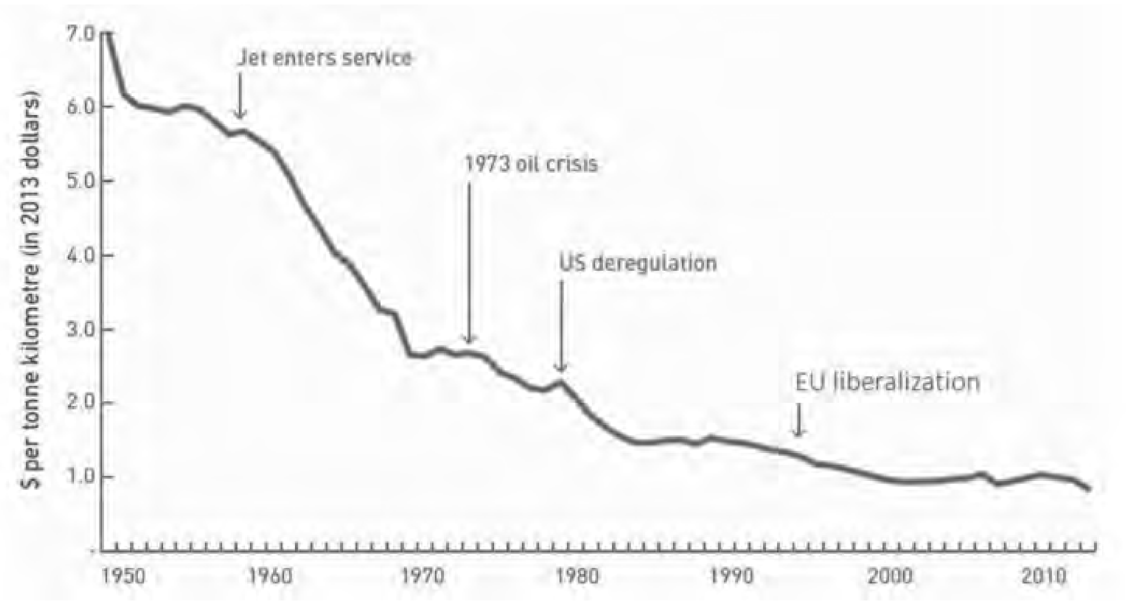


Figure 1.1: Evolution of Average Price of Air Travel. (Source: IATA Economics)

Air transportation is a driver of global trade and e-commerce, making possible the globalization of production. Aviation's advantage over other modes of transport in terms of speed and reliability allowed rapid delivery services and transportation of urgent or time-sensitive goods, such as food or medicine. Roughly 87 per cent of business-to-consumer e-commerce packages are currently transported by air. The utilization of airplanes by the e-commerce industry is anticipated to continue to grow.

The reliability of available air transport services provides people with access to decent lives, food, healthcare and education. Aviation is the world's safest and most efficient means to travel long distances. In same cases, it provides the only possible mode of transportation to provide health care to remote places and to deliver urgent humanitarian aid during emergencies caused by natural disasters and war. Moreover, educational opportunities are made available to students around the world, especially for those from developing countries who must travel abroad to gain access to higher quality education. Aviation also contributes to improving the life quality of people by giving them access to more leisure and cultural experiences. Last but not least, it supports awareness of other cultures.[12]

However, as its size and role grew many challenges appeared. The airlines' business became very complex. This complexity can be highlighted via the following points:

• As a result of economic considerations when generating the plans and schedules, all the resources are tightly utilized. As a consequence, little space is left to maneuver and to respond to disruptions that may occur.

• The working environment of airlines is very dynamic and uncertain. Many causes may lead to disruption of an original plan, resulting in irregular operations. The most common reasons behind such disruptions typically are bad weather, aircraft mechanical failures, crew sickness and even fuel shortages. What makes things even worse is that a small change in one place will affect the whole system.

• Large sets of strict rules governing the aircraft maintenance, crew schedules, runway availability, security issues and others, add many more restrictions and complicate even more the operation of an airline.

• Finally, the large operational scale of the airlines in terms of the number of different resources involved, such as aircraft, crew members and ground facillities, the large geographic regions serviced and the sheer number of passengers transferred, render the solution of airlines' operational and planning problems a daunting task.[25]

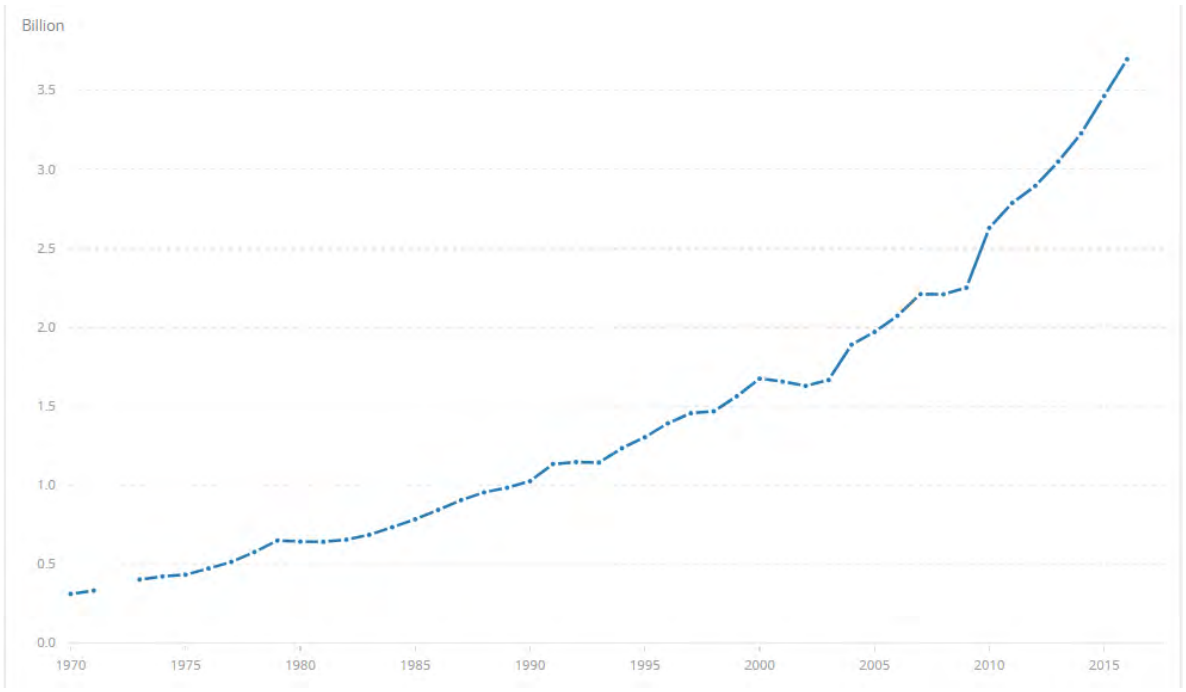


Figure 1.2: Air Transport - Passengers carried. (Source: International Civil Aviation Organization, Civil Aviation Statistics of the World and ICAO staff estimates, https://data.worldbank.org/indicator/IS.AIR.PSGR)

1.3 Applications of Operations Research in the Air Transport Industry

Fierce competition, great complexity, high consumers' standards and shareholders' expectations are all challenges faced by the airline firms. The management department of every airline is trying to find ways to gain and keep a leading edge in the competitive air transportation market and to run the airline as effectively and efficiently as possible, so as to respond to customers' needs. Since the 50s the airlines are using Operations Research models in solving their complex planning and operational problems. Operations Research (OR) has played a substantial role in helping the airline industry sustain high growth rates and make the transition from a luxury enjoyed by a few privileged people to a service industry for nearly everyone. The rest of this section is based on [2].

1.3.1 Revenue Management

A common theme for the airlines is that some flights have empty seats while others experience more passenger demand than that they can cover. In an effort to better match the demand for each flight with its capacity and to increase total revenues, airlines offer a variety of fare products at different prices for the same flight. Revenue management (RM) is used to determine the number of seats on each flight to be charged at each fare level, with the goal to fill each flight with the maximum possible revenue to maximize operating profit.

The sheer size and complexity of the airline seat inventory control problem led to the development of Information Technology solutions and particularly computerized RM systems. Even an average size airline might operate as many as 1,000 direct flights per day, using 10 fare classes and accepting bookings up to a year prior to each departure. The utilization of an IT system seems to be the only way to handle this situation as, at any point in time, this airline's seat inventory includes over three million booking limits, which can change with each booking accepted.

Previous booking data for the same flight and day of the week are combined with real-time reservation information for each future flight to produce a forecast of the total demand of each fare class for that flight.

These forecasts, along with estimates of the revenue value of each fare class, are subsequently fed into an optimization model that calculates the recommended booking limits for the examined flight departure. At the same time, the demand forecasts are fed into an overbooking model as well, which also utilizes historical information about passenger no-show rates to calculate an optimal overbooking level. Both the booking class limits and overbooking levels generated by mathematical models are then presented as recommendations to the manager.

The demand forecasts and booking limits are regularly updated during the flight booking period. If unexpected booking activity is observed, the system re-forecasts demand and re-optimizes its booking limit recommendations. A substantial proportion of the revenue gained due to the fare blend optimization is a result of this dynamic revision of booking limits.

Most airlines throughout the world have implemented RM systems and their benefits are notable. Effective use of techniques for overbooking and fare class mix alone have been estimated to generate revenue increases of as much as 4%–6% compared to cases in which such methods were not applied.

1.3.2 Applications to Aviation Infrastructure

Airports and air traffic management (ATM) systems are the main infrastructural elements of the global air transport network. Airports can be further subdivided into airside facilities (runways, aircraft stands) and landside facilities (passenger and cargo buildings), while ATM systems are viewed as being comprised of a tactical and a strategic control sub-system of air traffic. The design, development, and operation of all these facilities and systems has attracted the interest of operations researchers since the 60s and the 70s, when the industry experienced its greatest growth and the need to efficiently manage these systems became apparent.

Airport Operations: The runway complexes of major airports are considered among the scarcest resources of the international air transport system. New runways are very expensive to build, require great areas of land, and most importantly have environmental and other impacts that need long and complicated assessment processes. These challenges made the modeling of runway operations, specifically, an area of transportation science that received much attention. Analytical models and simulation tools are products of the research in the field.

Air Traffic Flow Management: Air traffic flow management (ATFM) gained major importance globally during the 80s, when rapid traffic growth rendered it necessary to manage air traffic more strategically. The goal of ATFM is to prevent local system overloading. More specifically, this is achieved by dynamically adjusting the flows of aircraft on a national or regional rather than the local level (holding arriving aircraft in the airspace nearby airports is a local measure of flow adjustment). It develops flow plans that attempt to dynamically match traffic demand with available system capacity over longer time horizons, typically extending from 3–12 hours in the future. Ground holding, i.e. intentionally delaying an aircraft's takeoff for a specified amount of time in order to avoid complications such as delays in the future, is the original application of ATFM and is encountered in periods of dense air traffic.

1.3.3 Aircraft and Crew Schedule Planning

Schedule planning in the airline industry entails designing future aircraft and crew schedules with the aim to maximize profitability. The effective combination of elements such as a network of flights, various aircraft types, airport slot and air traffic control restrictions, fleet maintenance requirements, crew work rules, and competitive, dynamic environments in which passenger demand is volatile, make the task of airline schedule planning daunting. Due to its complexity, no single optimization model has been solved, or even formulated, to address it as a whole. Instead, the decomposition of the overall problem into a set of sub-problems was a sensible alternative approach(see Figure 1.3) . These sub-problems are often defined as follows:

(1) *Schedule design*: Defining which markets to serve, how often and how to schedule flights to meet demand.
(2) *Fleet assignment*: Specifying what aircraft type to assign to each flight.
(3) *Aircraft maintenance routing*: Determining how to route aircraft to ensure satisfaction of maintenance requirements.
(4) *Crew scheduling*: Selecting which crews to assign to each flight to minimize crew

related costs.

Although sub-optimal, feasible aircraft and crew plans can be constructed by solving the sub-problems sequentially. The output of a sub-problem becomes the input for the subsequent one in the sequence. Even though these sub-problems are smaller and far simpler than the overall problem, they are still large-scale and complex.

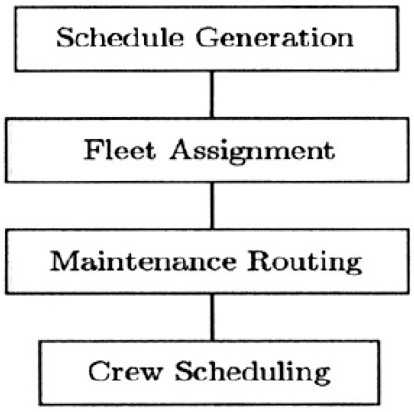


Figure 1.3: Schedule Planning. (Source: [2])

Aircraft Schedule Planning

Schedule Design: The flight schedule, specifies the flights to be operated and the departure time of each. A convenient flight schedule for the passengers is a great competitive advantage for the airlines and is thus a key factor for their profitability. Nevertheless, it is very challenging to design a schedule that covers the needs of the passengers and at the same time maximizes the firm's profits. The challenges, mainly, stem from the fact that the schedule affects and is affected by essentially all aircraft and crew scheduling decisions of the airline. Because of the incapability of optimization models to adequately solve the design problem, the common practice in many airlines in the past was to build flight schedules manually. Today, however, most airline firms employ decision-support systems to tackle this problem.

Fleet Assignment: With the flight schedule determined, the fleet assignment problem follows. The goal here is to find the cost-minimizing assignment of aircraft types to flights. Fleeting costs are comprised of:
(1) *Operating costs*: They represent the cost of flying a specific flight leg with a defined aircraft type.
(2) *Spill costs*: They measure the revenue lost when passenger demand for a flight is greater than the seating capacity of the assigned aircraft, which depends on its type.

Aircraft Maintenance Routing: With schedule design and fleet assignment decisions made, the next step is the aircraft maintenance routing. The goal here is to determine rotations for each aircraft in a fleet, so that it visits its maintenance stations regularly, as defined by the respective maintenance rules. A rotation is a sequence of flights, with the destination of one flight leg being the same as the origin of the next flight in the sequence.

Crew Schedule Planning

The employment of optimization techniques for the solution of crew scheduling problems with numerous, complex rules, and non-linear costs is a particularly sensible approach. With so many possible decisions, it is difficult to find even feasible solutions manually. Furthermore, crews represent the airlines' second highest operating cost after fuel, hence even slight improvements in their utilization can lead to significant savings. The crew scheduling problem is typically decomposed into two sequentially solved sub-problems, the crew pairing problem and the crew rostering problem(see Figure 1.4).

The crew pairing problem: The problem generates minimum-cost, multiple-day work schedules, called pairings (also termed flight routes). Regulatory agencies and crew union agreements specify the legal framework that defines how flight legs can be combined to create feasible flight routes. Work-rule restrictions include limits on the maximum number of flight hours in a day, the minimum number of rest hours between work periods, and the maximum time the crew may be away from their home base. Even with these limitations, the number of feasible pairings is immense. For example, in major U.S. airlines the number of pairings is in the order of billions. The costs of pairings is also complex because it is typically represented as a nonlinear function of flying time, total elapsed work time, and total time away from base of the crew.

The crew rostering problem: This problem combines the pairings generated previously into efficient month-long crew schedules, called rosters, assigning them to individual crew members. With rostering, which is primarily implemented in EU, schedules are constructed for and assigned to specific individuals, taking into account their particular needs and requests.With bidline generation, mostly utilized in the U.S., a generic set of schedules is constructed and, consequently, individual employees bid for these schedules, based on their preferences. The specific assignment of schedules to employees is usually based on seniority, with the more senior crew members most often assigned their preferred schedules.

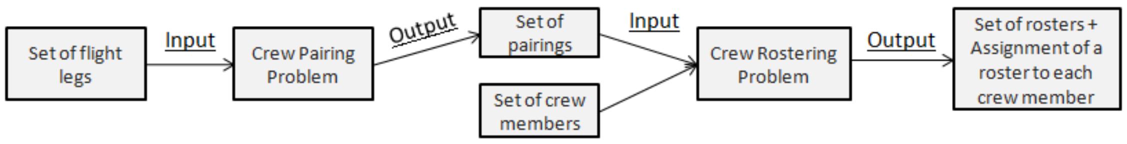


Figure 1.4: Crew Scheduling Process

To further explain, a pairing is a sequence of flight legs, the first of which departs from the airport base of the crew member and the last one in the sequence also ends at the home base. For example, if the home base of a pilot is Athens, then a probable pairing would constitute of the following flight legs: Athens - Thessaloniki, rest, Thessaloniki - Rome, rest, Rome - Berlin, rest, Berlin - Athens. Note that the pilot begins from his/her base, which in our case is Athens, and finally returns to it at the end of the pairing. Also note that between consecutive flight legs, the pilot should rest. This sequence of flights is to be assigned to one or more crew member working in one or more crew positions (termed ranks). Pilots, co-pilots and flight attendants are different ranks. The crew positions and the number of crew members at each position a pairing must be assigned to is referred to as crew complement.

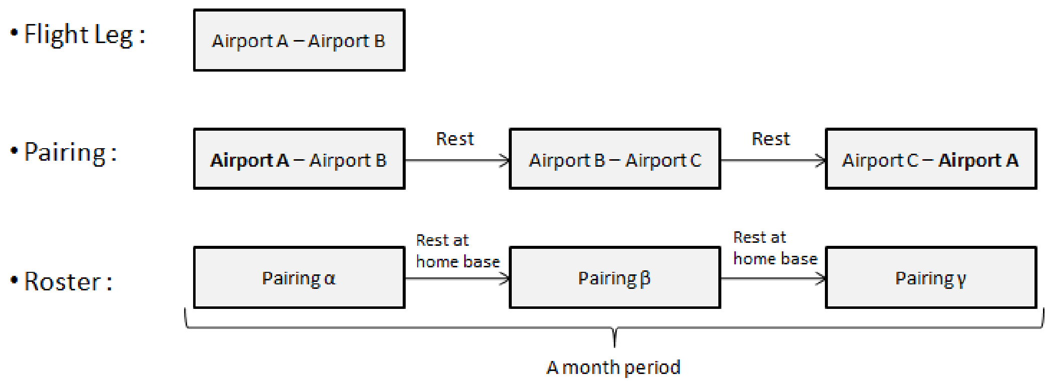


Figure 1.5: Schedule components

A roster is a sequence of pairings assigned to a crew member. It is his/her work schedule, usually for a month. For example, consider the case of our pilot. The first pairing that he/she is assigned is the one described above. The first flight (Athens - Thessaloniki) is on June 2. The last flight on his/her pairing (Berlin - Athens) is on June 5. Then the pilot rests for a few days in his/her home in Athens and afterwards should cover another pairing for this month. Suppose that this pairing is consisted of the following flight: Athens - London, rest, London - Dublin, rest, Dublin - Madrid, rest, Madrid - Athens. Suppose that the first flight (Athens - London) is scheduled on June 13 and the last (Madrid - Athens) on June 17. These two pairings and the rest days in the home base between them, are the monthly schedule or roster of the pilot (see Figure 1.5).

Finally, it is important to note that the crew scheduling problem is solved for each crew position, e.g. pilot, flight attendant, separately.

1.4 The Present Thesis

1.4.1 Focus: Crew Rostering Problem

In the present thesis we focus on the crew rostering problem. The problem takes as input a set of crew members and a set of pairings, which have been constructed during the preceding pairing generation phase. Each pairing is composed of a sequence of temporally and locally compatible flights interrupted by rest breaks. Temporal compatibility means that the arrival time of a flight leg in a particular pairing should always precede the departure time of the next flight in the sequence. Local compatibility means that the arrival airport of a flight leg is always the same with the departure airport of the next flight.

Each crew member has a history of flights operated in the near past and may also have a set of pre-assigned activities in the planning horizon, such as training sessions, standby duties, leaves, etc., which must be taken into consideration when designing rosters (also termed schedules) for him/her. Furthermore, each roster must comply with a large number of special rules, which are either enforced by government agencies or are the product of agreements between the aircrew unions and the airline administration (see Figure 1.6).

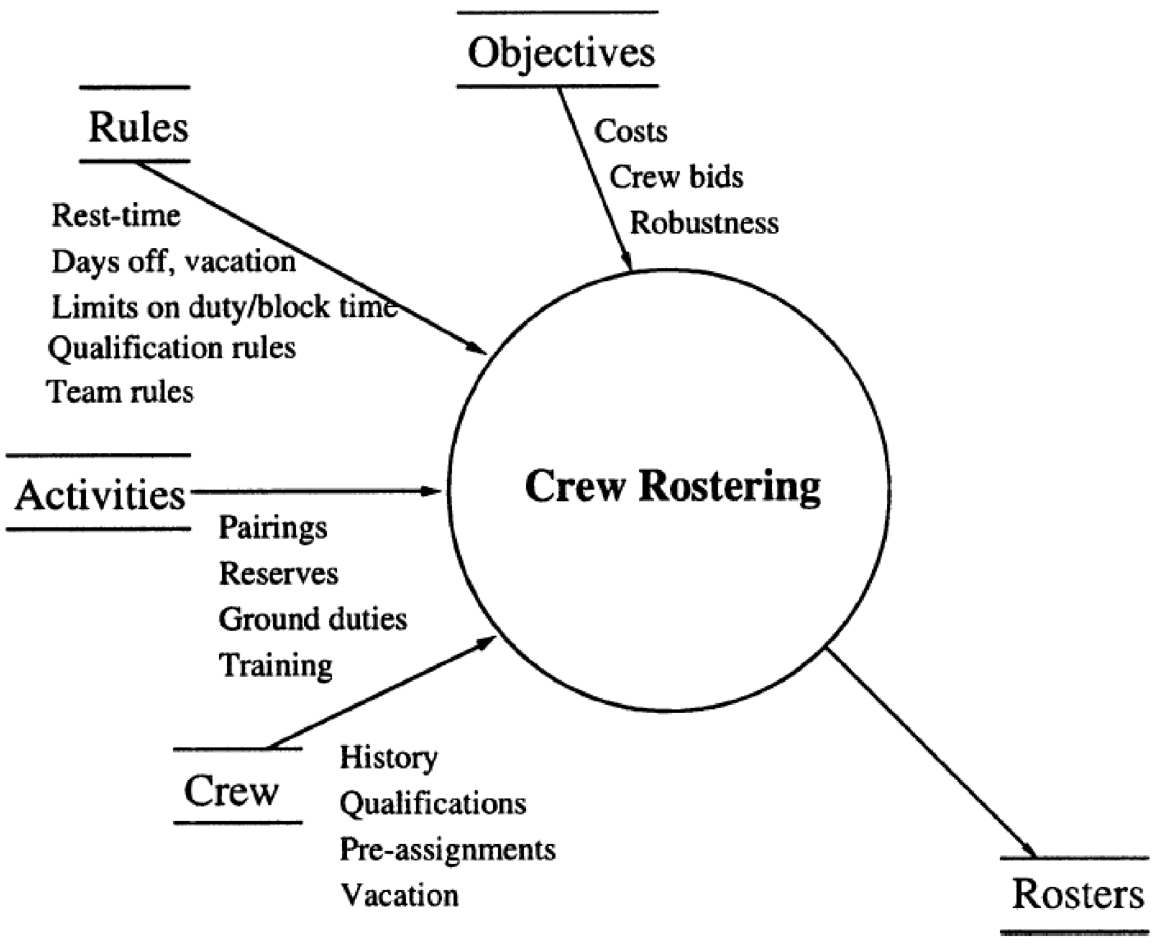
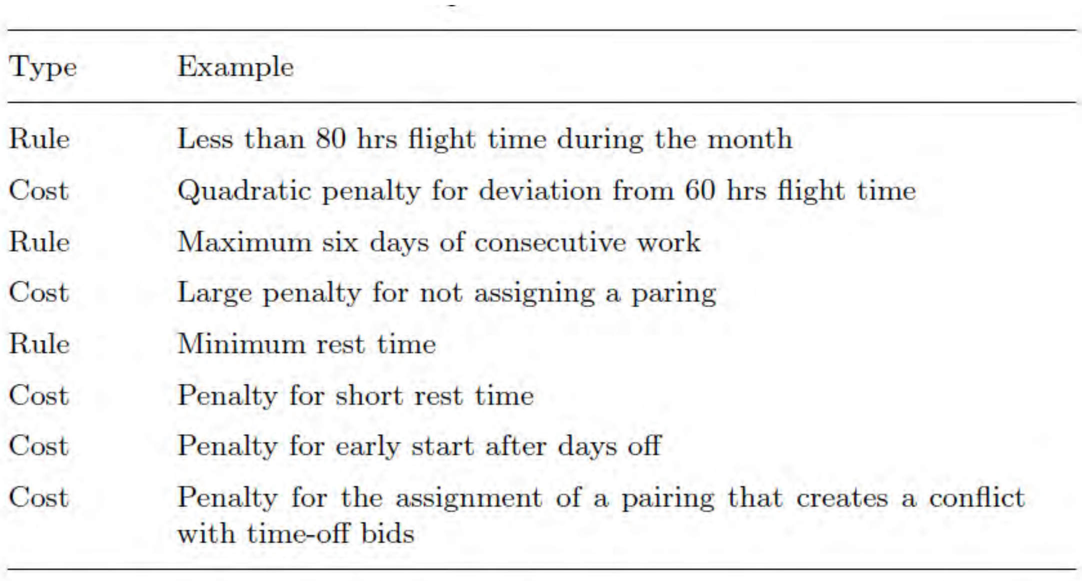


Figure 1.6: Representation of the airline crew rostering problem. (Source: [16])

At the same time, each roster is characterized by a quality cost, which is associated with the satisfaction of preferences of both the crew and the management. If requirements such as the achievement of total flight time within a certain time window, the avoidance of single duty days between two days off, the avoidance of consecutive night flight legs and other similar ones are satisfied then the roster quality cost is low. High quality rosters are imperative for the smooth and efficient operation of the airline.

Apart from the quality of the rosters, another crucial cost for the airlines is the coverage cost. This cost is zero when all the flights in the schedule of an airline are operated. However, due to the sheer size of the flight and crew member sets and to the complicating set of rules mentioned above, it is difficult, at times, to construct pairings and subsequently rosters that respect the rules and to assign them to crew members, so that all flights are covered. Many uncovered flights result in large coverage costs, which are harmful for the firm since they are related to customer discontent and other operational complications. Examples of rules and other types of costs are shown in Figure 1.7.



Type	Example
Rule	Less than 80 hrs flight time during the month
Cost	Quadratic penalty for deviation from 60 hrs flight time
Rule	Maximum six days of consecutive work
Cost	Large penalty for not assigning a paring
Rule	Minimum rest time
Cost	Penalty for short rest time
Cost	Penalty for early start after days off
Cost	Penalty for the assignment of a pairing that creates a conflict with time-off bids

Figure 1.7: Examples of roster-associated rules and costs. (Source: [14])

The crew scheduling problem aims to construct rosters for the crew members through the proper assignment of pairings, so as to maximize the coverage of the associated flight load while also maximizing roster quality. In other words, the output of the problem is the assignment of blocks of pairings interrupted by rest days in the home base to each crew member, in a way that minimizes the coverage and the roster quality costs.

According to [11], the crew rostering problem's solution is very challenging due to the following reasons:

1. The number of pairings is extremely large for many airlines. Generation of 100 million legal pairings even for medium-sized fleets and of several billions (!) of legal pairings for larger fleets is common.

2. Airlines have to comply with many complex work and safety regulations. One such rule is the 8-in-24 rule. One version of this rule requires that no more than eight hours of flying are assigned during any twenty-four hour period unless (i) a rest interval that is twice the flying time in the previous duty period is given to the crew member, and (ii) at least 14 hours of rest is given after the duty period during which the eight hour limit is violated. This is only one example of these rules. There are several other similar which the pairings and the rosters have to satisfy.

3. Crew costs depend on complex crew payment schemes and are highly non-linear.

The great impact and the difficulty of the rostering phase in the airline performance has motivated many operations researchers to invent advanced methods and techniques that can efficiently solve the challenging problem. The most successful of these techniques is the column generation algorithm. This algorithm, originally created to address huge linear programming problems, can be appropriately incorporated in a solution framework that can also tackle immense integer programming problems, such as the rostering problem. This framework is called branch & price and is a combination of column generation and branch & bound algorithm. Within this framework, the problem under consideration is solved using two optimization models, termed master problem and column generation sub-problem, respectively. The master problem is used as the means that enables the identification of a set of aircrew-rosters which cover the maximum number of flight routes at the minimum roster quality cost. On the other hand, the column generation sub-problem generates cost-efficient rosters which are inserted into the master problem with the aim to gradually improve its optimal solution. The master problem is typically formulated as a set-partition optimization model.

In a set-partition formulation, we demand that each flight route is covered exactly by the necessary number of crew member at each position (rank). For example, if a sequence of flights requires 1 pilot, 1 co-pilot and 3 flight attendants, then in the set-partition solution this pairing will be assigned to exactly this number of crew members.

1.4.2 Motivation

The discussed framework can facilitate the work of airline managers. It is integrated in commercial decision-support systems, a valuable tool in the hands of crew schedule planners. The solutions provided by this technique can be the final product of the crew scheduling process. Efficient rosters assigned to crew members. But before constructing the final schedules that are published to the crew, given a flight load (or a set of pairings) that needs to be covered, planners need to know whether a particular set of crew members can achieve its coverage. If this set can not provide adequate coverage then managers try a different set of crew. Generally, their aim is to find an as small as possible set of crew members that can result in low coverage cost.

To better illustrate the situation, an example is presented. Assume that a planner should achieve coverage of a set of 10 pairings, starting from and ending at the Athens' airport. The planner wants to utilize as less crew members, pilots for our case, as possible. Therefore, at first, he inserts a set of 2 pilots and the given set of 10 pairings as an input to the rostering problem. He solves the problem using the aforementioned framework. In the best solution that could be achieved by the program, only 6 of the 10 pairings are covered. This translates into large coverage costs. The planner is not satisfied by this solution. He resolves the problem, keeping the same set of pairings, which remains unchanged, but this time he inserts a set of 4 pilots. With this combination of sets, it turns out from the solution of the rostering problem that all pairings can be covered! However, the planner now wonders whether he could cover all of the pairings using a set of 3 pilots, instead of 4. Once again, he solves the problem.

Such situations are common in the airline management. In the first place, managers want to identify the smaller set of crew member that can cover a set of flight routes. When they find this set, they subsequently try to construct efficient rosters and to optimally assign them to crew members. In other words, the first concern of the planners is low coverage costs. Once they achieve that, they worry about the roster quality costs.

The situation described above can be very time-consuming. But time is a scarce resource for crew schedulers. Although the traditional solution framework can successfully provide efficient, final schedules with low overall roster quality cost, it is not the optimal way to go through the procedure described because, in many cases, it requires a large amount of time to produce a solution, that in fact is not intended to be published to crew. Therefore, the provision of a fast method to identify the maximum flight duty coverage that can be attained with a given set of crew members is of great value.

1.4.3 Contribution

This is what we try to do in the present thesis, building an alternative branch & price framework by formulating the master problem as a set-cover optimization model rather than a set-partition one, which is commonly used.

In a set-cover formulation, we allow each flight route to be covered by more crew members than those actually required. For example, in the previously discussed flight route, in which 1 pilot, 1 co-pilot and 3 flight attendants were needed, the set-cover solution is allowed to assign 3 pilots, 2 co-pilots and 4 flight-attendants or even more. This solution does not correspond to reality of course and requires further treatment, which we provide with an optimization model. This model suitably removes the over-covered pairings from the schedules of excess crew members. In the final solution, the flight route will be covered by (or will be included in the rosters of) 1 pilot, 1 co-pilot and 3 flight attendants, exactly as needed.

The reason why we do this is because we anticipate that the set-cover formulation can be solved faster than the set-partition one, which is proved by our tests.

It is important to note, however, that our design does not aim to identify a final solution that will be implementable, but to help crew planners identify the attainable coverage very quickly compared to the traditional approach and to take appropriate corrective actions in case of open flight duties, such as utilizing a larger number of crew members.

For the needs of our research, we developed a computer program, written in C programming language, in which we implemented the branch & price framework. Both the traditional set-partition and our proposed set-cover formulations were subsequently integrated into the program and their comparative performance was tested.

This thesis and the computer program created are also a contribution to the development of a platform for research purposes, where programs managing individual airline operations, written by students and researchers of the Systems Optimization Laboratory, could be integrated. In this way, research on the optimization of the whole airline operation will be possible. Therefore, apart from being the means for the conduction of the comparative analysis of two different modeling approaches, this program will also be a part of the system to be developed.

1.5 Organization of the Thesis

We began this thesis by exploring the history of aviation. Key facts and figures from this section hopefully showed to the reader the reasons behind the great challenges that the management of the airline industry faces. Operations research and its methods have proven an effective answer to these challenges for many operations of this industry. An operation to which this scientific field has substantially contributed is Crew Scheduling. We presented one of the two problems into which the crew scheduling is decomposed, the crew rostering problem, and discussed it in detail. The first chapter concluded with the motivation and the contribution of the present thesis.

A literature review, which presents the scientific work used in our thesis, follows in the second chapter. There the reader can find the basic ideas behind and a brief description of the algorithms used in the present thesis along with valuable resources on solution methodologies developed to tackle huge integer and linear programming problems. Also, literature dealing with the aircrew scheduling problems is provided. The review is narrowly focused on these two themes because they are tightly associated with the present thesis. There is a vast amount of research published in the field of operations research, combinatorial optimization and airline operations but a more elaborate presentation of all these fields was considered out of the scope of this thesis and would probably confuse the reader.

In the following chapter [3], we focus on the mathematical models that facilitate the development of an efficient branch and price solution methodology for the crew rostering problem. We begin by focusing on the alternative master problem formulations that were deployed. The optimization models, termed as master problems in our solution framework, basically try to (efficiently) cover the maximum possible number of pairings of a given set, with a particular set of crew members. We discuss thoroughly the traditional set-partition and the proposed set-cover formulations, which are the master problems, highlight their differences and explain the anticipated benefits of the set-cover formulation for our intended use. Subsequently, we also present an optimization problem that converts the set-cover solution to a set-partition one. This conversion is necessary because the set-partition solution is that which actually has practical meaning, as we will explain. A practical example of this conversion is demonstrated. Finally, we present the column generation problem, which generates cost-effective rosters that are fed into the master problem. This problem is modeled as a shortest path problem with side constraints. For the modeling needs of the column generation problem, we consider a network of crew members and pairings.

Next, in the fourth chapter, we present the algorithmic framework utilized in our program. Due to the complexity of our method, at least for a reader unfamiliar with such themes, we provide a gradual presentation. At first, we present the parts of our methodology. Next, we present the specific framework that was used in our program. More specifically, we demonstrate how the parts discussed above are synthesized into an effective solution framework (synthesis) with the help of a small case study, solved by our program. We display a small case, in which a set of 16 flight routes must be covered by a set of 3 crew members and we follow the program step by step, presenting the results at each stage and the final product, which is a monthly schedule for each crew member.

In the fifth chapter, we present 2 alternative program designs that were deployed. The first design utilizes the proposed set-cover formulation and subsequently turns the set-cover solution to a set-partition one. The second design uses the traditional set-partition solution. We offer experimental results from tests performed in near-realistic environments, which demonstrate the comparative performance of the two alternative formulations and we report the conditions under which each of them is more suitable.

Next, in the sixth chapter, we show future directions of research and of further development of our program.

Finally, we close with a summary and the bibliography.

Chapter 2

Literature Review

Operations research methods are utilized across the spectrum of airline operations. Much of the research in this scientific field was motivated by the problems faced by airlines. As a result, there is a huge amount of relative resources in the literature. For the needs of this thesis, we are going to focus only to a small fraction of this work and particularly in articles that were studied for the development of the present thesis. The literature review is divided into two parts. In the first part, research focusing on the algorithms employed in our program is presented. The second part is dedicated to literature concerning aircrew scheduling and implementation of optimization algorithms in this context. The aim of this review is to introduce the reader to aircrew scheduling and to the optimization techniques used in the management of this operation, by giving a set of useful, relevant sources.

2.1 Optimization methods and Algorithms

Column Generation

In simple terms, column generation is a way of beginning with a small, tractable part of a problem (specifically, a few of the total decision variables), solving this sub-problem, analyzing its solution to identify the next part of the problem (in particular, one or more decision variables) to add to the formulation, and then resolving the enlarged problem. Column generation repeats that process until it achieves a satisfactory solution to the whole of the problem.

In other words, column generation is a way of solving a linear programming problem by adding columns (corresponding to variables) during the pricing phase of the simplex method of solving the problem.

Column generation takes advantage of the fact that in the simplex method, the solver does not need access to all the variables of the problem simultaneously. In fact, a solver can start with only the basis (i.e. a particular subset of the variables) and subsequently use reduced cost to identify which other variables to access as required.[6]

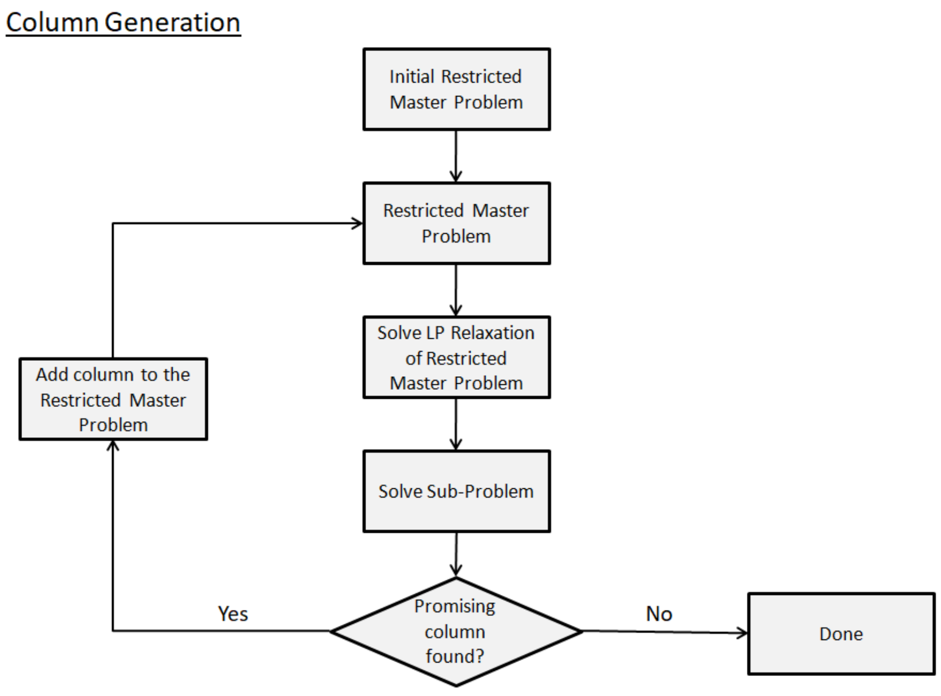

Column Generation
Initial Restricted Master Problem
Restricted Master Problem
Solve LP Relaxation of Restricted Master Problem
Solve Sub-Problem
Promising column found?
Yes
No
Done
Add column to the Restricted Master Problem

Figure 2.1: Column Generation flowchart

In [7], pioneers in the field of optimization techniques and especially in the column generation algorithm, present many successful application of this technique, in various industry contexts.

The book starts with A Primer in Column Generation by Jacques Desrosiers and Marco E. Lübbecke. [9] introduces the utilization of column generation method in integer programming problems. The relevant theory and more advanced ideas necessary to tackle large-scale practical problems are illustrated with various examples. More specifically, issues discussed there include the integration of Dantzig-Wolfe decomposition and Lagrangian relaxation within a branch-and-bound framework, deriving natural branching and cutting rules with the utilization of a compact formulation, and understanding and influencing the behavior of dual variables during column generation. The presentation of most concepts is facilitated via a small example, which makes this work very didactic and a great introduction to column generation.

Branch & Bound

The branch and bound framework is a fundamental methodology for producing exact solutions to NP-hard optimization problems. This algorithm implicitly enumerates all possible solutions to the problem under consideration, by saving partial solutions called sub-problems in a search tree (see Figure 2.2). Unexplored nodes in the tree generate, so-called, children by partitioning the solution space into smaller regions that can be solved repeatedly (i.e. branching), and rules are used to cut off regions of the search space that are proved sub-optimal (i.e. bounding). Once the entire tree has been explored, the search is over and the best solution found is given.[21]

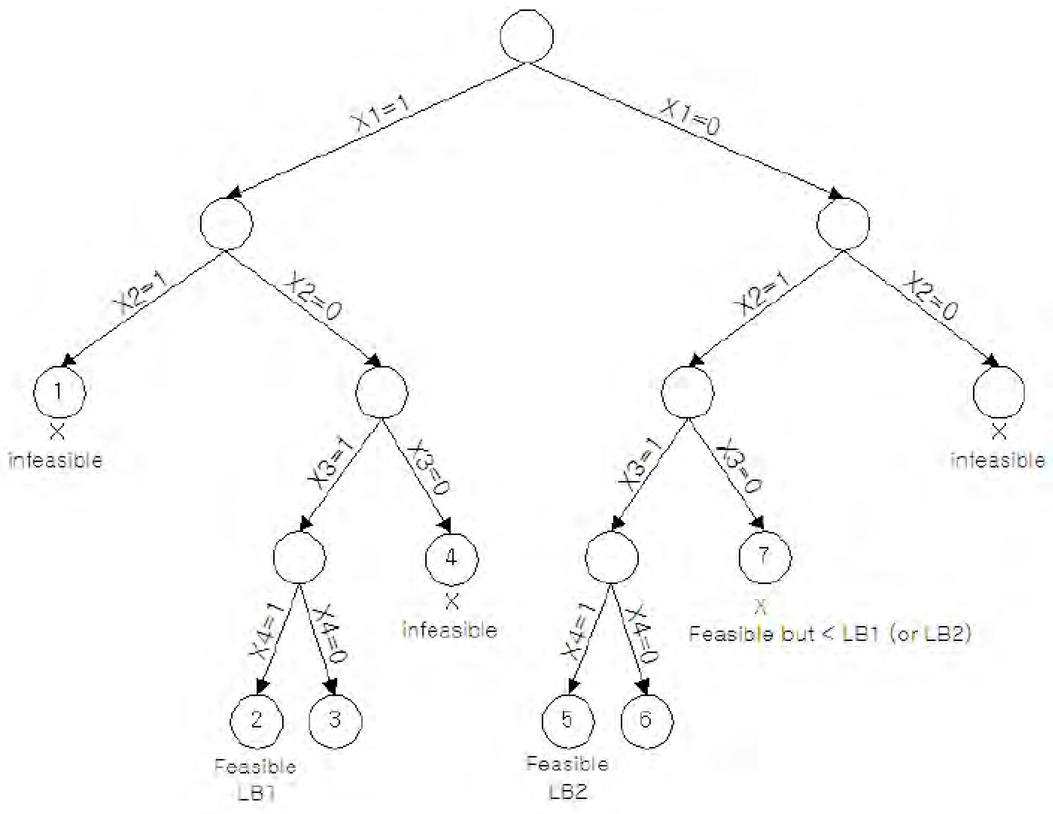


Figure 2.2: Branch and Bound search tree example (Source: [23])

In [21], the researchers provide an introduction to the Branch & Bound algorithm and the basic ideas behind it. Subsequently they identify three algorithmic components in B & B that can be specified by the user to fine-tune the behavior of the algorithm. These components are the search strategy, the branching strategy, and the pruning rules. Their survey presents a description of recent research advances in the design of B & B algorithms, particularly with regard to these three components.

Branch & Price

Branch and price is a method of combinatorial optimization for solving integer programming problems with a huge number of variables or columns. The method is a combination of branch and bound and column generation algorithms.

In more detail, branch and price is a branch and bound method in which at each node of the search tree, columns may be added to the LP relaxation of a master problem. At the start of the algorithm, sets of columns are excluded from the LP relaxation in order to reduce the computational requirements and then some columns are added back to the LP relaxation appropriately. The approach is based on the observation that for large problems most columns will be non-basic and have their corresponding variable equal to zero in the optimal solutions. Hence, the large majority of the columns are irrelevant for solving the problem and they do not provide any useful information.

The algorithm typically begins with a master problem formulation which describes the actual problem faced. However, the master problem usually contains many variables and so a modified version, called the Restricted Master Problem, that only considers a subset of the columns is solved. Subsequently, to check for optimality, a sub-problem called the pricing problem (or the column generation problem) is solved to find columns that can enter the basis and reduce the objective function (for a minimization problem). This involves finding a column that has negative reduced cost. Note that the sub-problem must be solved to completion in order to ensure that an optimal solution to the Restricted Master Problem is also an optimal solution to the Master Problem. Each time a column with negative reduced cost is found, it is added to the Restricted Master Problem and the relaxation is re-optimized. If no columns can enter the basis and the solution to the relaxation is not integer, then branching occurs. At the end of this procedure, an integer solution is produced.

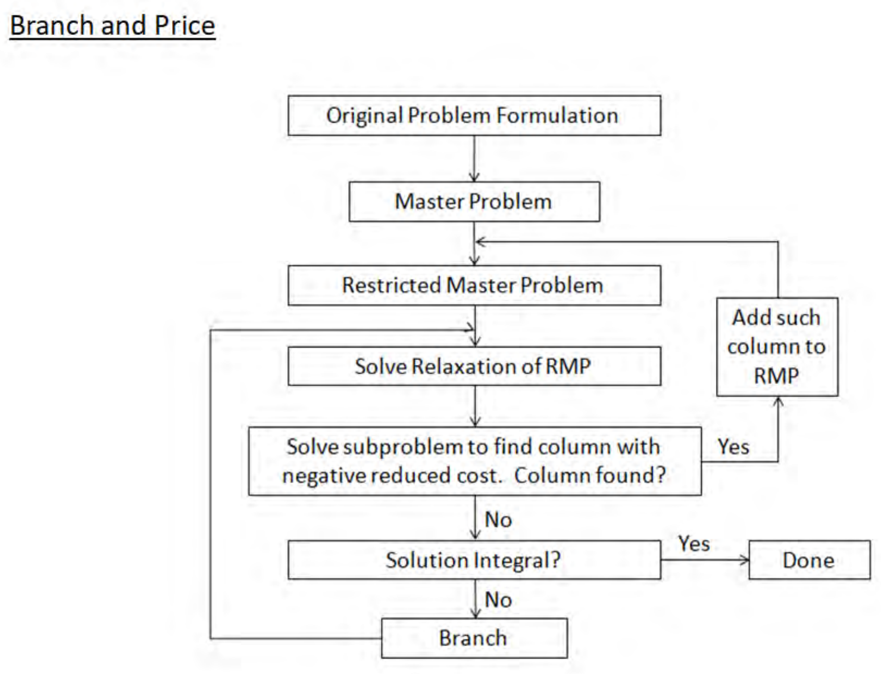

Branch and Price
Original Problem Formulation
Master Problem
Restricted Master Problem
Add such column to RMP
Solve Relaxation of RMP
Solve subproblem to find column with negative reduced cost. Column found?
Yes
No
Solution Integral?
Yes
Done
No
Branch

Figure 2.3: Branch and Price flowchart (Source: [1])

A significant resource regarding column generation and its integration to a solution method, i.e. branch and price, that addresses integer programming problems, is [4]. The researchers present classes of models for which this approach decomposes the problem, provides tighter LP relaxations and eliminates symmetry. Subsequently, they discuss computational issues and implementation of column generation, branch & bound algorithms, including special branching rules and efficient ways to solve the LP relaxation. The branch & price framework was also implemented for the present thesis and the discussed article was a valuable resource.

Acyclic Shortest Path Algorithm with Labels

In the second chapter of [7], Shortest Path Problems with Resource Constraints [8], Stefan Irnich and Guy Desaulniers report a comprehensive survey of the problems used to cast the sub-problem (i.e. column generation problem) in most vehicle routing and crew scheduling applications solved by column generation. The researchers propose a classification and a generic formulation for the SPPRCs, briefly discuss complex modeling issues involving resources and finally present the most commonly used SPPRC solution methods.

The Shortest Path Problem with Side Constraints (SPPRC) consists of finding a shortest path among all feasible paths that start from a source and end at a sink node. Each path is required to satisfy a set of constraints defined over a set of resources. A resource corresponds to a quantity. For example time, the load transferred by a vehicle or the duration of a rest interval in a work shift. These quantities change along a path according to functions, called resource extension functions (REFs). A REF is defined for every arc in the graph and for every resource considered. It imposes a lower bound on the value that the associated resource can take at the head node of the corresponding arc, taking into account the values assumed by all the resources at its tail node. The resource constraints are represented as intervals, often called resource windows, which restrict the values that can be taken by the resources at every node along a path. Such a constraint is defined for every node in the graph and every resource considered.[8]

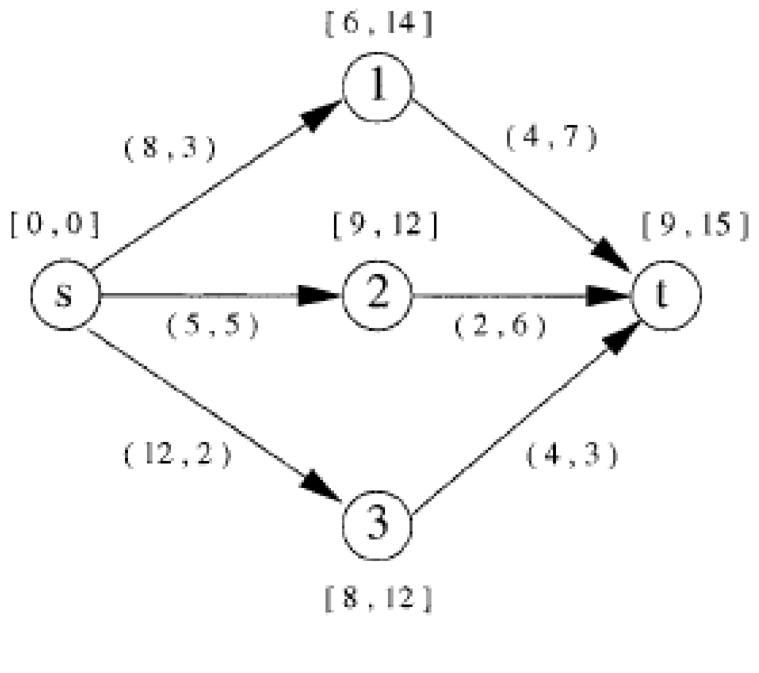


Figure 2.4: A small SPPRC example (Source: [8])

Another great insight to the solution of the sub-problem of the column generation algorithm employed in the present thesis was [8]. The article presents the Shortest Path Problem with Time Windows (SPPTW) and an efficient generalized permanent labeling algorithm to solve it. This algorithm is based on the definition of the concept of a generalized bucket, which is a core part of our sub-problem solution approach. The computational efficiency of this algorithm, utilized to solve a specific case problem, is also discussed in this article.

Finally, in [20], an algorithm for solving the minimal path problem in the general directed acyclic graph with labelled edges is presented. This resource was particularly useful for our work because the SPP we faced was also acyclic and the specific algorithm is taking advantage of the fact that the associated network is acyclic, resulting in an efficient solution method.

Ranking Paths Algorithm

Another useful algorithm for our framework can be found in [19]. In this paper an algorithm for determining the K best paths that may contain cycles in a directed network is presented. The proposed algorithm can be used not only for the typical K shortest paths problem but also for ranking paths under a non-linear objective function. Note that an algorithm to determine the best path exists is needed though. Computational results are also presented and comparisons with other solution approaches for this problem are made. The algorithm introduced in this paper is used in our program with the aim to prohibit the creation of a specific roster in following steps of the solution process.

The basic idea behind the algorithm is, in short, once the shortest path has been determined it is excluded from the network in such a way that no new path is formed and no more paths are excluded. This step leads to an enlarged network where all the paths, but the shortest one, can be determined.

Solution Methodology

Finally, but definitely not least, in [17], Kozanidis considers the problem of covering a set of shipments in a logistic distribution network with a fleet of aircraft. Although in a different context, his proposed methodology, which utilizes a master problem that covers the maximum possible number of shipments using a given set of aircraft-routes, and a column generation sub-problem that generates cost-effective aircraft-routes which are fed into the master problem, is used in the present thesis, after making necessary modifications to address the crew rostering problem that we face. Its didactic value is enhanced by elaborate descriptions of the algorithms used and by helpful figures presented.

2.2 Aircrew Scheduling

A great introduction to the problem of aircrew scheduling can be provided by [3]. In this work, the reader learns about the different phases of crew scheduling in the airlines, its components and its challenges. Mathematical formulations and alternative solution methods for these problems are offered. Integration of different scheduling operations is discussed as well.

The pairing generation problem precedes the rostering one in the aircrew scheduling process. It has concentrated the interest of many researchers and for a reason, as generating efficient pairings is a challenging task but one that yields many benefits. In [18], apart from an introduction and a definition, researchers proposed a novel approach to crew-pairing problem. The problem is first formulated as a large scale set covering problem with many columns, with each column representing a crew-pairing. They then suggest a solution procedure for the continuous relaxation of this large scale problem, based on generalized linear programming, in which the column generation sub-problem is shown to be equivalent to a shortest path problem in an associated graph. Computational results obtained on various real problems are also reported.

In [16], the authors give a more comprehensive description of real-world airline crew rostering problems and the mathematical models used to capture the various constraints and objectives found in the airline industry. Linking the theory with the real-world application is very important because it shows the impact that the techniques discussed here have to the industry. Furthermore, they present the solution methods employed in a commercial crew rostering program, the Carmen Crew Rostering system, which is currently in use at several major European airlines including British Airways, KLM and others, as well as at one of the world's largest passenger transportation companies, Deutsche Bahn (German State Railways). [14] is one more article featuring the Carmen Crew Rostering system. It describes the sequence of algorithms that are applied and the way these have been modified and extended in order to cope with the largest problems. At this point, it is important to highlight the application of the same optimization techniques across a spectrum of transportation industries.

Another research that demonstrates the success of the optimization methods in real-life cases is [10]. This article describes a method for solving the crew rostering problem in air transportation. Specifically, a generalized set partitioning model and a method using column generation are employed. Adaptation of this method in a number of ways is done to take advantage of the nature of the problem and to accelerate solution. Numerical tests on problems from Air France illustrate that this method is capable of solving very large scale problems with thousands of constraints and hundreds of sub-problems. According to the researchers, the tests also show that their proposed enhancements are capable of reducing solution time by a factor of about a thousand (!). Finally, results from this method are compared with those obtained with a method used at Air France.

[15] proposes that by using sophisticated optimization models and algorithms many airlines are able to improve profitability. The authors review these models and the underlying solution methodologies. They focus on models involving strategic business as well as operational processes. The former models include schedule design and fleeting, aircraft routing, and crew scheduling, while the latter models deal with disruptions and irregular operations.

At this point, it may seem that column generation related techniques monopolized our interest. But it holds true that column generation and the frameworks developed around it revolutionized the way operations researchers deal with programming problems with a vast number of decision variables. Aircrew problems exhibit such number of decision variables. As a result, it is true that frameworks integrating column generation are predominantly used in the aircrew scheduling field [2]. Therefore, it seems sensible to the author to focus on the relevant literature for the needs of the present thesis.

Chapter 3

Model Development

In this chapter, we focus on the mathematical models that facilitate the development of an efficient branch and price solution methodology for the crew rostering problem. We begin by focusing on the alternative master problem formulations that were deployed. The optimization models, termed as master problems in our solution framework, basically try to (efficiently) cover the maximum possible number of pairings of a given set, with a particular set of crew members. We discuss thoroughly the traditional set-partition and the proposed set-cover formulations, which are the master problems, highlight their differences and explain the anticipated benefits of the set-cover formulation for our intended use. Subsequently, we also present an optimization problem that converts the set-cover solution to a set-partition one. This conversion is necessary because the set-partition solution is that which actually has practical meaning, as we will explain. A practical example of this conversion is demonstrated. Finally, we present the column generation problem, which generates cost-effective rosters that are fed into the master problem. This problem is modeled as a shortest path problem with side constraints. For the modeling needs of the column generation problem, we consider a network of crew members and pairings.

3.1 Master Problem

3.1.1 Problem Type Discrimination: Set-Partitioning versus Set-Covering

Before we present the mathematical models, we would like to make sure that the reader understands the difference between the two problem types: the set-partition and the set-cover. In the set-partition problem we demand that each pairing is covered exactly by the number of crew members needed to actually operate these flights. Flights require crew in various positions. For example, one pilot (or captain) is always needed. Moreover, a co-pilot is also required, as well as a number of flight attendants. The required number of crew members of each rank is termed as the complement of a pairing. The complement may vary between different pairings and positions. For example, pairings that are operated with larger aircraft would usually have a larger complement in the flight attendant's position.

To best illustrate this situation we present Figure 3.1. Suppose that we deal with the scheduling of pilots at this point. In the left side of Figure 3.1, there is a set of pairings. Each pairing requires 1 pilot (complement=1). On the right side, there is a set of pilots. As the reader can see, each pairing is assigned to exactly one pilot. Pilot A is assigned Pairing i and Pairing ii. Pilot B is assigned Pairing iii and Pilot C, Pairing iv. As a result, Figure 3.1 could represent a potential solution of a set-partition formulation.

On the other hand, in the set-cover problem, we allow the assignment of a pairing to more pilots than the number actually required (see Figure 3.2). Once again each pairing requires one pilot (complement=1). However, Pairing ii is now assigned both to Pilot A and to Pilot B and Pairing iii is assigned both to Pilot B and to Pilot C. To illustrate, in reality, that would roughly mean that Pilot A and Pilot B would both enter the airplane to operate the flights constituting Pairing ii, while actually just one is needed. Of course, this is not sensible. Figure 3.2 could represent a potential solution of a set-cover formulation.

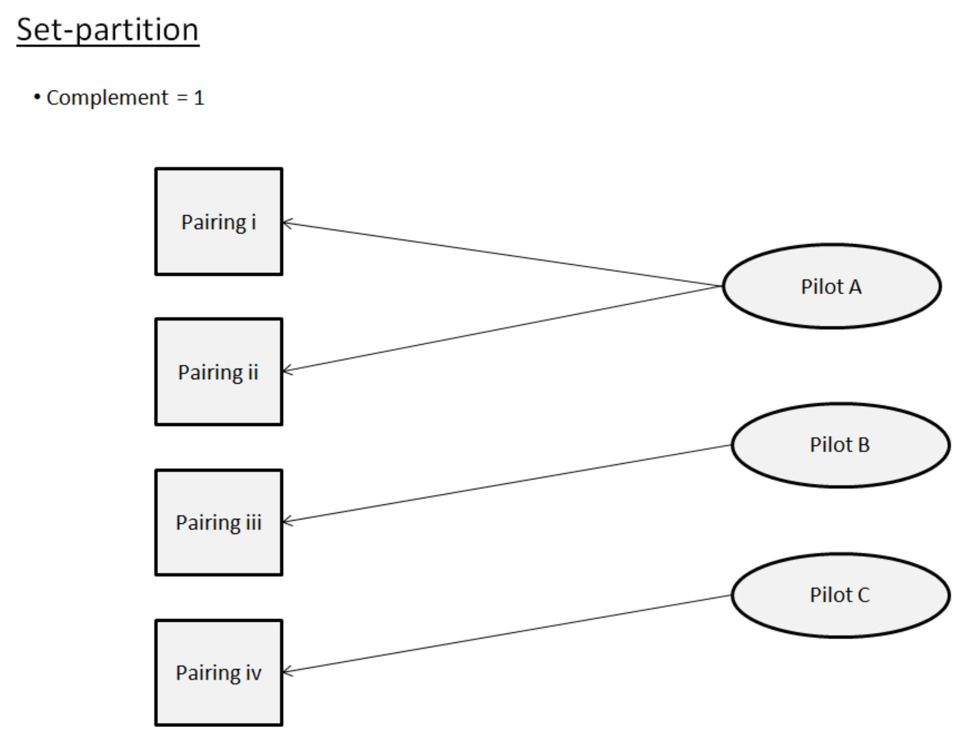


Figure 3.1: Set-partition - Exact coverage required

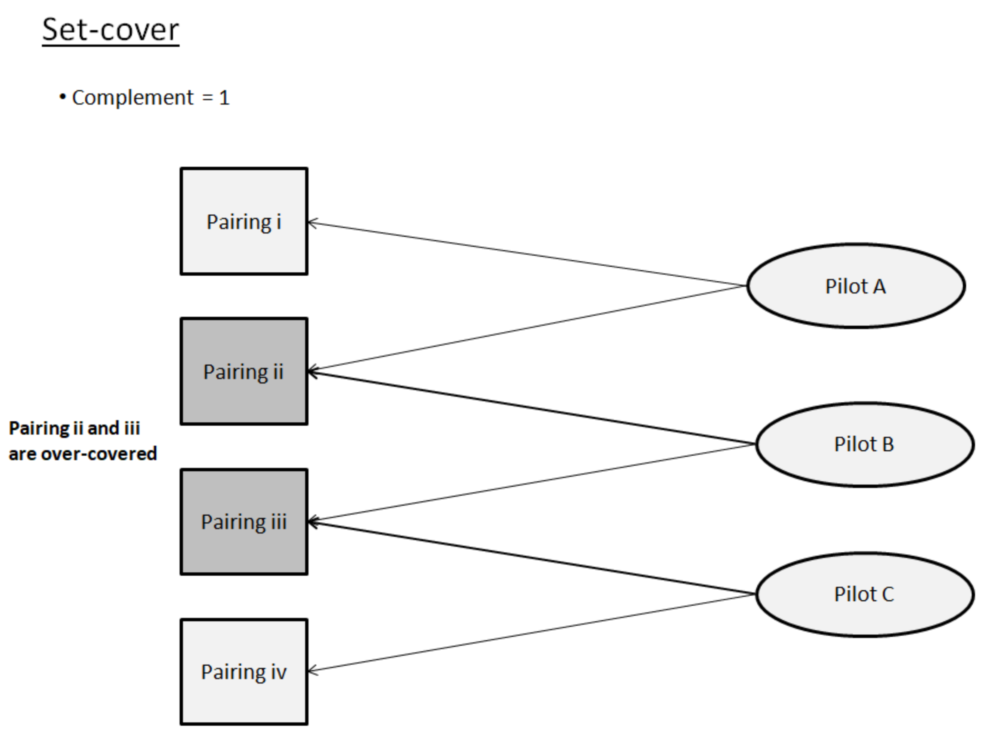
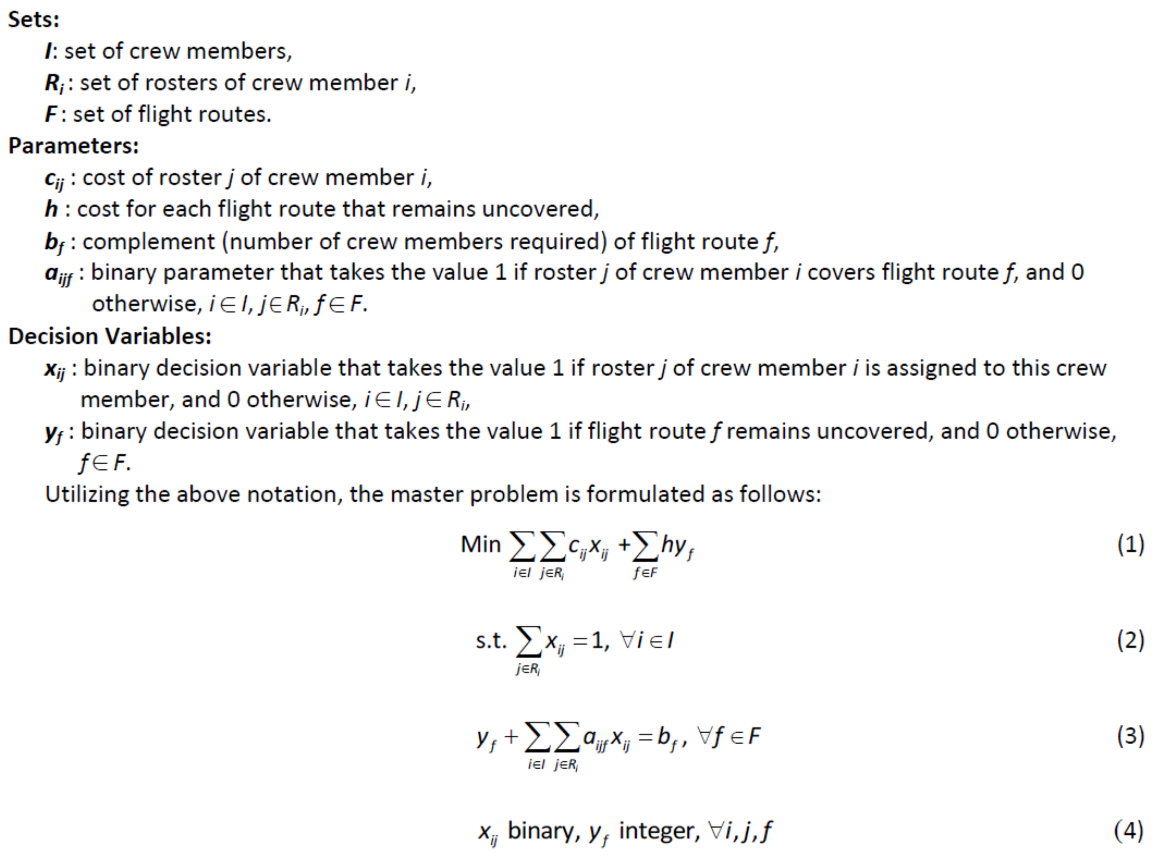


Figure 3.2: Set-cover - Over-coverage allowed

3.1.2 Set-Partition Formulation

First, we present the set-partition formulation(see Figure 3.3).



Sets:

I: set of crew members,

***R**$_i$*: set of rosters of crew member *i*,

F: set of flight routes.

Parameters:

***c**$_{ij}$* : cost of roster *j* of crew member *i*,

h : cost for each flight route that remains uncovered,

***b**$_f$* : complement (number of crew members required) of flight route *f*,

***a**$_{ijf}$* : binary parameter that takes the value 1 if roster *j* of crew member *i* covers flight route *f*, and 0 otherwise, $i \in I, j \in R_i, f \in F$.

Decision Variables:

***x**$_{ij}$* : binary decision variable that takes the value 1 if roster *j* of crew member *i* is assigned to this crew member, and 0 otherwise, $i \in I, j \in R_i$,

***y**$_f$* : binary decision variable that takes the value 1 if flight route *f* remains uncovered, and 0 otherwise, $f \in F$.

Utilizing the above notation, the master problem is formulated as follows:

$$\text{Min} \sum_{i \in I} \sum_{j \in R_i} c_{ij} x_{ij} + \sum_{f \in F} h y_f \tag{1}$$

$$\text{s.t.} \sum_{j \in R_i} x_{ij} = 1, \ \forall i \in I \tag{2}$$

$$y_f + \sum_{i \in I} \sum_{j \in R_i} a_{ijf} x_{ij} = b_f, \ \forall f \in F \tag{3}$$

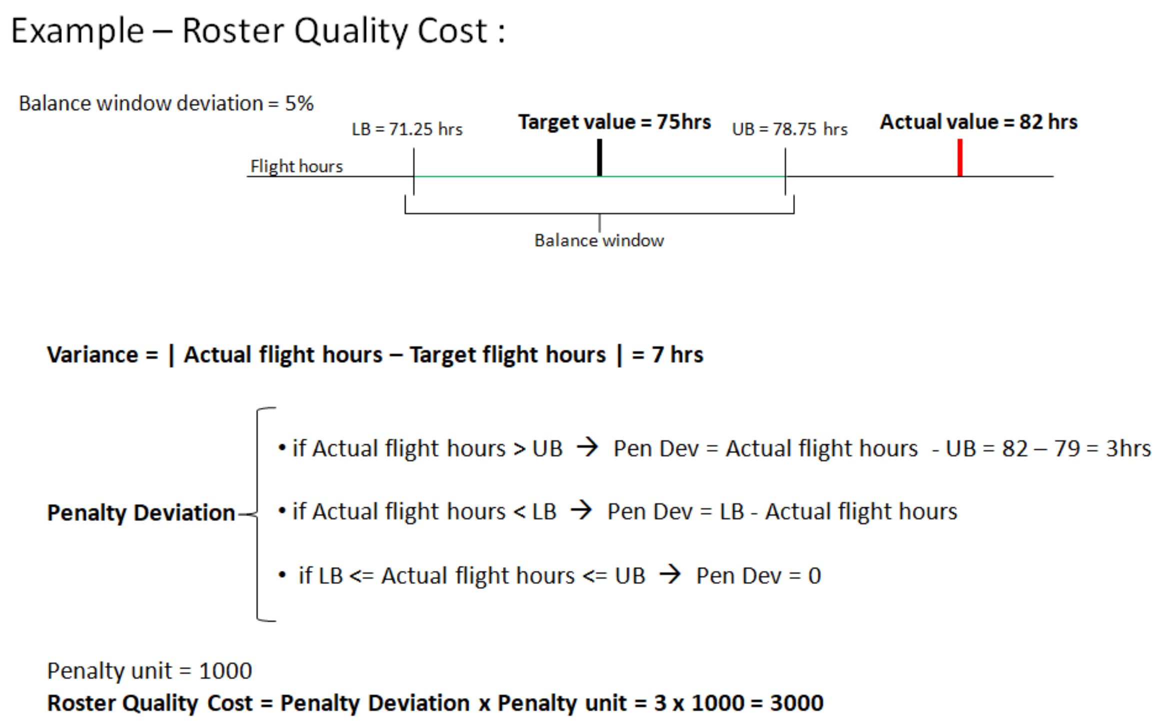
$$x_{ij} \text{ binary}, \ y_f \text{ integer}, \ \forall i, j, f \tag{4}$$

Figure 3.3: Set-partition formulation

The objective function (1) minimizes the total cost, which comprises of the cost of uncovered pairings plus the roster quality cost. Constraint set (2) ensures that exactly one roster is assigned to each crew member. Constraint set (3) states that each flight route must be covered exactly by the required number of crew members. Variable y_f becomes positive when flight route f is not fully covered, in which case the corresponding penalty h is imposed in the objective. Finally, constraint set (4) restricts the decision variables of the problem to integral values.

Roster Quality Cost

In the present work we assume that the cost of each roster is determined by a single quality factor, i.e., flight time balance. More specifically, a desirable window ($\mathrm{LB_i}$, $\mathrm{UB_i}$) is defined for the total flight time to be assigned to crew member i, and a penalty Pen is assigned for each unit time deviation of the actual flight time outside this window. We present an example to illustrate (see Figure 3.4).


Example – Roster Quality Cost :

Variance = | Actual flight hours – Target flight hours | = 7 hrs

Penalty Deviation
- if Actual flight hours > UB → Pen Dev = Actual flight hours - UB = 82 – 79 = 3hrs
- if Actual flight hours < LB → Pen Dev = LB - Actual flight hours
- if LB <= Actual flight hours <= UB → Pen Dev = 0

Penalty unit = 1000
Roster Quality Cost = Penalty Deviation x Penalty unit = 3 x 1000 = 3000

Figure 3.4: Roster quality cost - Example

The total variance is a measure of the total balance, since the variance of a particular crew member is defined as the absolute difference between his actual flight time and his target flight time (midpoint of associated balance window). If the flight time of a roster falls within the corresponding balance window, then its cost is equal to 0. This means that, for sufficiently small variance, the associated cost is 0. Otherwise, the cost is equal to a unit penalty multiplied by the flight time deviation outside the corresponding window. In our program, the hourly balance penalty was set equal to 1000. In our example, the flight time deviation over which penalty is imposed is set equal to 5% of the corresponding target value. Thus, for a flight time target of 75 hours, the corresponding balance window is [71.25, 78.75]. In practice, the flight time deviation outside the balance window is rounded to the next hour. In our case, the total flight hours for the crew member in the planning horizon (i.e. month) are 82. This exceeds the upper bound of his/her balance window by 3 hours and thus a penalty is imposed. There are reasons behind this penalization. The extra hours may cause discontent to the crew member or maybe the airline has to pay him/her more money due to overtime. On the other hand, if the actual flight hours are less than the lower bound of the balance window, the associated crew member is underutilized, which is considered bad management. In our example, the quality cost of our examined roster is 3000.

Coverage Cost

Cost coefficient h is always much larger than cost coefficients c_{ij}, imposing the relative priority between the two objectives, since an uncovered pairing is considered much worse than a bad quality roster. Remember what a pairing is. It is a sequence of flight legs. If the airline can not cover a pairing that means that a few flights will be canceled. This cancellation results in hundreds of angry passengers, which translates to bad reputation for the firm and of course compensations for the passengers. As a result, an uncovered pairing is something that the airlines definitely seek to avoid, even at the expense of bad quality rosters. This justifies our decision to set the value of h equal to 1000000, a much higher value than that of the hourly balance penalty (it is set to 1000). To illustrate we present an example in which a pairing remains uncovered (see Figure 3.4). The reason why a pairing may not be covered is that the set of crew members proposed for these duties is not sufficient. In many cases, a larger set of crew member, e.g. more pilots, could provide total coverage. In our example, it is likely that Pairing iv overlaps with another pairing. That means that the flights constituting the pairings are scheduled on the same dates. Since there are only 3 crew members and 4 pairings, one pairing remains uncovered. Another possible explanation is that the work rules discussed previously prohibit the full coverage of this set of pairings with this particular set of crew members. In any case, another set of crew members is required to cover all flights.

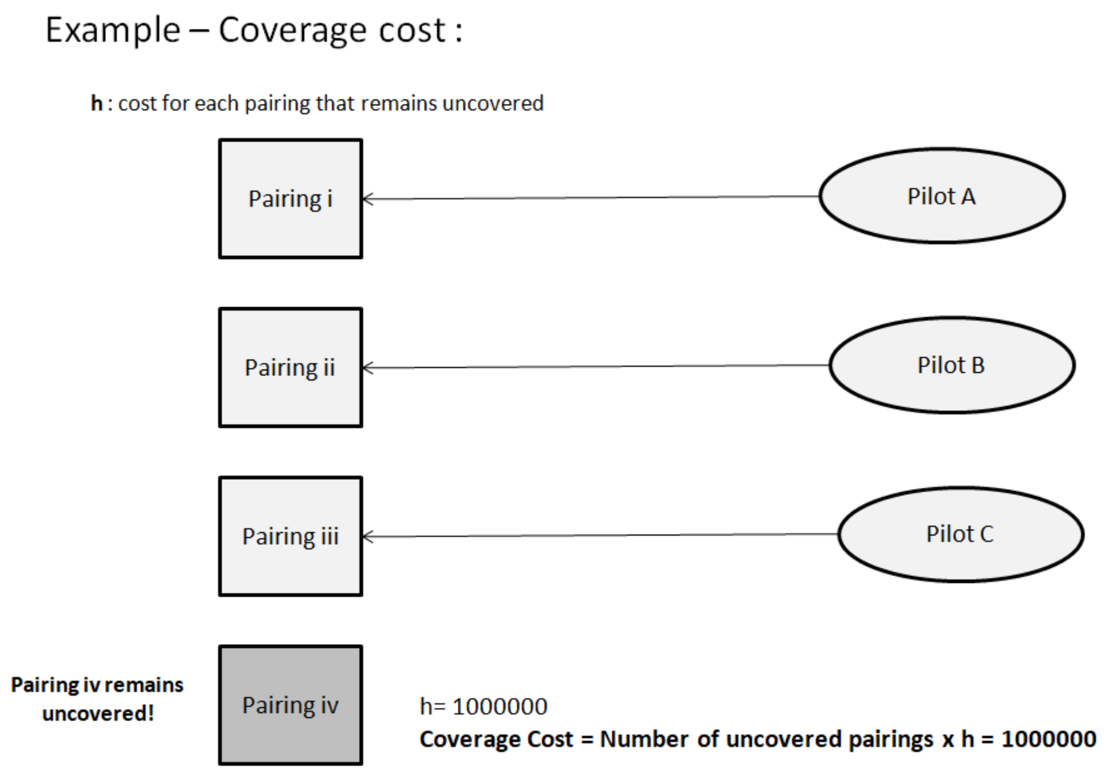
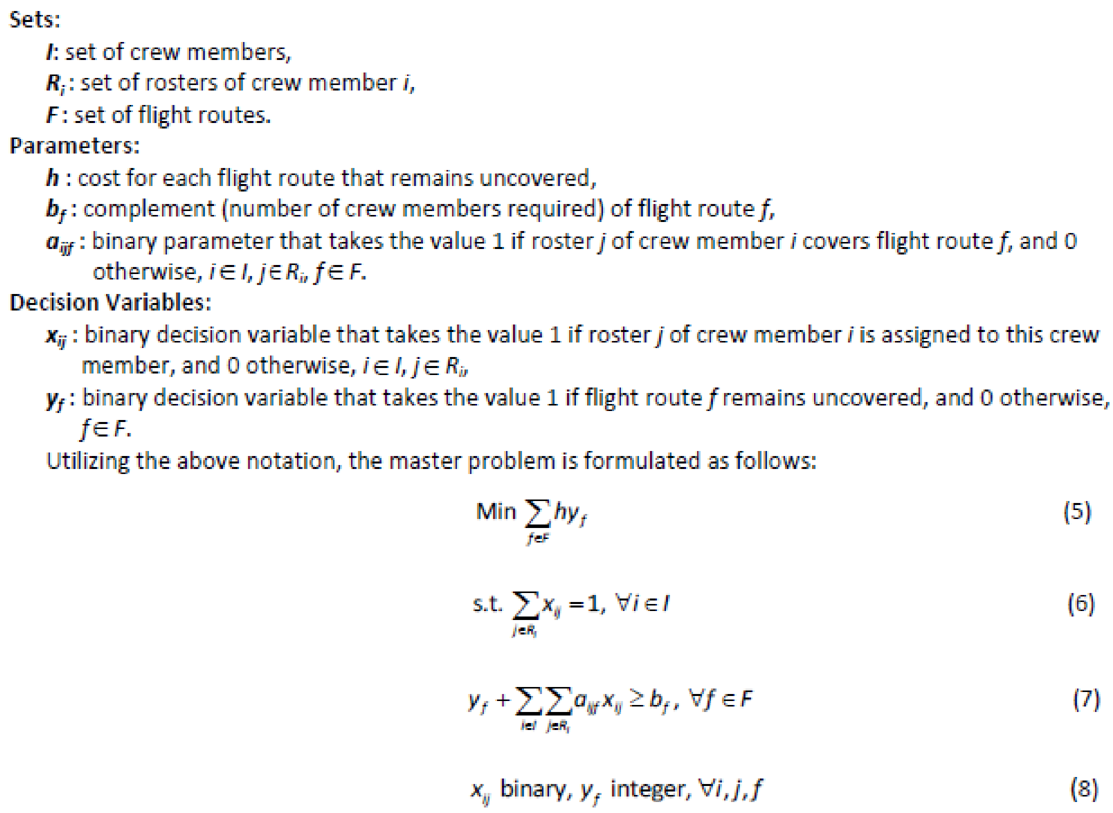

Example – Coverage cost :
h : cost for each pairing that remains uncovered
Pairing i
Pilot A
Pairing ii
Pilot B
Pairing iii
Pilot C
Pairing iv remains uncovered!
Pairing iv
h= 1000000
Coverage Cost = Number of uncovered pairings x h = 1000000

Figure 3.5: Coverage cost - Example

3.1.3 Set-Cover Formulation

At this point we present the set-cover formulation (see Figure 3.6).



Sets:

I: set of crew members,

R_i: set of rosters of crew member i,

F: set of flight routes.

Parameters:

h : cost for each flight route that remains uncovered,

b_f : complement (number of crew members required) of flight route f,

a_{ijf} : binary parameter that takes the value 1 if roster j of crew member i covers flight route f, and 0 otherwise, $i \in I, j \in R_i, f \in F$.

Decision Variables:

x_{ij} : binary decision variable that takes the value 1 if roster j of crew member i is assigned to this crew member, and 0 otherwise, $i \in I, j \in R_i$,

y_f : binary decision variable that takes the value 1 if flight route f remains uncovered, and 0 otherwise, $f \in F$.

Utilizing the above notation, the master problem is formulated as follows:

$$\text{Min} \sum_{f \in F} h y_f \tag{5}$$

$$\text{s.t.} \sum_{j \in R_i} x_{ij} = 1, \ \forall i \in I \tag{6}$$

$$y_f + \sum_{i \in I} \sum_{j \in R_i} a_{ijf} x_{ij} \geq b_f, \ \forall f \in F \tag{7}$$

$$x_{ij} \text{ binary}, \ y_f \text{ integer}, \ \forall i, j, f \tag{8}$$

Figure 3.6: Set-cover formulation

There are two differences between this formulation and the set-partition one.

The first is in the objective function (5). In the set-cover formulation only the coverage cost is included, while the roster quality cost is intentionally omitted.

The second difference is in constraint set (7). Instead of equality, here, we utilize inequality constraints ($\geq$) . We remind that this set of constraints regards the coverage of pairings. When equality, in set-partition formulation, these constraints state that the number of crew members covering each pairing should be equal to its complement b_f. However, in the set-cover formulation, the sum of crew members that cover each flight route is allowed to be greater than the required number.

The optimal solution identified with this model results in the best possible flight coverage achievable with the given set of crew, although some flight routes may be covered more times than the corresponding complement.

Anticipated Benefits of the Set-Cover Formulation

These two differences may seem minor. Nevertheless, they have a major impact to the program, since they expedite the procedure for maximizing crew utilization. We anticipate that, by removing the roster quality cost from the objective function and by allowing the over-coverage of pairings, the column generation procedure will be completed much faster. As we will explain in the following chapter, column generation requires many computational resources and accounts for much of the total solution time. Therefore, its expedition is expected to accelerate the solution procedure substantially.

3.2 Converting a Set-Covering to a Set-Partitioning Solution

3.2.1 Justification of Correctness

As it should be obvious by now, the set-cover solution does not correspond to reality. As explained previously, each flight duty may be assigned to more than the actually required crew members. If this is the case however, then why do we use this formulation to solve the crew rostering problem? Can we treat the set-cover solution in a way that it would acquire practical meaning?

The answer to the second question is yes. With proper treatment, the set-cover solution can be converted to a set-partition one, which does correspond to reality. The treatment includes the removal of over-covered pairings from rosters of excess crew members. To better illustrate we present an example (see Figure 3.7). In our case, Pairing ii and iii are covered by two pilots, whereas only one is needed (complement=1). To turn this solution to a set-partition one, we delete these pairings from the rosters of excess pilots. Pairing ii is removed from the schedule of Pilot A and will be operated only by Pilot B. Pilot A will just cover Pairing i after the solution treatment. In the same way, we remove Pairing iii from the roster of Pilot C. This pairing will now be operated by Pilot B, alone. As a result, the only pairing consisting the schedule of Pilot C, finally, is iv.

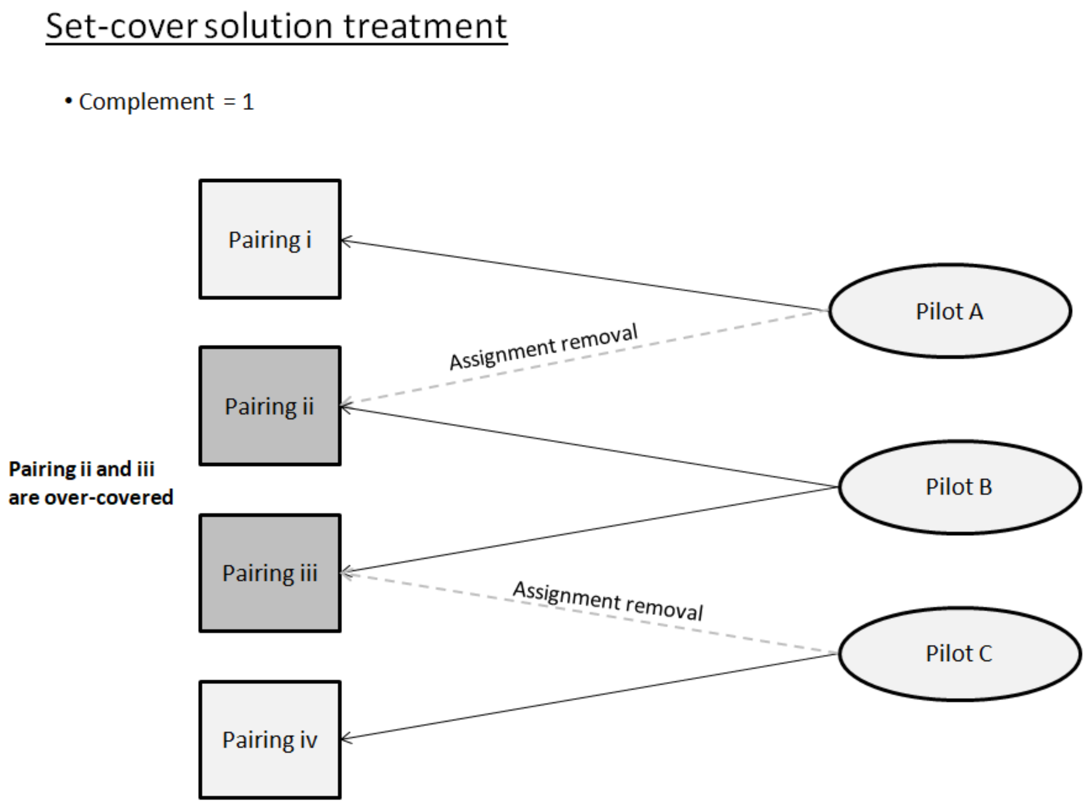


Figure 3.7: Set-cover solution treatment

The key fact that removing flight duties from a legal crew roster does not negate its legality or its feasibility justifies the correctness of our approach.

Legality

The first one stems from the fact that removing flight route(s) from a legal roster always results in a legal roster; in other words, removing one or more flight routes from a legal roster can not introduce violations. Roster-associated work rules, are not limited to but, typically include:

- An upper limit of flight hours in a month
- A minimum number of rest days between pairings
- An upper limit of time the crew may be away from their home base

By removing a pairing from a roster, we reduce the monthly flight load of the crew member, allowing more rest days, and thus no rules are violated.

Feasibility

The second one stems from the fact that removing flight routes from a roster can not lead to an inoperable roster for the corresponding crew member, a consequence of the fact that each flight route is always cyclic, i.e. its first flight departs from the crew member's home base and after various flights between, its last arrives at the base again. If we remove a pairing, the crew member would still be in the base after the completion of a preceding pairing, from where he/she must start his/her next duty (see Figure 3.8).

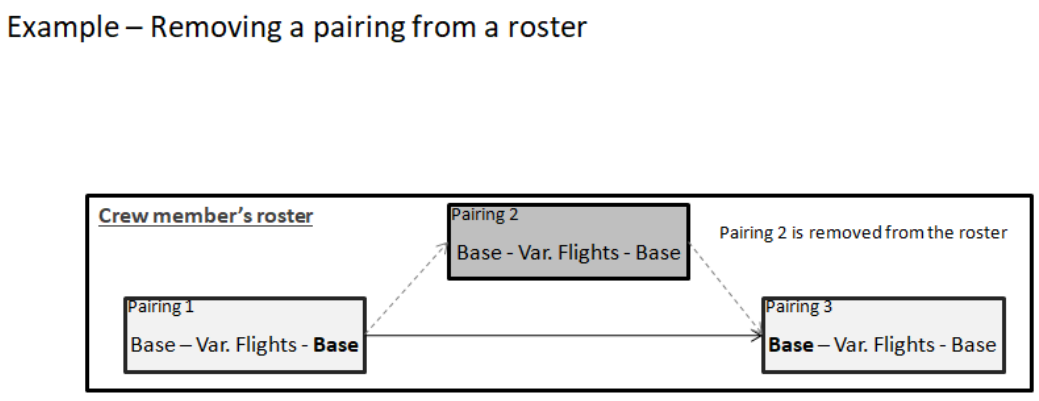
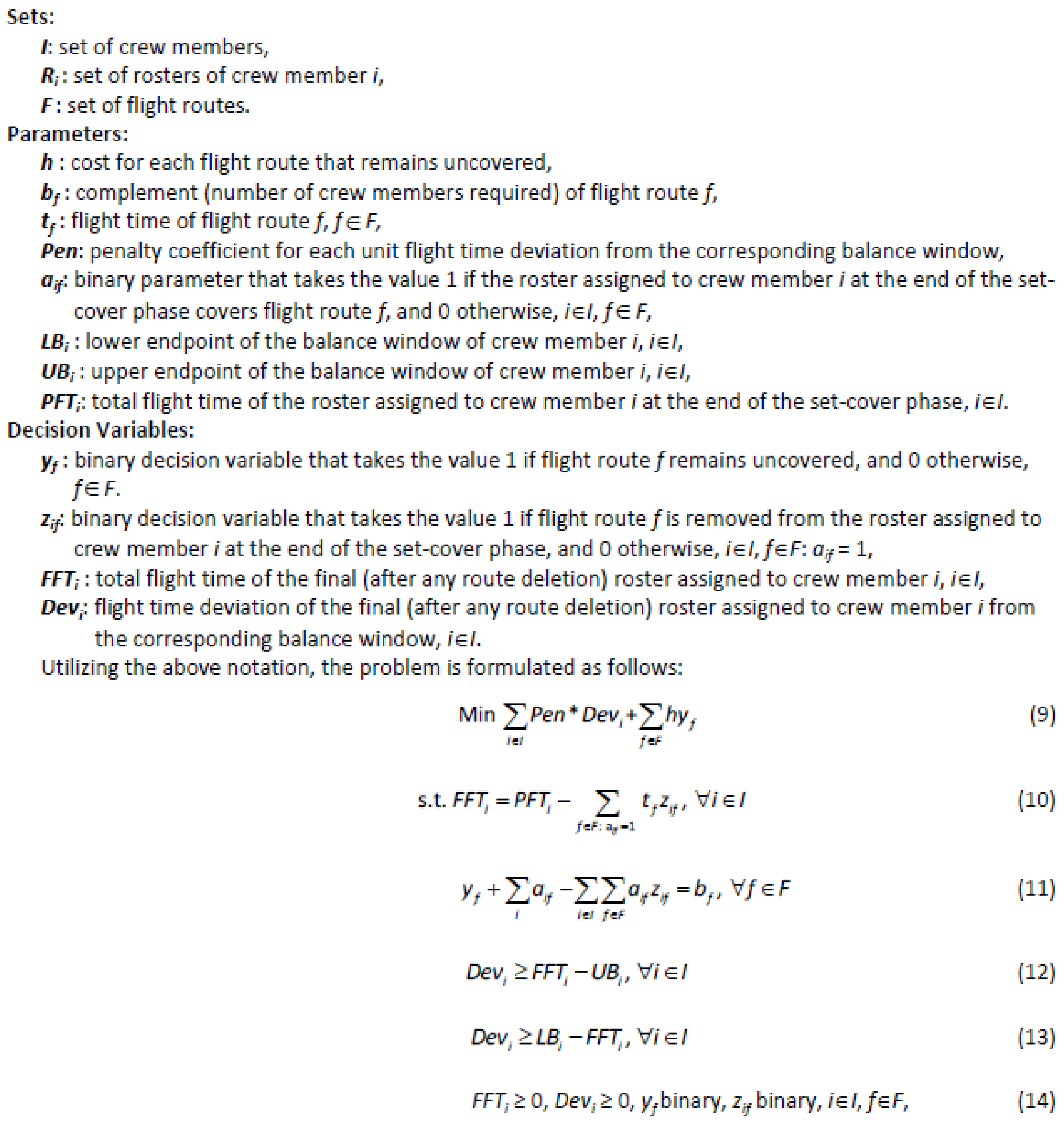

Example – Removing a pairing from a roster
Crew member's roster
Pairing 1
Base – Var. Flights - Base
Pairing 2
Base - Var. Flights - Base
Pairing 2 is removed from the roster
Pairing 3
Base – Var. Flights - Base

Figure 3.8: Removing a pairing from a roster

3.2.2 The "Converting" Optimization Model

In our program, we utilize an optimization model to efficiently delete over-covered pairings from crew members' rosters (see Figure 3.9). To formulate this model we use the solution of the set-cover problem. The product of this model is the final solution of the set-cover phase.



Sets:

I: set of crew members,
R$_i$: set of rosters of crew member *i*,
F : set of flight routes.

Parameters:

h : cost for each flight route that remains uncovered,
b$_f$: complement (number of crew members required) of flight route *f*,
t$_f$: flight time of flight route *f*, $f \in F$,
Pen: penalty coefficient for each unit flight time deviation from the corresponding balance window,
a$_{if}$: binary parameter that takes the value 1 if the roster assigned to crew member *i* at the end of the set-cover phase covers flight route *f*, and 0 otherwise, $i \in I$, $f \in F$,
LB$_i$: lower endpoint of the balance window of crew member *i*, $i \in I$,
UB$_i$: upper endpoint of the balance window of crew member *i*, $i \in I$,
PFT$_i$: total flight time of the roster assigned to crew member *i* at the end of the set-cover phase, $i \in I$.

Decision Variables:

y$_f$: binary decision variable that takes the value 1 if flight route *f* remains uncovered, and 0 otherwise, $f \in F$.
z$_{if}$: binary decision variable that takes the value 1 if flight route *f* is removed from the roster assigned to crew member *i* at the end of the set-cover phase, and 0 otherwise, $i \in I$, $f \in F$: $a_{if} = 1$,
FFT$_i$: total flight time of the final (after any route deletion) roster assigned to crew member *i*, $i \in I$,
Dev$_i$: flight time deviation of the final (after any route deletion) roster assigned to crew member *i* from the corresponding balance window, $i \in I$.

Utilizing the above notation, the problem is formulated as follows:

$$\text{Min} \sum_{i \in I} Pen * Dev_i + \sum_{f \in F} h y_f \tag{9}$$

$$\text{s.t. } FFT_i = PFT_i - \sum_{f \in F:\, a_{if}=1} t_f z_{if}, \ \forall i \in I \tag{10}$$

$$y_f + \sum_i a_{if} - \sum_{i \in I} \sum_{f \in F} a_{if} z_{if} = b_f, \ \forall f \in F \tag{11}$$

$$Dev_i \geq FFT_i - UB_i, \ \forall i \in I \tag{12}$$

$$Dev_i \geq LB_i - FFT_i, \ \forall i \in I \tag{13}$$

$$FFT_i \geq 0,\ Dev_i \geq 0,\ y_f \text{binary},\ z_{if} \text{binary},\ i \in I,\ f \in F, \tag{14}$$

Figure 3.9: The optimization model converting the set-cover to a set-partition solution

The objective function (9) minimizes the total cost of the final solution, which consists of the cost of uncovered duties plus the balance cost of the final rosters. Constraint set (10) computes the flight time of the roster that will be finally assigned to crew member i, based on the pairings that will be removed from the roster he/she was assigned at the end of the set-cover phase. Constraint set (11) ensures that the total number of times each pairing is covered at the final solution is exactly equal to the corresponding complement. For each crew member i, constraint sets (12) and (13) are used to calculate the total flight time deviation from the associated balance window after any pairing removal from his/her roster. Finally, constraint set (14) imposes non-negativity to decision variables FFT_i and Dev_i, and integrality to decision variables y_f and z_{if}. Note that PFT_i is a parameter since the total flight time of the roster assigned to crew member i in the set-cover solution is known, while FFT_i is a decision variable, since his/her final flight time will vary depending on which flight routes are removed from his/her roster. For a given crew member i, at most one of the two right hand sides of constraints (8) and (9) is strictly positive, while both of them are non-positive when the total flight time of his/her final roster is within his balance window, i.e., when $\mathrm{LB_i} \leq \mathrm{FFT_i} \leq \mathrm{UB_i}$. If a pairing is over-covered in the set-cover solution, then constraints (11) ensure that this pairing will be removed from some rosters appropriately, so that the complement is satisfied exactly at the end. Otherwise, no deletion of this pairing from any roster will take place. On the other hand, if a pairing is uncovered in the set-cover solution, it will remain uncovered at the final solution, too. Finally, note that Dev_i is an auxiliary decision variable, since its value is determined automatically by the associated decision variable FFT_i.

3.2.3 A small case study

To recapitulate, this model dictates which pairings should be removed from each crew member's schedule and provides the final schedule cost, after the appropriate pairing deletions. To illustrate we present a solution provided by the computer program we developed. In our case there are 3 crew members and 15 flight routes or pairings. In Figure 3.10 we demonstrate the set-cover solution of our case and in Figure 3.11 we present the solution of the "converting" model. One can notice that in the set-cover solution, flight routes 7 and 10 are covered both by Crew Member 0 and Crew Member 1, while their complement is one. Another thing to notice is that, since all the pairings are covered, the objective value of the set-cover model equals to zero. Remember that in the objective function of the set-cover formulation only the coverage cost is taken into consideration. Since the complement of all flight routes equals to one, they should be covered by just one crew member. As a result, each of pairings 7 and 10 must be removed either from Crew Member's 0 or Crew Member's 1 rosters. This is done by the "converting" model. Furthermore, in this model, the final total cost of the schedule is offered, after the deletion of pairings is done, by calculating both the coverage and the roster quality cost. In our case, as stated previously, all pairings are covered, thus the coverage cost is zero at the end.

In the "converting" model solution, both Pairing 7 and Pairing 10 are removed from Crew Member's 0 roster. It would be interesting to examine the reasons why both these were deleted from his/her roster and why, for example, one was not removed from Crew Member's 1 and the other from Crew Member's 0. As we explained earlier, the roster quality is determined by a single factor in our model, which is the deviation of the actual flight hours of a crew member from a balance window. This window has the target value of flight hours for this crew member at its center. Remember that the target value may differ between crew members. If the actual flight hours of a crew member are within his/her balance window, then his/her roster quality cost is zero. If, however, they are out of its bounds, then for each hour of deviation a penalty is imposed. In our example, we display the bounds that define the balance window, as Lower Bound and Upper Bound. In the set-cover solution, it becomes obvious that the roster that deviates the most from the balance window is Crew Member's 1, since the actual flight hours in this are 98, while the upper bound of the window is just 76. That is 22 flight hours deviation. Crew Member's 0 deviation is just 4 hours and Crew Member's 2, 6. As a result, the roster that would cost the most is Crew Member's 1. The "converting" model should therefore try to make this roster lighter, instead of Crew Member's 0, in its quest to turn the set-cover solution to a set-partition one. As we mentioned previously, crew members 0 and 1 share pairings 7 and 10. Hence, the "converting" model removes both Pairing 7 and Pairing 10 from the roster of this member, which actually means that these two routes will be finally operated by Crew Member 0. After this modification, which can be seen in the "converting" model's solution, the actual flight hours of Crew Member 1 are 82, which is 16 hours less than before the removals. Now, his/her deviation from the upper bound is just 6 hours. The rosters of crew members 0 and 2 remain unchanged in the final solution. After the deletions, the total deviation of all crew members from their respective windows is 16 hours and therefore the total cost, in the absence of an uncovered duty, is this value multiplied by the penalty per hour of deviation which is set to 1000 in our case, resulting in 16000. In the final solution, each flight route or pairing is assigned exactly to a single crew member, as it should, since their complement is one. Finally, we would like to state once again that the changes were made in this way because this was the most cost effective way to delete pairings, so that each is assigned to just one crew member.

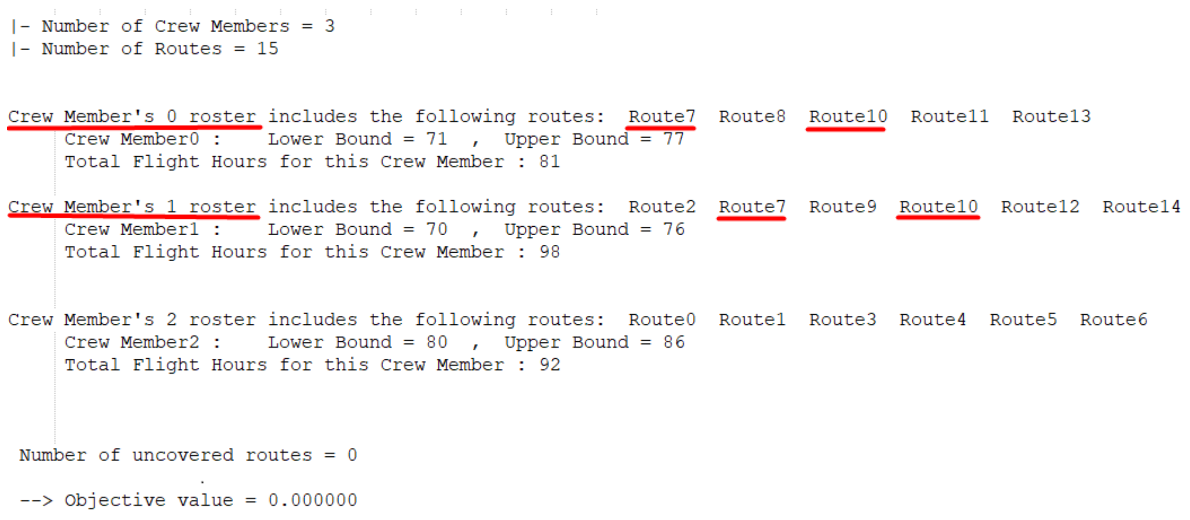
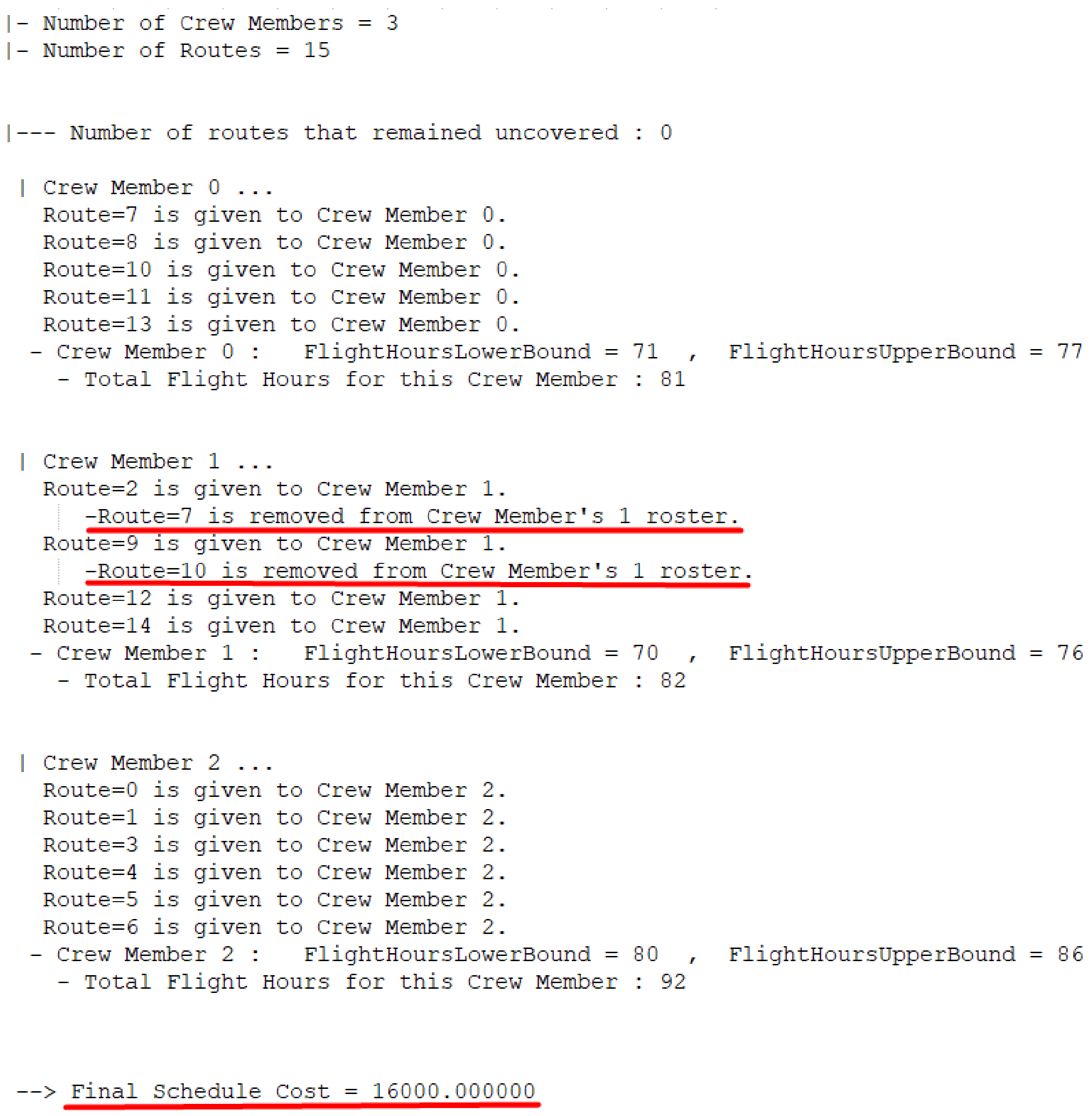

```
|- Number of Crew Members = 3
|- Number of Routes = 15


Crew Member's 0 roster includes the following routes:  Route7   Route8   Route10   Route11   Route13
     Crew Member0 :     Lower Bound = 71   ,  Upper Bound = 77
     Total Flight Hours for this Crew Member : 81

Crew Member's 1 roster includes the following routes:  Route2   Route7   Route9   Route10   Route12   Route14
     Crew Member1 :     Lower Bound = 70   ,  Upper Bound = 76
     Total Flight Hours for this Crew Member : 98


Crew Member's 2 roster includes the following routes:  Route0   Route1   Route3   Route4   Route5   Route6
     Crew Member2 :     Lower Bound = 80   ,  Upper Bound = 86
     Total Flight Hours for this Crew Member : 92



 Number of uncovered routes = 0

 --> Objective value = 0.000000
```

Figure 3.10: Set-cover solution: An example



```
|- Number of Crew Members = 3
|- Number of Routes = 15


|--- Number of routes that remained uncovered : 0

 | Crew Member 0 ...
   Route=7 is given to Crew Member 0.
   Route=8 is given to Crew Member 0.
   Route=10 is given to Crew Member 0.
   Route=11 is given to Crew Member 0.
   Route=13 is given to Crew Member 0.
  - Crew Member 0 :   FlightHoursLowerBound = 71  ,  FlightHoursUpperBound = 77
    - Total Flight Hours for this Crew Member : 81


 | Crew Member 1 ...
   Route=2 is given to Crew Member 1.
     -Route=7 is removed from Crew Member's 1 roster.
   Route=9 is given to Crew Member 1.
     -Route=10 is removed from Crew Member's 1 roster.
   Route=12 is given to Crew Member 1.
   Route=14 is given to Crew Member 1.
  - Crew Member 1 :   FlightHoursLowerBound = 70  ,  FlightHoursUpperBound = 76
    - Total Flight Hours for this Crew Member : 82


 | Crew Member 2 ...
   Route=0 is given to Crew Member 2.
   Route=1 is given to Crew Member 2.
   Route=3 is given to Crew Member 2.
   Route=4 is given to Crew Member 2.
   Route=5 is given to Crew Member 2.
   Route=6 is given to Crew Member 2.
  - Crew Member 2 :   FlightHoursLowerBound = 80  ,  FlightHoursUpperBound = 86
    - Total Flight Hours for this Crew Member : 92



 --> Final Schedule Cost = 16000.000000
```

Figure 3.11: Converting a set-cover to a set-partition solution: An example

3.3 Column Generation Problem

The column generation problem aims to identify rosters that can improve the solution of the master problem. Such rosters have negative reduced cost with respect to the current master LP solution. If the reduced cost of a roster is negative, this is an indication that the associated roster has the potential to improve this solution; therefore, it is added to the master problem, causing an update of the dual solution. If not, this is an indication that this roster can not improve the solution; hence it is not inserted to master problem. The column generation problem is modeled as a shortest path problem with side constraints.

3.3.1 The Crew Member - Pairing Network

For the modeling needs of the column generation problem, we consider a network G = V, A. Each network path in G corresponds to a roster. A very simple network is presented in Figure 3.12. The set of vertices, V, of network G includes one node for each crew member (CM_i), one node for each pairing (P_f), as well as one node, which acts as the terminal node (Fict.). This last node is fictitious, since it does not correspond to a real entity but rather represents the ending of the crew member's schedule for the current planning horizon. The set of arcs, A, comprises of edges connecting CM-P ([A]) and P-P ([B]) compatible node pairs, as well as a set of edges connecting the pairing nodes with the terminal node ([C]). A network path comprising of (CM1-P1), (P1-P2), (P2-Fict.) arcs, corresponds to a roster of Crew Member 1, made of Pairing 1 and Pairing 2.

Network G is a directed acyclic network. Acyclic because it consists of finitely many vertices and edges, with each edge directed from one vertex to another, such that there is no way to start at any vertex v (CM_i) and follow a consistently-directed sequence of edges that eventually leads back to v again. Directed means that its arcs have a direction associated with them. A flow from a pairing node P to a crew member CM is not allowed. Furthermore, flow from a P node (P2) to another P node (P1), with the latter situated on the left of the former, is prohibited (lack of correspondence).

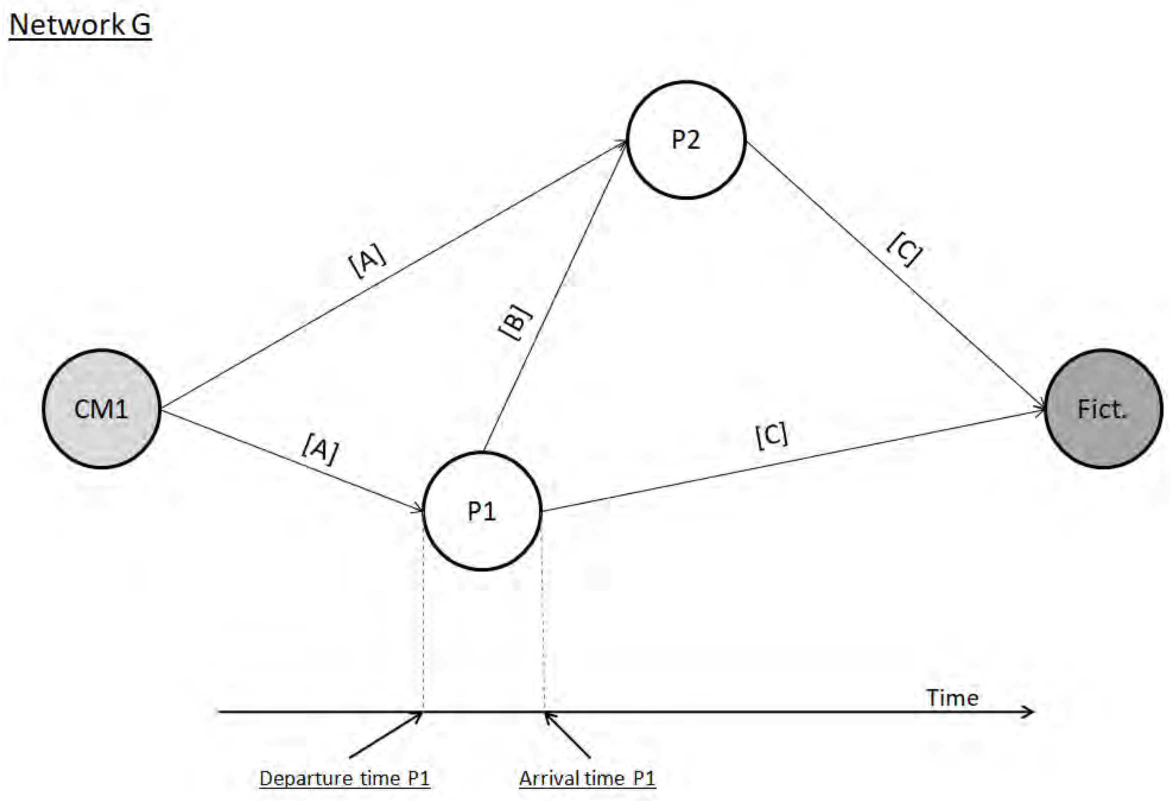


Figure 3.12: A simple Network example

Node Compatibility

For each pairing there is a departure and an arrival time. The departure time coincides with the departure time of the first flight leg of the sequence of legs constituting the pairing and the arrival time with the arrival time of the last flight leg in the sequence. In our example, Pairing 1 starts earlier than Pairing 2. In our network, the pairing nodes are put in a chronological order, from left to right, as shown in Figure 3.12. In other words, network G is a directed graph that has a topological ordering, a sequence of the vertices such that every edge is directed from earlier to later in the sequence.

We mentioned previously that the set of arcs connects compatible node pairs. A pair of P nodes is considered compatible when the arrival time of the first of the two is earlier than the departure time of the second. In Figure 3.13, nodes P1 and P2 are not compatible because, as it is shown, the arrival time of Pairing 1 is later than the departure of Pairing 2. As a result, arc P1-P2 does not exist in our network. The reason why these two pairings are not considered compatible, or in other words, the reason why they can not be both included in the same roster, is because a single crew member can not operate them both, due to lack of correspondence between them.

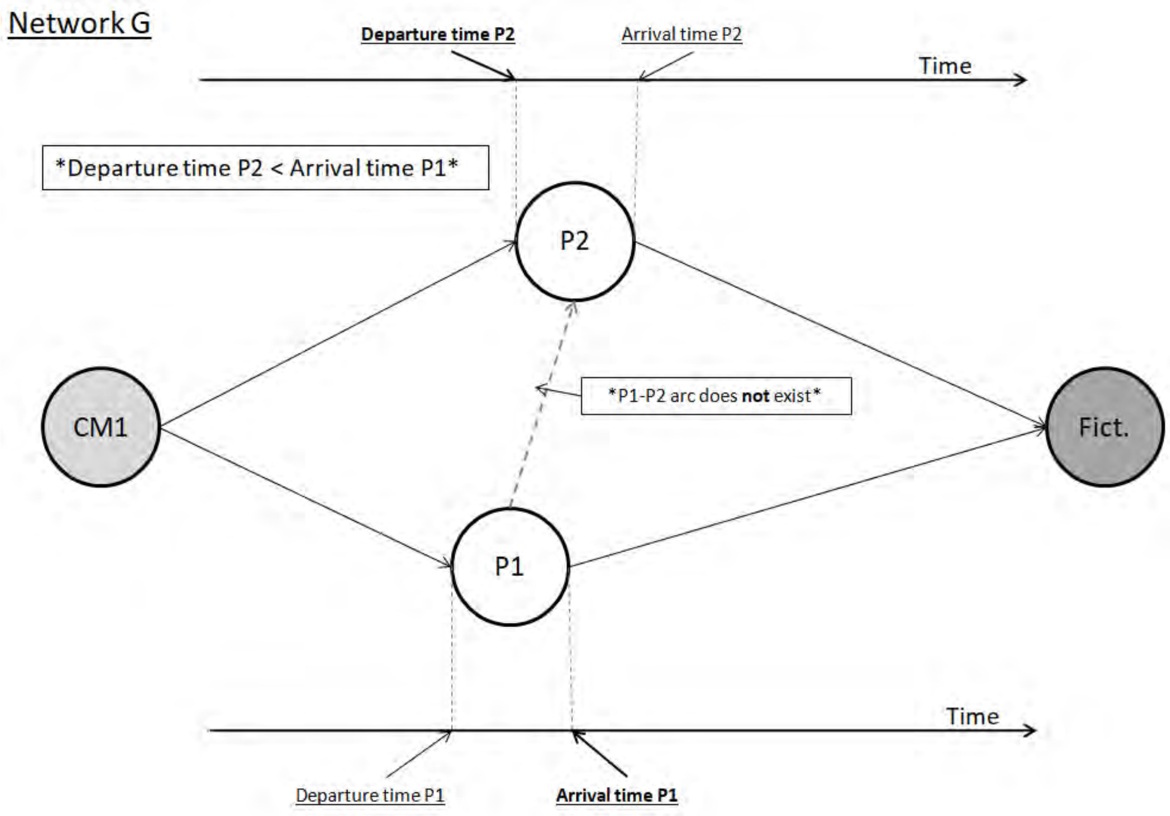


Figure 3.13: Pairing nodes' compatibility

3.3.2 Column Generation Problem: A Shortest Path Problem with Side Constraints

In most branch & price frameworks, the master problem of the column generation method is a set partitioning or set covering problem, where most of the variables, if not all, are associated with vehicle routes or crew schedules. These route and schedule variables are generated by a sub-problem (i.e. column generation problem), corresponding to a shortest path problem with resource constraints (SPPRC). The SPPRC has contributed to the success of the column generation method. One reason behind its contribution is that, through its resource constraints, it constitutes a flexible tool for modeling complex sets of rules that define the feasibility or legality of a route or a schedule. Another reason is that, there exist efficient algorithms that deal with some important variants of the SPPRC. Such algorithms aim to find a shortest path among all paths that start from a source node (a CM node is our case), end at a sink node (Fict. in our case). Each path is required to satisfy a set of constraints defined over a set of resources. A resource corresponds to a quantity. For example time, the load transferred by a vehicle or the duration of a rest interval in a work shift. These quantities change along a path according to functions, called resource extension functions (REFs). A REF is defined for every arc in the graph and for every resource considered. It imposes a lower bound on the value that the associated resource can take at the head node of the corresponding arc, taking into account the values assumed by all the resources at its tail node.[13]

The column generation problem takes as input a network of crew members and pairings, with information about these (i.e. Crew members' balance window, their history of flights, pairings' departure and arrival time), a set of work rules that apply to the construction of rosters and information included in the optimal dual solution of the associated master problem. The output of this problem is "promising" crew roster(s), in the sense that it (they) might improve the master problem's solution.

Work Rules

The side constraints are the work rules that determine which roster (or path in network G) is legal (or feasible). The work rules included in our program are:

(1) *Maximum flight load*: Maximum total flight hours that a crew member is permitted to fly in the planning horizon (month).
(2) *Minimum Days Off*: Minimum days in the planning horizon that a crew member is resting in his/her home base.
(3) *XHoursBreak / YFlightHours*: This rule demands that a crew member has X hours of rest in his/her home base, a break without a flight in between, each time he/she completes Y hours of flight.

It is also important to define the off days. In our case, [A] there must be a minimum hours break between consecutive duties, in order to be considered a day off. Moreover, [B] the latest arrival time of the first of two consecutive pairings so that the next day of the first pairing can be considered a day off and the earliest departure time of the second of two consecutive pairings so that the previous day of the second pairing can be considered a day off are defined. We present an example below (see Figure 3.14).

In our example, the crew member must rest at least for 34 hours in his/her home base, between two consecutive flight duties to be considered that he/she took a day off. After these 34 hours, each 24 hours of rest in home base are considered a day off (see [A] in Figure 3.14). Also in our example, if the arrival time of Pairing 1 is later than 23:00 in Day 1, then Day 2 can not be considered a day off for this crew member. Furthermore, if the departure time of Pairing 2 is earlier than 06:00 in Day 4, then Day

3 can not be considered a day off. It is interesting to note that if Arrival time P1 was for example 23:30 and Departure time P2 was 05:00, then it would be considered that the crew member did not take a day off between these two pairings, even though two whole days lie between these duties.

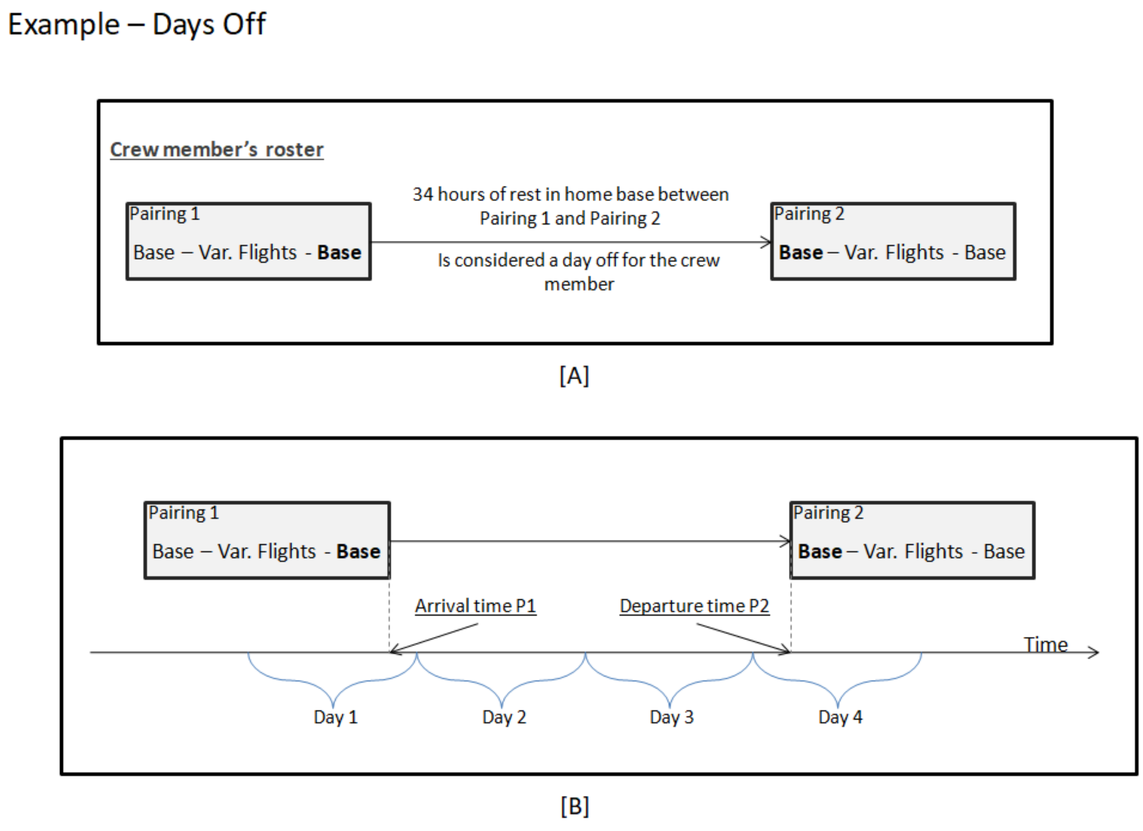

Example – Days Off
Crew member's roster
Pairing 1
Base – Var. Flights - Base
34 hours of rest in home base between Pairing 1 and Pairing 2
Is considered a day off for the crew member
Pairing 2
Base – Var. Flights - Base
[A]
Pairing 1
Base – Var. Flights - Base
Pairing 2
Base – Var. Flights - Base
Arrival time P1
Departure time P2
Time
Day 1
Day 2
Day 3
Day 4
[B]

Figure 3.14: Days Off example

It is important to note at this point that these are only a small subset of the rules that govern roster creation in reality. Although few and simple, these rules effectively provide an outline of the actual regulation scheme. We purposely did not include more rules to keep things simple in this phase of our program's development.

3.3.3 Column Generation Problem's Solution

The aim of the column generation problem is to identify the longest (minimum negative-distance, which means that we actually solve a shortest path problem, to be precise), legal path in this network that begins in one of the crew member nodes, visits at least one pairing node, and ends at the terminal node. The inclusion of at least one pairing is not imposed explicitly, however, since we do not attribute a cost to the utilization of a crew member, while we penalize the actual flight hours deviation from the balance window, the program tends to assign at least one pairing to each member. This is because, if not a single pairing is assigned, the actual flight hours of this crew member would be zero and the deviation from the lower bound of his/her balance window would be great, leading to substantial cost, thus avoided. The length of any path is equal to $-c_{ij} + l_i + \Sigma_{f \in F} d_f$, where i is the index of the crew member associated with the node this path begins from, l_i is the dual value of the corresponding crew member row i, $f \subset F$ is the set of pairing nodes this path visits, and d_f is the dual value of pairing row f in the current master LP optimal solution. Since the cost of each roster is equal to c_{ij}, this is equivalent to finding the roster with the minimum reduced-cost. One important thing to note here is that in the set-cover formulation the roster cost (or roster quality cost) is set to zero. Therefore, when employing the set-cover formulation for our master problem, the length of any path is equal to $l_i + \Sigma_{f \in F} d_f$.

To recapitulate, each time a column generation problem is solved, one (or more as we will see in the next chapter) legal roster, comprised of a set of compatible pairings is generated. This is inserted to the master problem as a decision variable x_{ij}, included in the appropriate crew member and pairing constraints. The aim of the column generation problem is to construct rosters that are "promising", i.e. likely to improve the objective function value of the master problem after their insertion to it. The column generation problem is modeled as a shortest path problem with side constraints that dictate the feasibility of a path in the associated network and is solved using a shortest path algorithm for acyclic networks. More information on the solution method is given in the following chapter.

Chapter 4

Solution Methodology

In this chapter, we present the algorithmic framework utilized in our program. Due to the complexity of our method, at least for a reader unfamiliar with such themes, we provide a gradual presentation. At first, we present the parts of our methodology. Next, we present the specific framework that was used in our program. More specifically, we demonstrate how the parts discussed above are synthesized into an effective solution framework (synthesis) with the help of a small case study, solved by our program. We display a small case, in which a set of 16 flight routes must be covered by a set of 3 crew members and we follow the program step by step, presenting the results at each stage and the final product, which is a monthly schedule for each crew member.

4.1 Solution Methodology: An Outline

We solve the problem with a branch and price algorithm that utilizes an associated branch and price tree. At each node of this tree, the algorithm solves the LP relaxation of the associated master problem with column generation. If the optimal LP solution of this node is non-integral, the algorithm creates two new sub-problems, a left and a right, and their corresponding tree nodes by branching on a fractional decision variable. We solve the LP relaxation of the right sub-problem with column generation and each of these two new sub-problems is added to the set of active nodes; at the same time, the parent node is removed from this set. The program follows the right flow, where a roster is assigned to a crew member, i.e. a crew member-roster variable's value is set equal to 1, because this flow leads to the final solution, i.e. an assignment of a roster to each crew member. The left sub-problems of each branch are created in case a backtrack is required. The procedure continues similarly, until the optimal solution to the LP relaxation of the sub-problem selected next for exploration is integer.[17]

4.2 Components: Solution Method's Parts

4.2.1 Solving the Master LP Relaxation

Each node of the branch and price tree is associated with a unique master problem and its column generation sub-problem, which were described in the previous chapter. The master problem is formulated either as a set-partitioning or as a set-covering problem, depending on the user's preferences. The column generation sub-problem is modeled as a Shortest Path Problem with Side Constraints. The differences between the master and the column generation problems of two distinct tree nodes lie in the fact that in each tree node the assignment of rosters to crew members may differ, as a result of branching. Each time we branch a roster is "fixed" (or assigned) to a crew member by properly modifying the associated constraints of the master problem. Moreover, when a roster is "fixed" to a crew member, this crew member's vertex is essentially deleted from the associated network of the column generation problem. Hence, at different points (nodes) of the search tree, both the formulation of the master problems and the network of the column generation sub-problems differ. These differences will become more clear in the following section, where we present an example.

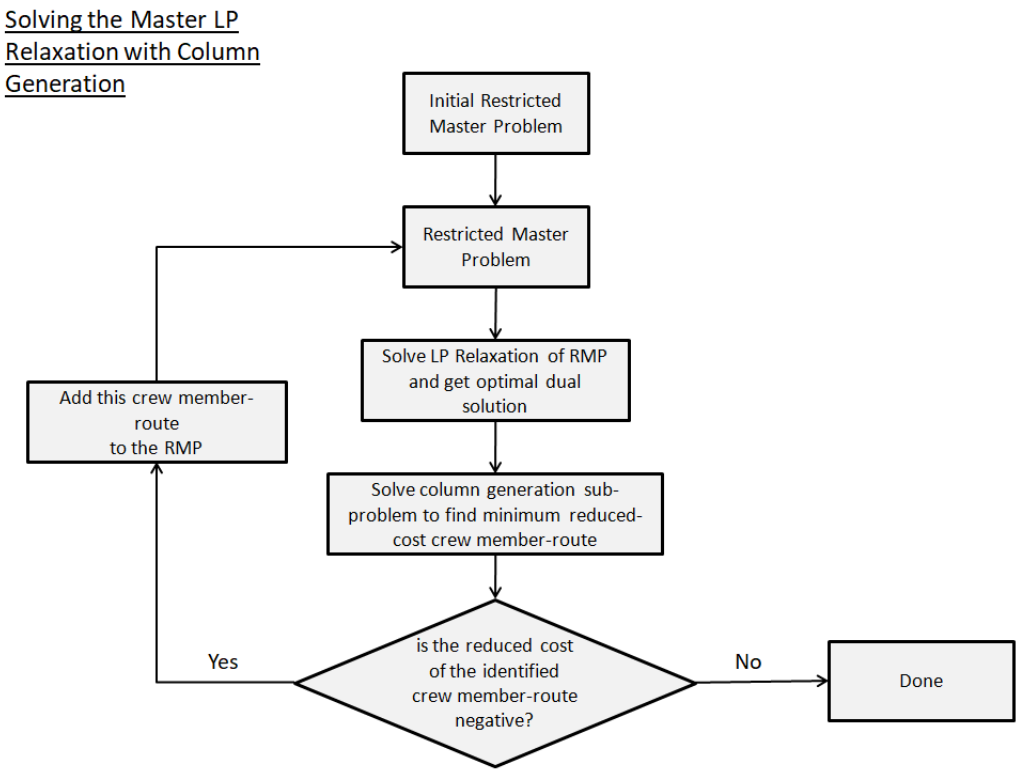

Solving the Master LP Relaxation with Column Generation
Initial Restricted Master Problem
Restricted Master Problem
Solve LP Relaxation of RMP and get optimal dual solution
Solve column generation sub-problem to find minimum reduced-cost crew member-route
is the reduced cost of the identified crew member-route negative?
Yes
No
Add this crew member-route to the RMP
Done

Figure 4.1: Solving the Master LP Relaxation with Column Generation - Flowchart

The optimal solution of each master LP relaxation is obtained with column generation, according to the flowchart presented in Figure 4.1. For the initialization of the algorithm, a feasible solution to the master LP relaxation at the root node of the branch and price tree must be provided. This solution is formed with an empty set of rosters for each crew member and the slack variable y_f of each pairing set equal to 1. Variables y_f are never removed even if they take 0-value, as this could cause feasible solutions to be overlooked. This might happen when the modification of constraints during branching leads to an infeasible master problem (one of the pairings remains uncovered, for example), even though this infeasibility can be repaired through the generation of additional crew member-rosters in the current node. In such cases, variables y_f preserve the feasibility of the model, enabling the continuation of the column generation procedure.

For the solution of the master LP relaxation at each node of the branch and price tree we utilize IBM ILOG CPLEX 12.6 [5], which is a commercial optimization software.

4.2.2 Solving the Column Generation Sub-Problem

As we mentioned previously, we solve the column generation problem with a shortest path algorithm for acyclic networks. The main idea is to initialize first the reduced cost (negative total distance) of each pairing node f to a very large positive value, and then scan the network arcs in topological order (see Figure 4.2), identifying possible path extensions through pairs of compatible nodes, and updating the corresponding distances accordingly.

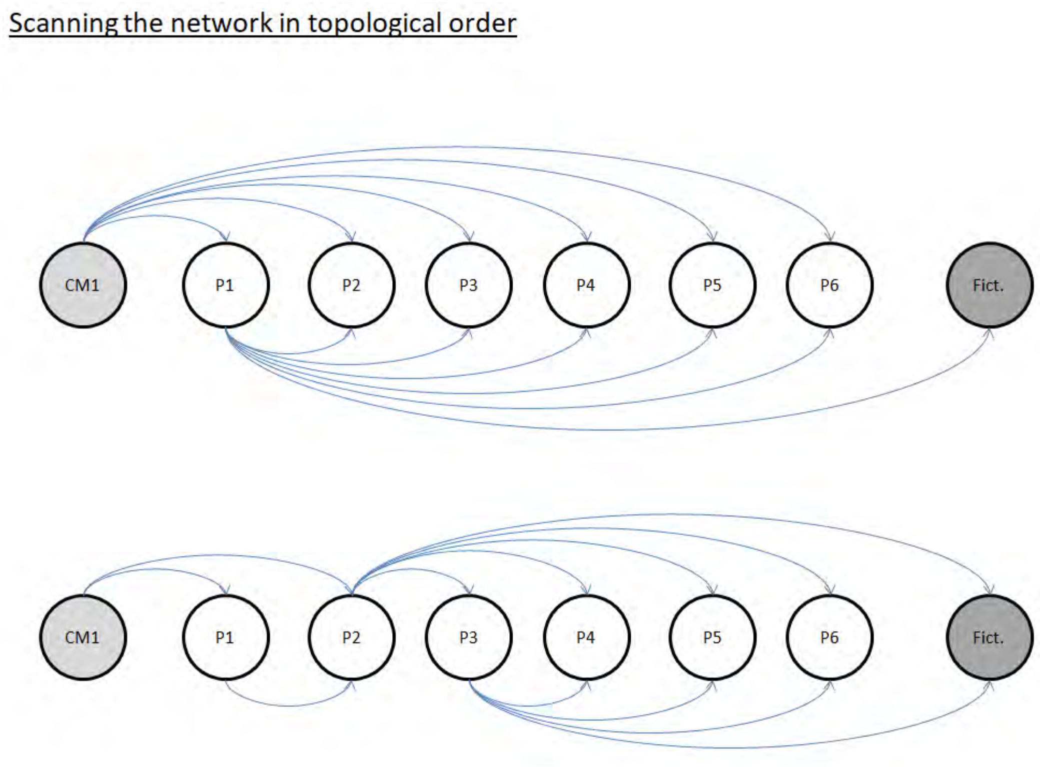


Figure 4.2: Scanning the network in topological order

For each node, a label indicating the reduced cost of the best path ending at that node is stored, which is updated accordingly each time an improved path is found. An example of such an update is illustrated in Figure 4.3. Assume that the flow from node s to node t is feasible (nodes are compatible) and legal, and the path extension from node s to node t is considered. This path extension corresponds to including both these two nodes consecutively on the current path. Suppose that the optimal dual value of node t is equal to 15 and the reduced cost of the best path ending at node s that has been found so far is equal to -20, while that of the best path ending at node t is equal to -30. This entails that, by extending the path from node s to node t, a new path ending at node t is constructed with improved reduced cost equal to -35. As a result, this value replaces the previous one, while the index (prev) of the node from which the best incoming flow to node t comes from is also updated at node t, with the aim to expedite the identification of the optimal path's complete node sequence at the end of the algorithm.[17]

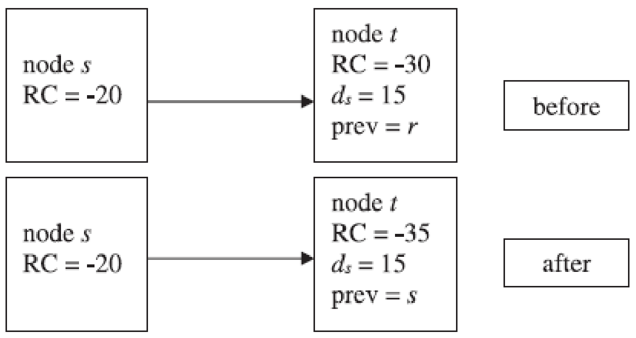


Figure 4.3: Solving the Column Generation Sub-Problem - Shortest Path Identification (Source: [17])

Algorithmic improvements that considerably advance the efficiency of the proposed solution methodology involve the existence of more than one label at each pairing node of the network and the addition of multiple crew member-roster columns with negative reduced-cost to the master problem for each crew member, which actually means that the fictitious node has more than one labels as well. These enhancements enable the solution of the master LP relaxation considerably faster than if there was only a single label at each node or if only the minimum reduced-cost crew member-roster was added in the master formulation. The simultaneous addition of multiple crew member-rosters to the master problem is simple, since the Acyclic Shortest Path Algorithm (ASPA) typically solves the column generation sub-problem separately for each crew member[17]. The number of labels of the pairing nodes and of the fictitious, i.e. the number of roster-variables inserted to master for each crew member after the solution of the column generation sub-problem, are user defined in our program. Typically, the labels of fictitious are more than the labels of the pairing nodes. For example, if the pairing nodes' labels are 3, then each pairing node could be included to 3 paths at the same time, or to put it differently, each pairing node can be a part of 3 under-construction-rosters. If the fictitious' labels are 20, it means that at most 20, the best out of all created (those with the lowest reduced cost) completed paths (or rosters) will be added to the master formulation.

Instead of identifying the optimal roster of each crew member with the same set of dual values, these dual values are updated each time a roster is added to the master problem. However, rather than solving the associated master problem at each point with CPLEX to get the updated dual values, we prefer a different approach that requires far less computational effort. The dual value of a master row covered by a specific crew member-roster added to the master problem is updated "manually" as dualnew = dualold + rc/ξK, where rc is the reduced cost of this roster, K is the number of rows this roster covers, and ξ is a smoothing parameter. Note that, this re-evaluation technique does not necessarily lead to a valid feasible dual solution, but this does not pose a problem, since we are only interested in penalizing the covered rows in order to avoid the generation of similar crew member rosters subsequently.[17]

4.2.3 Branching

When the optimal solution to the master LP relaxation is non-integer, the algorithm branches on a fractional variable in order to gradually lead to an integer solution, by consecutively fixing variables to integer values. Similarly as in the case of a typical branch and bound algorithm, new sub-problems are created through the addition of constraints that eliminate fractional solutions. A typical design involves the selection of one such fractional decision variable and the partition of the solution space by setting this variable equal to 0 and 1. In our program, we branch on the crew member-roster decision variable with the largest fractional value (see Figure 4.4).

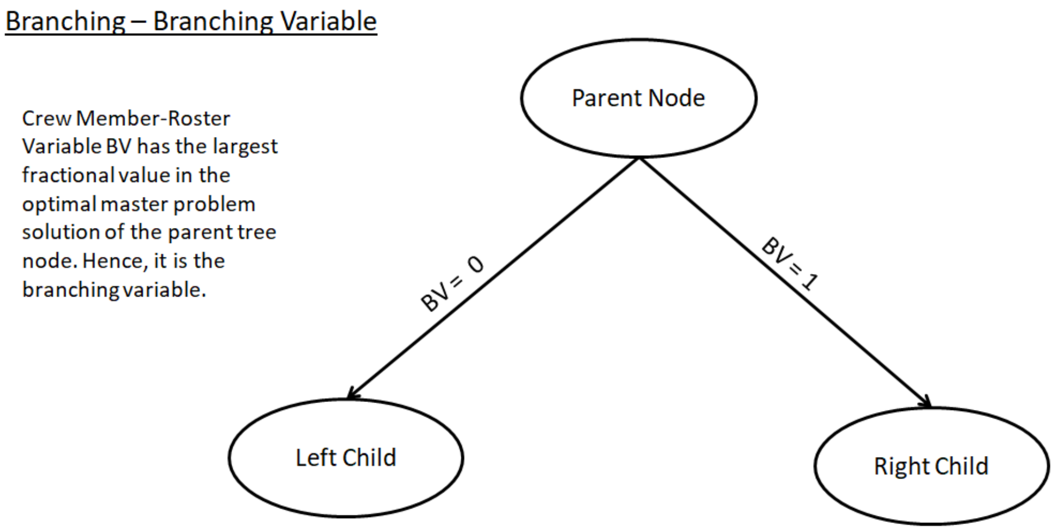

Branching – Branching Variable
Crew Member-Roster Variable BV has the largest fractional value in the optimal master problem solution of the parent tree node. Hence, it is the branching variable.
Parent Node
BV = 0
BV = 1
Left Child
Right Child

Figure 4.4: Branching on a variable: Example

As mentioned earlier, the program follows the right flow, because setting a crew member-roster variable to 1-value is equivalent to assigning a roster to a crew member. Our goal in the crew rostering problem is to assign a roster to each crew member. Thus, an assignment is a progress. In contrast, in the left branches, we do not assign anything and just one roster out of the vast number of all the potential rosters is excluded from the final solution. Hence, by following the right flow we get one step closer to the solution each time (see Figure 4.5).

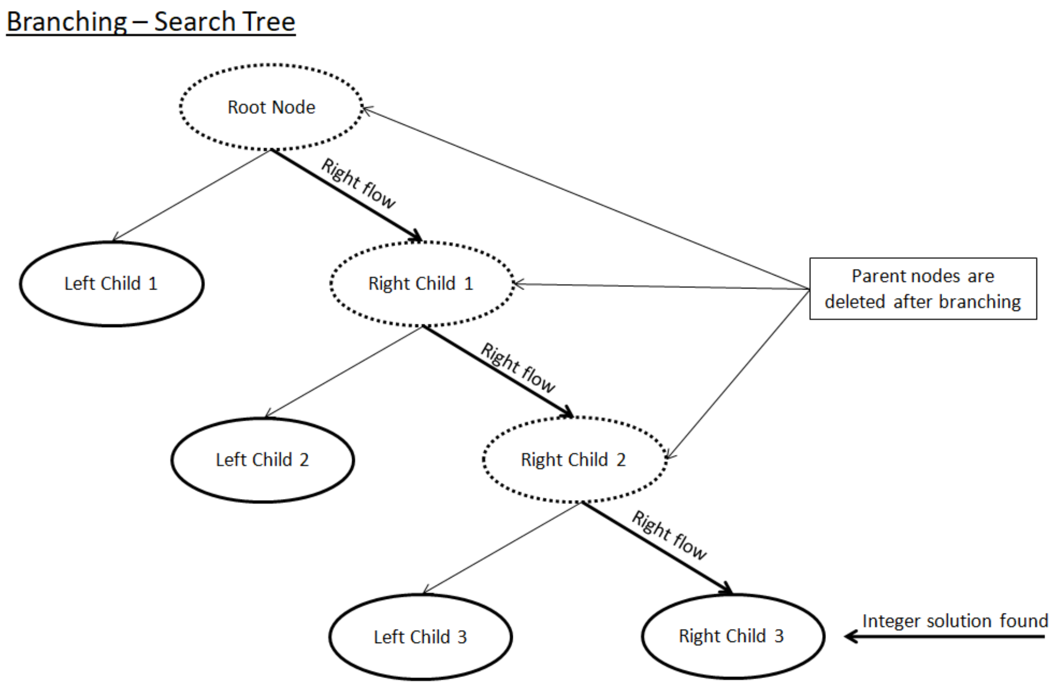


Figure 4.5: Search Tree: Example

Rather than appending the branching constraints to the master problem, we incorporate them directly into the existing formulation instead. The benefit of this is that we keep the same number of master problem constraints, thus avoiding the trouble of dealing with extra dual variables.

In the left sub-problem, the branching variable is set equal to 0. Actually, this variable is deleted from the master problem formulation and a suitable modification of the associated network (of the column generation sub-problem) for this tree node is done, in order to ensure that this particular path (i.e. this particular roster of the associated crew member) will not be generated again in this or in the search tree nodes following this node, which we term left child.

For the modification of the associated network, the algorithm proposed by [19] is utilized. As explained in [17], at each iteration, this algorithm finds the shortest path of the underlying network, and then modifies this network appropriately, so as to eliminate this path, without, however, eliminating any other path. We utilize this elimination procedure in order to incorporate the 0-branch constraints into ASPA. In short, this procedure works as follows. Assume that the node sequence of the path that we want to exclude is $s_1, . . . , s_m$. Note that s_1 is a crew member node and s_m is the terminal node in our case, while m is greater or equal to 3, since no direct flow from a crew member node to the terminal node is expected. The proposed procedure deletes the directed arc from node s_{m-1} to s_m, and adds a new node s'_i for each node i = 2, . . . , m-1 with dual value equal to that of s_i. We call the nodes s_i and s'_i pairing correspondent, because they relate to the same pairing. From each node of the original network which is connected through a directed arc to node i (i = 2, . . . , m-1) and is not pairing correspondent to node s_{i-1}, the algorithm adds a new directed arc to node s'_i. For each node i (i = 2, . . . , m-2), it adds a directed arc from node s'_1 to node s'_{i+1}. Finally, it adds a directed arc from node s'_{m-1} to node s_m. In [17], the reader can also find an illustrative example of the aforementioned algorithm. The figure shown below (Figure 4.6) is derived from this article. In this example the author wants to prohibit the creation of S-1-2-F.

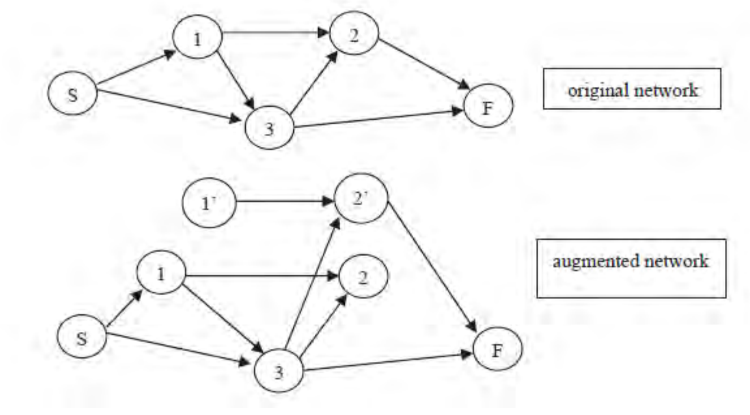


Figure 4.6: Modifying the network to prohibit the creation of a specific path: Example (Source: [17])

In the right sub-problem the branching variable is set equal to 1. This is achieved by deleting all the other roster-variables belonging to the corresponding crew member from the master problem formulation of the parent node, along with all the other roster-variables including a pairing that is covered by the roster-variable on which we branch. Furthermore, suitable, auxiliary side-constraints prohibiting the generation of additional rosters in the associated sub-tree that belong to crew member i or include a pairing which is covered by this particular roster-variable (on which we branched) are added to the network of the column generation sub-problem of the right tree node.

4.2.4 Backtracking

Backtracking is a general algorithm for finding all (or some) solutions to computational, constraint satisfaction problems, that gradually builds candidates to the solutions, and abandons a candidate ("backtracks") as soon as it determines that the candidate cannot possibly be completed to a valid or a satisfactory solution.

In our case, backtrack translates to abandoning a tree node if its master problem after the column generation procedure does not satisfy some conditions defined below, and to going to another tree node, which was created in a previous step, generating columns in its master problem and continuing branching from there until we find an integer solution. Due to our strategy to follow the right flow, in all cases we backtrack to a left child and specifically to the left child with the most rosters assigned (i.e. the left child that is in the lowest position , because this is the closest we can get to an integer solution (see Figure 4.7).

In our program, backtrack is conducted in two conditions. The first is in case of an infeasible master problem. This can happen because of the branching on a variable. Fixing a variable to 1-value may lead to an infeasible master problem in a right-child node. If we encounter this situation then we need to find a node with a feasible master problem to continue our search for an integer solution from there. The other case in which we backtrack is if the objective function of a master problem in a tree node is much worse than the objective function of the lp relaxation of the master problem

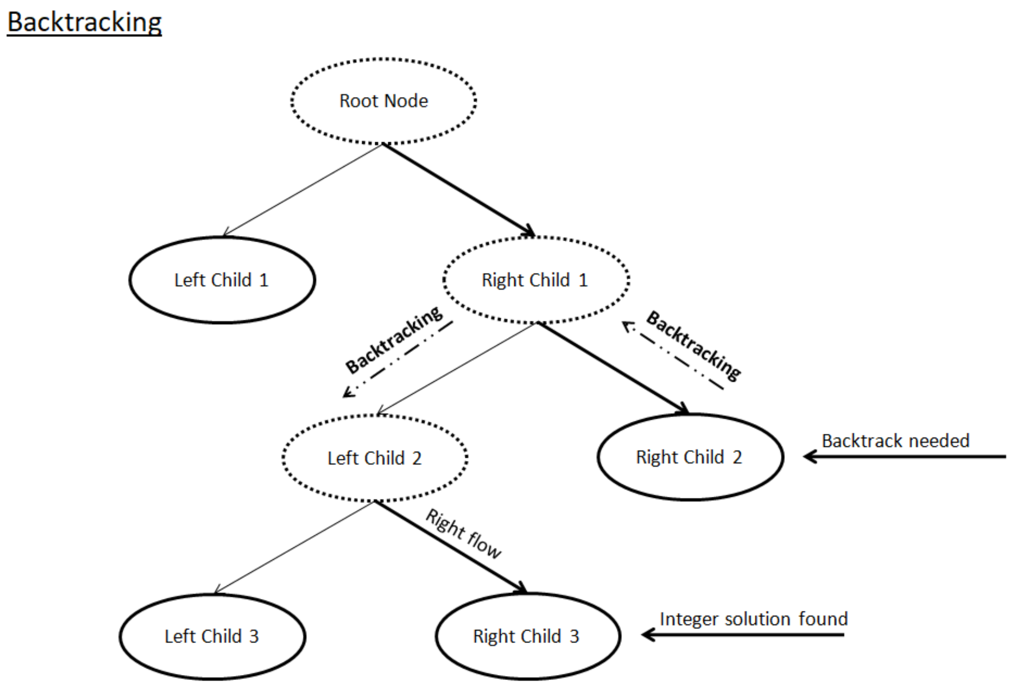


Figure 4.7: Backtracking: Example

found in the root node. We define an absolute and a relative backtrack gap between the objective function of the master problem of a tree node and the objective function of the master problem of the root node. If the absolute and the relative differences between the objective functions of these two nodes are both greater than the defined backtrack gaps, then we backtrack.

4.2.5 Terminating Conditions

MIP Gap Tolerance

Our goal is to find an integer solution, in which each crew member is assigned a roster. However, we do not necessarily terminate the program when one is found. We define an absolute and a relative MIP gap tolerance. If the absolute and the relative difference between the objective function of the integer solution and the objective function of the master problem in the root node are both greater than the MIP gaps defined then we continue our search for a better integer solution, with a smaller objective function. We delete all nodes in the search tree that are not promising, i.e. their lower bound is greater than the objective function of the master problem in the root node, and start our new search from the node that has the most variables fixed to integer values, because this one will most likely lead us faster to an integer solution. If promising nodes do not exist or an integer solution with an objective function within the defined gaps is found, then the program is terminated and we keep the best integer solution found so far.

Maximum Number of Backtracks

If an integer solution has been found, but it is not satisfactory and we continue our search for a better one, then from that point, we begin counting the number of backtracks that are conducted. If this number exceeds an upper limit of backtracks, that we defined at the beginning of the program, before we find an integer solution satisfying the MIP gap tolerances, then we terminate it and we keep the best integer solution found so far.

Maximum Number of Tree Nodes created

Similarly, if we find a non-satisfactory integer solution and we continue our search for a better one, we begin counting the number of tree nodes that the program creates, which is similar to counting the branches. If the maximum number of tree nodes allowed is exceeded, before we find an integer solution satisfying the MIP gap tolerances, then the program is terminated and the best integer solution found that far is provided to the user.

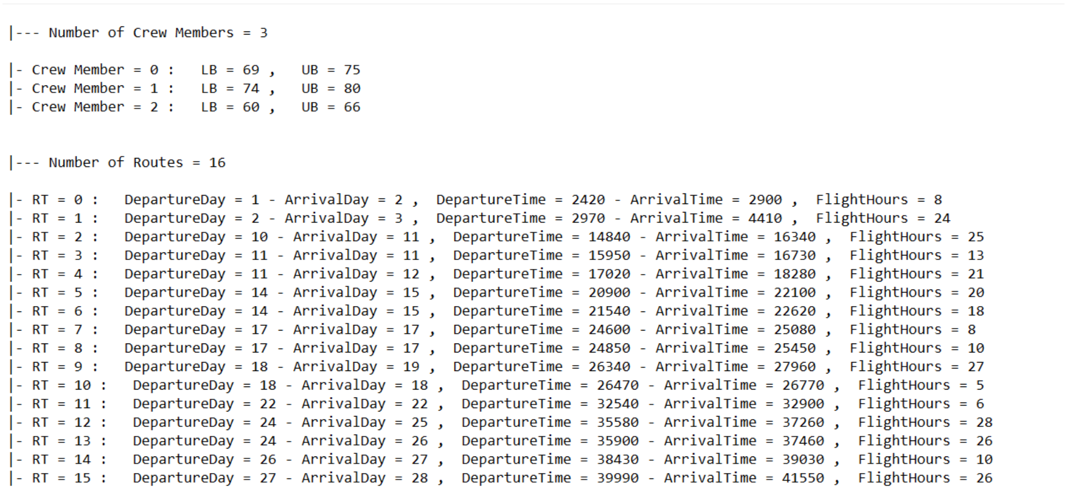
4.3 Synthesis: A Small Case Study

It seems sensible to the author at this point to present a small case study utilizing the program developed during this thesis. It is a tiny problem compared to the real life cases, which are huge, but its small size is ideal for illustration purposes. We choose to present the set-cover formulation, in order to also display the conversion of the solution at the end of the set-cover phase to a set-partition solution. For our case, there are 3 crew members, let us assume that these are pilots, and 16 pairings to be covered.

Input

The program begins with a data generator which produces data for the crew members and for the pairings. This is necessary because we do not possess real data from airlines at this point. As the reader can see in Figure 4.8 each crew member has an ID and a lower (LB) and upper bound (UB). These bounds define the balance window of each crew member defined in Chapter 3. We remind that this window defines the amount of actual flight hours a member can fly in a planning horizon without a balance cost for the airline (if the actual flight hours are within the bounds)(see Figure 3.4). The data generated for pairings is the Departure day and time, the Arrival day and time and the total flight hours of each pairing. We remind at this point that both the departure and the arrival are from and to the base of the crew member set we schedule. We count time in minutes from the beginning of the schedule horizon, when time is set equal to 0. As the reader can see, the pairings are spread throughout the planning horizon, which in our case is a month (30 days). This is the data input to our program.

In reality, the pairings are generated in the first phase of crew scheduling and they respect a large number of regulations. Pairing generation is a very complex problem and has received much attention from operations researchers. For the needs of our program, however, we create pairings randomly and we assume that they are legal.



```
|--- Number of Crew Members = 3

|- Crew Member = 0 :   LB = 69 ,   UB = 75
|- Crew Member = 1 :   LB = 74 ,   UB = 80
|- Crew Member = 2 :   LB = 60 ,   UB = 66


|--- Number of Routes = 16

|- RT = 0 :   DepartureDay = 1 - ArrivalDay = 2 ,  DepartureTime = 2420 - ArrivalTime = 2900 ,  FlightHours = 8
|- RT = 1 :   DepartureDay = 2 - ArrivalDay = 3 ,  DepartureTime = 2970 - ArrivalTime = 4410 ,  FlightHours = 24
|- RT = 2 :   DepartureDay = 10 - ArrivalDay = 11 ,  DepartureTime = 14840 - ArrivalTime = 16340 ,  FlightHours = 25
|- RT = 3 :   DepartureDay = 11 - ArrivalDay = 11 ,  DepartureTime = 15950 - ArrivalTime = 16730 ,  FlightHours = 13
|- RT = 4 :   DepartureDay = 11 - ArrivalDay = 12 ,  DepartureTime = 17020 - ArrivalTime = 18280 ,  FlightHours = 21
|- RT = 5 :   DepartureDay = 14 - ArrivalDay = 15 ,  DepartureTime = 20900 - ArrivalTime = 22100 ,  FlightHours = 20
|- RT = 6 :   DepartureDay = 14 - ArrivalDay = 15 ,  DepartureTime = 21540 - ArrivalTime = 22620 ,  FlightHours = 18
|- RT = 7 :   DepartureDay = 17 - ArrivalDay = 17 ,  DepartureTime = 24600 - ArrivalTime = 25080 ,  FlightHours = 8
|- RT = 8 :   DepartureDay = 17 - ArrivalDay = 17 ,  DepartureTime = 24850 - ArrivalTime = 25450 ,  FlightHours = 10
|- RT = 9 :   DepartureDay = 18 - ArrivalDay = 19 ,  DepartureTime = 26340 - ArrivalTime = 27960 ,  FlightHours = 27
|- RT = 10 :   DepartureDay = 18 - ArrivalDay = 18 ,  DepartureTime = 26470 - ArrivalTime = 26770 ,  FlightHours = 5
|- RT = 11 :   DepartureDay = 22 - ArrivalDay = 22 ,  DepartureTime = 32540 - ArrivalTime = 32900 ,  FlightHours = 6
|- RT = 12 :   DepartureDay = 24 - ArrivalDay = 25 ,  DepartureTime = 35580 - ArrivalTime = 37260 ,  FlightHours = 28
|- RT = 13 :   DepartureDay = 24 - ArrivalDay = 26 ,  DepartureTime = 35900 - ArrivalTime = 37460 ,  FlightHours = 26
|- RT = 14 :   DepartureDay = 26 - ArrivalDay = 27 ,  DepartureTime = 38430 - ArrivalTime = 39030 ,  FlightHours = 10
|- RT = 15 :   DepartureDay = 27 - ArrivalDay = 28 ,  DepartureTime = 39990 - ArrivalTime = 41550 ,  FlightHours = 26
```

Figure 4.8: Crew members and pairings data generated

Apart from the data input, we also insert a number of rules that define the legality of rosters and are used as side-constraints in the column generation problem, where we construct rosters by solving an ASPA. We have already discussed about the rules we inserted in our program in the previous chapter.

Solution Process

As soon as we have the data, we begin the solution process. To start the branch and price algorithm we need an initial restricted master problem, which will be used as a seed to the column generation process taking place in the root node of the search tree (see Figure 4.9).

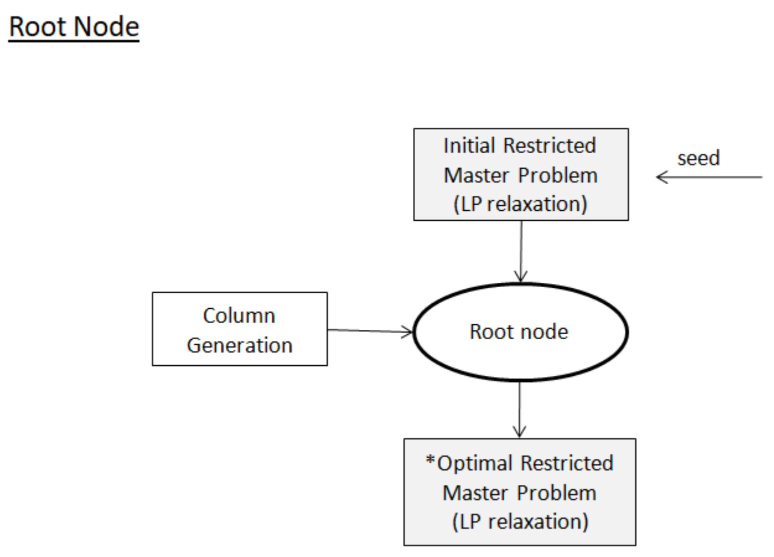
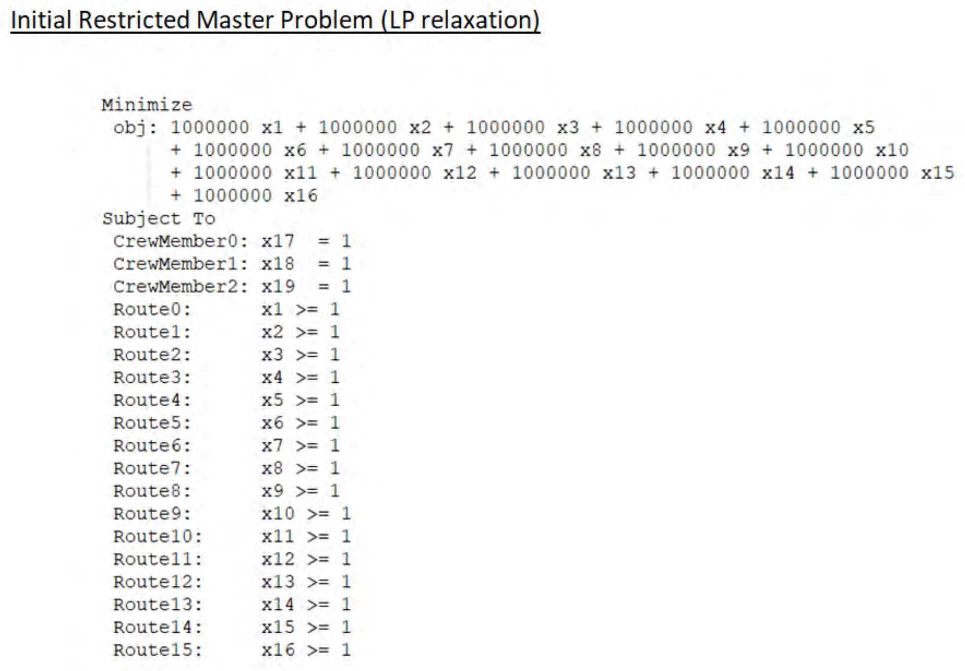

Root Node
Initial Restricted Master Problem (LP relaxation)
seed
Column Generation
Root node
*Optimal Restricted Master Problem (LP relaxation)

Figure 4.9: Starting the solution procedure: Root node

For this reason we start with a set of empty rosters for each crew member and the slack variables of each pairing, which are set equal to 1. In Figure 4.10 variables *x1-x16* are slack variables, which are equal to 1 if the associated pairing (or route) is not covered and *x17-x19* are the empty roster variables. Notice that the cost of an uncovered pairing in the objective function is set equal to 1000000. Also, notice that the complement of each pairing is set equal to 1 since we face the pilot scheduling problem. Finally, as the reader can see, the route constraints are inequalities, since we formulate the master problem as a set-covering model.



Initial Restricted Master Problem (LP relaxation)

```
Minimize
 obj: 1000000 x1 + 1000000 x2 + 1000000 x3 + 1000000 x4 + 1000000 x5
      + 1000000 x6 + 1000000 x7 + 1000000 x8 + 1000000 x9 + 1000000 x10
      + 1000000 x11 + 1000000 x12 + 1000000 x13 + 1000000 x14 + 1000000 x15
      + 1000000 x16
Subject To
 CrewMember0: x17  = 1
 CrewMember1: x18  = 1
 CrewMember2: x19  = 1
 Route0:      x1 >= 1
 Route1:      x2 >= 1
 Route2:      x3 >= 1
 Route3:      x4 >= 1
 Route4:      x5 >= 1
 Route5:      x6 >= 1
 Route6:      x7 >= 1
 Route7:      x8 >= 1
 Route8:      x9 >= 1
 Route9:      x10 >= 1
 Route10:     x11 >= 1
 Route11:     x12 >= 1
 Route12:     x13 >= 1
 Route13:     x14 >= 1
 Route14:     x15 >= 1
 Route15:     x16 >= 1
```

Figure 4.10: Seed problem

Subsequently, the column generation begins building on the initial master problem. As mentioned previously, the ASPA is solved for each crew member separately and the process stops when no more columns with negative reduced cost can be identified (see Figure 4.11). We will present how the column generation procedure builds on the initial restricted master problem gradually, at least for the first iteration of the process, i.e. from the beginning until we encounter "Column added?" block for the first time. *CG ID$_i$ is the "updated" master problem, after the generation of the first pairings for crew member i (see Figure 4.12, Figure 4.13, Figure 4.14).

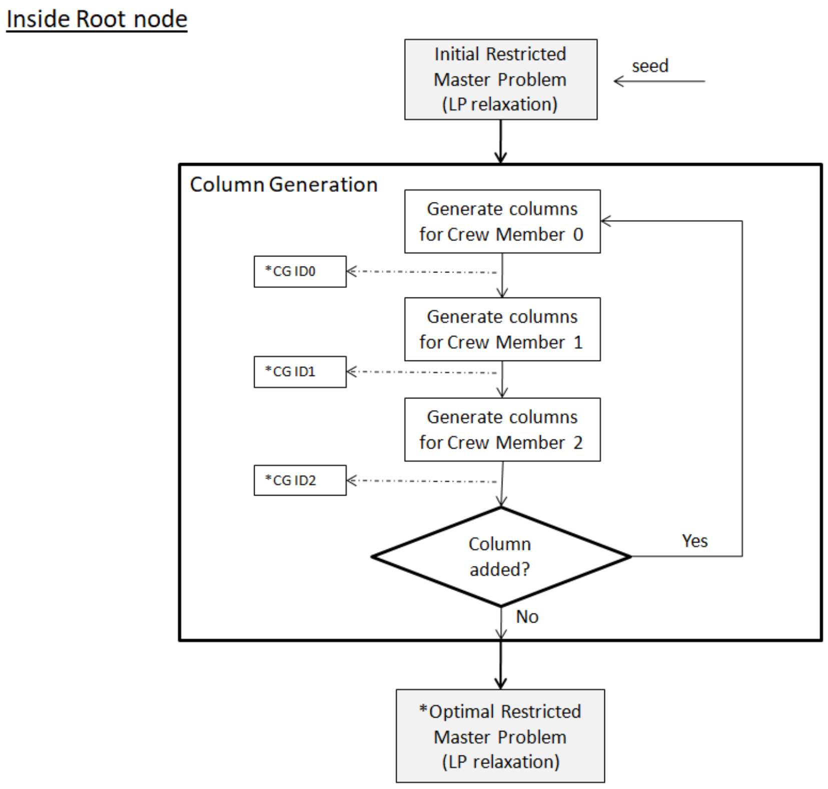
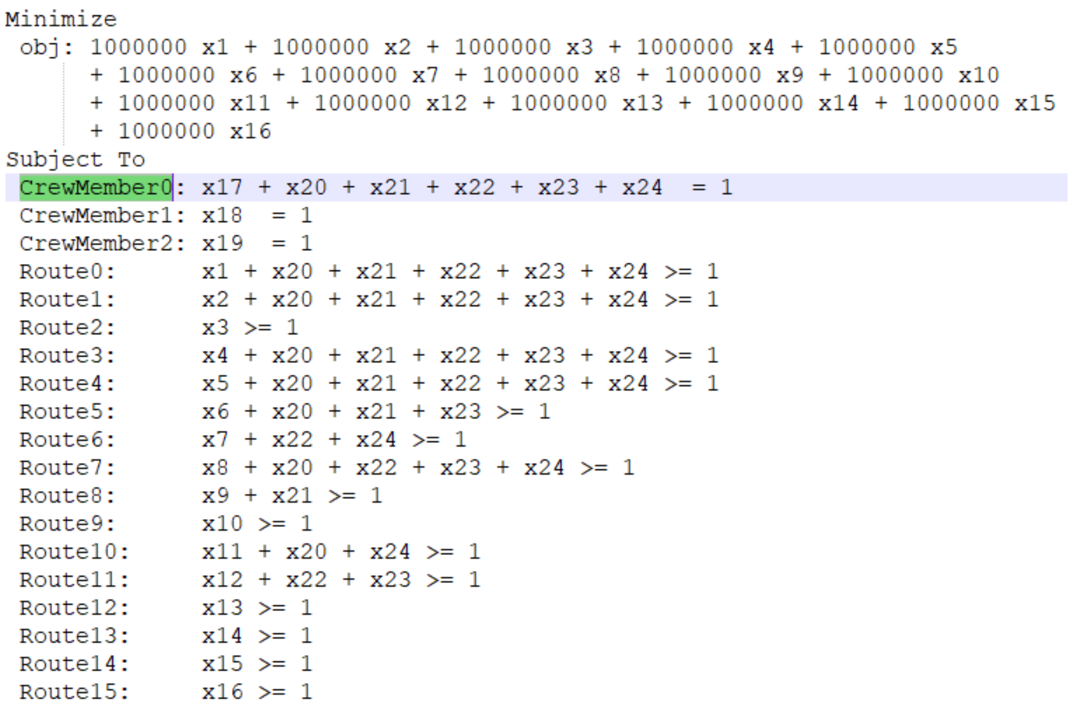
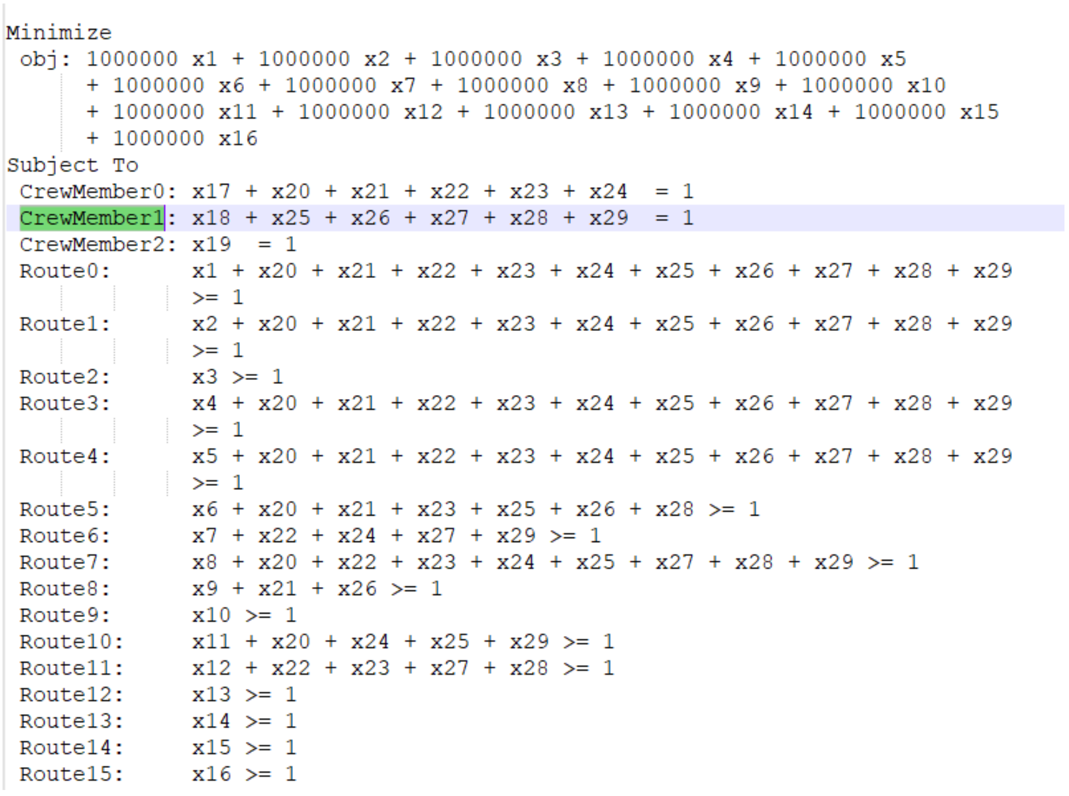

Inside Root node
Initial Restricted Master Problem (LP relaxation)
seed
Column Generation
Generate columns for Crew Member 0
*CG ID0
Generate columns for Crew Member 1
*CG ID1
Generate columns for Crew Member 2
*CG ID2
Column added?
Yes
No
*Optimal Restricted Master Problem (LP relaxation)

Figure 4.11: Column generation in Root node



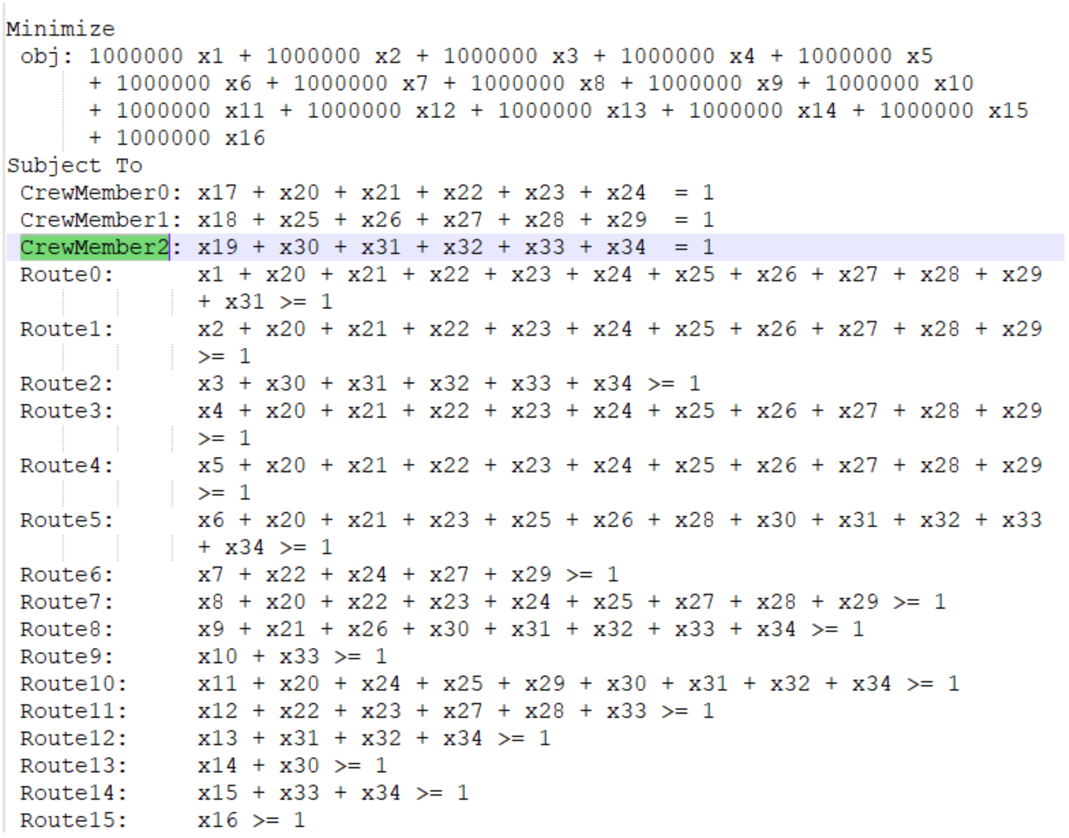
```
Minimize
 obj: 1000000 x1 + 1000000 x2 + 1000000 x3 + 1000000 x4 + 1000000 x5
      + 1000000 x6 + 1000000 x7 + 1000000 x8 + 1000000 x9 + 1000000 x10
      + 1000000 x11 + 1000000 x12 + 1000000 x13 + 1000000 x14 + 1000000 x15
      + 1000000 x16
Subject To
 CrewMember0: x17 + x20 + x21 + x22 + x23 + x24  = 1
 CrewMember1: x18  = 1
 CrewMember2: x19  = 1
 Route0:      x1 + x20 + x21 + x22 + x23 + x24 >= 1
 Route1:      x2 + x20 + x21 + x22 + x23 + x24 >= 1
 Route2:      x3 >= 1
 Route3:      x4 + x20 + x21 + x22 + x23 + x24 >= 1
 Route4:      x5 + x20 + x21 + x22 + x23 + x24 >= 1
 Route5:      x6 + x20 + x21 + x23 >= 1
 Route6:      x7 + x22 + x24 >= 1
 Route7:      x8 + x20 + x22 + x23 + x24 >= 1
 Route8:      x9 + x21 >= 1
 Route9:      x10 >= 1
 Route10:     x11 + x20 + x24 >= 1
 Route11:     x12 + x22 + x23 >= 1
 Route12:     x13 >= 1
 Route13:     x14 >= 1
 Route14:     x15 >= 1
 Route15:     x16 >= 1
```

Figure 4.12: *CG ID0



```
Minimize
 obj: 1000000 x1 + 1000000 x2 + 1000000 x3 + 1000000 x4 + 1000000 x5
      + 1000000 x6 + 1000000 x7 + 1000000 x8 + 1000000 x9 + 1000000 x10
      + 1000000 x11 + 1000000 x12 + 1000000 x13 + 1000000 x14 + 1000000 x15
      + 1000000 x16
Subject To
 CrewMember0: x17 + x20 + x21 + x22 + x23 + x24  = 1
 CrewMember1: x18 + x25 + x26 + x27 + x28 + x29  = 1
 CrewMember2: x19  = 1
 Route0:      x1 + x20 + x21 + x22 + x23 + x24 + x25 + x26 + x27 + x28 + x29
              >= 1
 Route1:      x2 + x20 + x21 + x22 + x23 + x24 + x25 + x26 + x27 + x28 + x29
              >= 1
 Route2:      x3 >= 1
 Route3:      x4 + x20 + x21 + x22 + x23 + x24 + x25 + x26 + x27 + x28 + x29
              >= 1
 Route4:      x5 + x20 + x21 + x22 + x23 + x24 + x25 + x26 + x27 + x28 + x29
              >= 1
 Route5:      x6 + x20 + x21 + x23 + x25 + x26 + x28 >= 1
 Route6:      x7 + x22 + x24 + x27 + x29 >= 1
 Route7:      x8 + x20 + x22 + x23 + x24 + x25 + x27 + x28 + x29 >= 1
 Route8:      x9 + x21 + x26 >= 1
 Route9:      x10 >= 1
 Route10:     x11 + x20 + x24 + x25 + x29 >= 1
 Route11:     x12 + x22 + x23 + x27 + x28 >= 1
 Route12:     x13 >= 1
 Route13:     x14 >= 1
 Route14:     x15 >= 1
 Route15:     x16 >= 1
```

Figure 4.13: *CG ID1

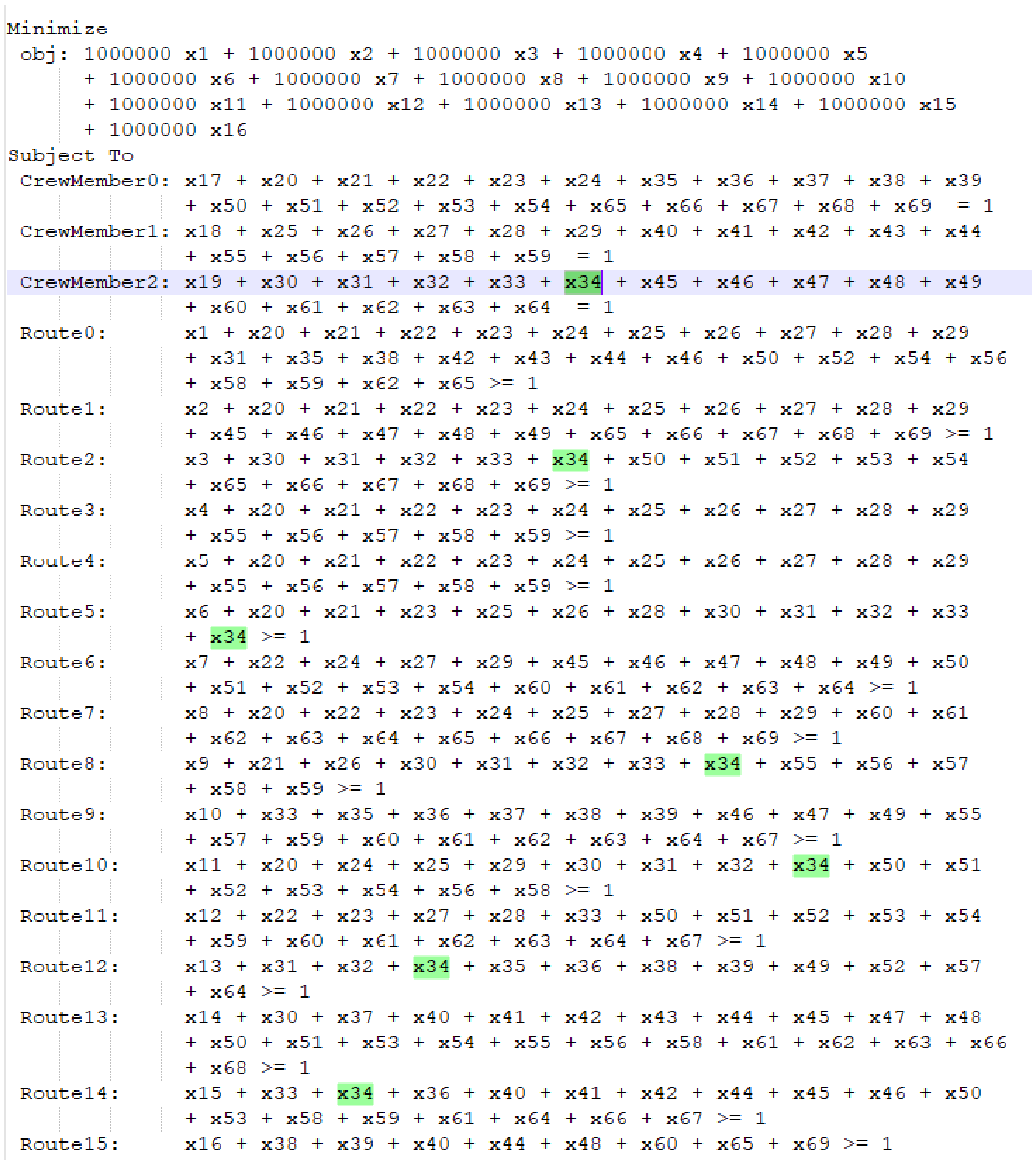


```
Minimize
 obj: 1000000 x1 + 1000000 x2 + 1000000 x3 + 1000000 x4 + 1000000 x5
      + 1000000 x6 + 1000000 x7 + 1000000 x8 + 1000000 x9 + 1000000 x10
      + 1000000 x11 + 1000000 x12 + 1000000 x13 + 1000000 x14 + 1000000 x15
      + 1000000 x16
Subject To
 CrewMember0: x17 + x20 + x21 + x22 + x23 + x24  = 1
 CrewMember1: x18 + x25 + x26 + x27 + x28 + x29  = 1
 CrewMember2: x19 + x30 + x31 + x32 + x33 + x34  = 1
 Route0:      x1 + x20 + x21 + x22 + x23 + x24 + x25 + x26 + x27 + x28 + x29
              + x31 >= 1
 Route1:      x2 + x20 + x21 + x22 + x23 + x24 + x25 + x26 + x27 + x28 + x29
              >= 1
 Route2:      x3 + x30 + x31 + x32 + x33 + x34 >= 1
 Route3:      x4 + x20 + x21 + x22 + x23 + x24 + x25 + x26 + x27 + x28 + x29
              >= 1
 Route4:      x5 + x20 + x21 + x22 + x23 + x24 + x25 + x26 + x27 + x28 + x29
              >= 1
 Route5:      x6 + x20 + x21 + x23 + x25 + x26 + x28 + x30 + x31 + x32 + x33
              + x34 >= 1
 Route6:      x7 + x22 + x24 + x27 + x29 >= 1
 Route7:      x8 + x20 + x22 + x23 + x24 + x25 + x27 + x28 + x29 >= 1
 Route8:      x9 + x21 + x26 + x30 + x31 + x32 + x33 + x34 >= 1
 Route9:      x10 + x33 >= 1
 Route10:     x11 + x20 + x24 + x25 + x29 + x30 + x31 + x32 + x34 >= 1
 Route11:     x12 + x22 + x23 + x27 + x28 + x33 >= 1
 Route12:     x13 + x31 + x32 + x34 >= 1
 Route13:     x14 + x30 >= 1
 Route14:     x15 + x33 + x34 >= 1
 Route15:     x16 >= 1
```

Figure 4.14: *CG ID2

As the reader can notice, at each step rosters are generated for a single crew member. Another thing to notice is the absence of roster-variables from the objective function. Remember that in the set-cover formulation we intentionally ignore roster quality. We would also like to highlight that 5 rosters were created for each crew member. This number is user defined and it is the number of labels of the fictitious node. We choose to insert to the master problem the best 5 paths that were identified for each crew member.

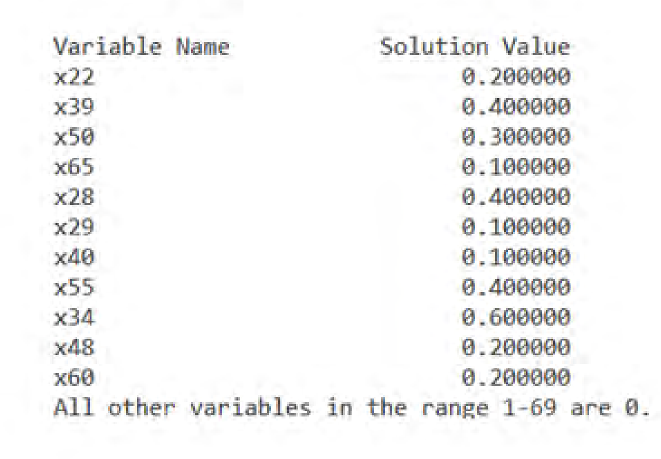
The column generation procedure continues until no column with negative reduced cost can be identified for all crew member. For our case, the product of the column generation procedure in the root node of the search tree can be seen in Figure 4.15.



```
Minimize
 obj: 1000000 x1 + 1000000 x2 + 1000000 x3 + 1000000 x4 + 1000000 x5
      + 1000000 x6 + 1000000 x7 + 1000000 x8 + 1000000 x9 + 1000000 x10
      + 1000000 x11 + 1000000 x12 + 1000000 x13 + 1000000 x14 + 1000000 x15
      + 1000000 x16
Subject To
 CrewMember0: x17 + x20 + x21 + x22 + x23 + x24 + x35 + x36 + x37 + x38 + x39
              + x50 + x51 + x52 + x53 + x54 + x65 + x66 + x67 + x68 + x69  = 1
 CrewMember1: x18 + x25 + x26 + x27 + x28 + x29 + x40 + x41 + x42 + x43 + x44
              + x55 + x56 + x57 + x58 + x59  = 1
 CrewMember2: x19 + x30 + x31 + x32 + x33 + x34 + x45 + x46 + x47 + x48 + x49
              + x60 + x61 + x62 + x63 + x64  = 1
 Route0:      x1 + x20 + x21 + x22 + x23 + x24 + x25 + x26 + x27 + x28 + x29
              + x31 + x35 + x38 + x42 + x43 + x44 + x46 + x50 + x52 + x54 + x56
              + x58 + x59 + x62 + x65 >= 1
 Route1:      x2 + x20 + x21 + x22 + x23 + x24 + x25 + x26 + x27 + x28 + x29
              + x45 + x46 + x47 + x48 + x49 + x65 + x66 + x67 + x68 + x69 >= 1
 Route2:      x3 + x30 + x31 + x32 + x33 + x34 + x50 + x51 + x52 + x53 + x54
              + x65 + x66 + x67 + x68 + x69 >= 1
 Route3:      x4 + x20 + x21 + x22 + x23 + x24 + x25 + x26 + x27 + x28 + x29
              + x55 + x56 + x57 + x58 + x59 >= 1
 Route4:      x5 + x20 + x21 + x22 + x23 + x24 + x25 + x26 + x27 + x28 + x29
              + x55 + x56 + x57 + x58 + x59 >= 1
 Route5:      x6 + x20 + x21 + x23 + x25 + x26 + x28 + x30 + x31 + x32 + x33
              + x34 >= 1
 Route6:      x7 + x22 + x24 + x27 + x29 + x45 + x46 + x47 + x48 + x49 + x50
              + x51 + x52 + x53 + x54 + x60 + x61 + x62 + x63 + x64 >= 1
 Route7:      x8 + x20 + x22 + x23 + x24 + x25 + x27 + x28 + x29 + x60 + x61
              + x62 + x63 + x64 + x65 + x66 + x67 + x68 + x69 >= 1
 Route8:      x9 + x21 + x26 + x30 + x31 + x32 + x33 + x34 + x55 + x56 + x57
              + x58 + x59 >= 1
 Route9:      x10 + x33 + x35 + x36 + x37 + x38 + x39 + x46 + x47 + x49 + x55
              + x57 + x59 + x60 + x61 + x62 + x63 + x64 + x67 >= 1
 Route10:     x11 + x20 + x24 + x25 + x29 + x30 + x31 + x32 + x34 + x50 + x51
              + x52 + x53 + x54 + x56 + x58 >= 1
 Route11:     x12 + x22 + x23 + x27 + x28 + x33 + x50 + x51 + x52 + x53 + x54
              + x59 + x60 + x61 + x62 + x63 + x64 + x67 >= 1
 Route12:     x13 + x31 + x32 + x34 + x35 + x36 + x38 + x39 + x49 + x52 + x57
              + x64 >= 1
 Route13:     x14 + x30 + x37 + x40 + x41 + x42 + x43 + x44 + x45 + x47 + x48
              + x50 + x51 + x53 + x54 + x55 + x56 + x58 + x61 + x62 + x63 + x66
              + x68 >= 1
 Route14:     x15 + x33 + x34 + x36 + x40 + x41 + x42 + x44 + x45 + x46 + x50
              + x53 + x58 + x59 + x61 + x64 + x66 + x67 >= 1
 Route15:     x16 + x38 + x39 + x40 + x44 + x48 + x60 + x65 + x69 >= 1
```

Figure 4.15: *Optimal Restricted Master Problem (LP relaxation) at Root node

At this point, we have produced a master problem formulation, the solution of which is the optimal solution of the lp relaxation of our problem (see Figure 4.16).



```
Variable Name          Solution Value
x22                          0.200000
x39                          0.400000
x50                          0.300000
x65                          0.100000
x28                          0.400000
x29                          0.100000
x40                          0.100000
x55                          0.400000
x34                          0.600000
x48                          0.200000
x60                          0.200000
All other variables in the range 1-69 are 0.
```

Figure 4.16: Optimal solution of the lp relaxation in the root node

Apparently, the solution is not integer. Therefore, we need to start the branching phase in order to obtain an integer solution. As we stated previously, we choose to branch on a single variable, the one with the largest fractional value in the optimal solution of the lp relaxation. In our case this roster-variable is *x34*, which is equal to 0.6. By reviewing Figure 4.15 (*x34* is highlighted), one can notice that this roster belongs to Crew Member 2 and includes pairings 2, 5, 8, 10, 12, 14. As a result, we branch on variable *x34* (see Figure 4.17).

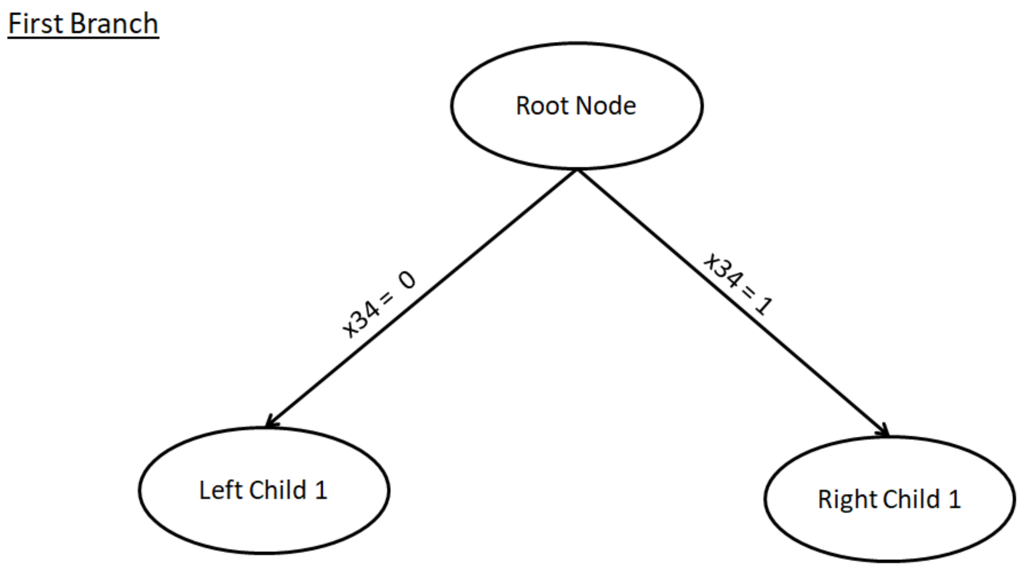
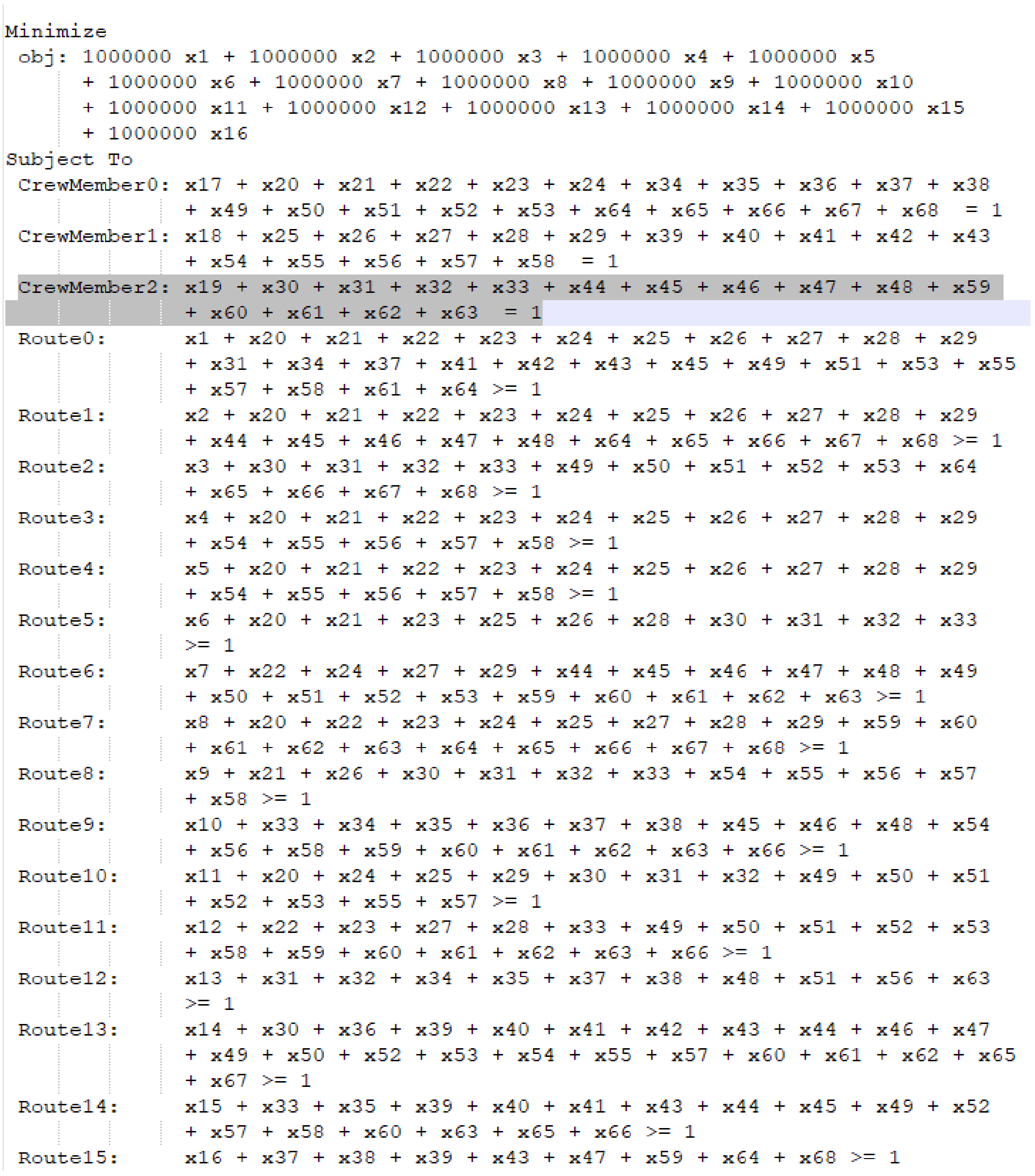


Figure 4.17: Branching on variable *x34*

We now present the sub-problems of each child after the branching. In Figure 4.18, the left sub-problem is presented. The variable *x34* which belonged to Crew Member 2 has been deleted from this problem. If the reader pays more attention he/she will notice that in this formulation a *x34* variable continues to exist after the deletion and this one is a roster-variable of Crew Member 0. The reason behind this is that when we erase a variable in a CPLEX optimization problem, the indices of all variables with an index larger of that erased, are decreased by 1. That means that variable *x34* in the left sub-problem formulation is actually variable *x35* of the parent formulation. If the reader examines more carefully the formulation, it will become apparent that all variables x_i with i>34 in the parent formulation, are represented as x_{i-1} in the left sub-problem.

Apart from this modification in the master problem formulation, modifications are also done in the column generation sub-problem of the left child. The network is augmented as shown in Figure 4.6 and some auxiliary, side-constraints are added to prohibit the generation of the same path in the subsequent tree nodes.



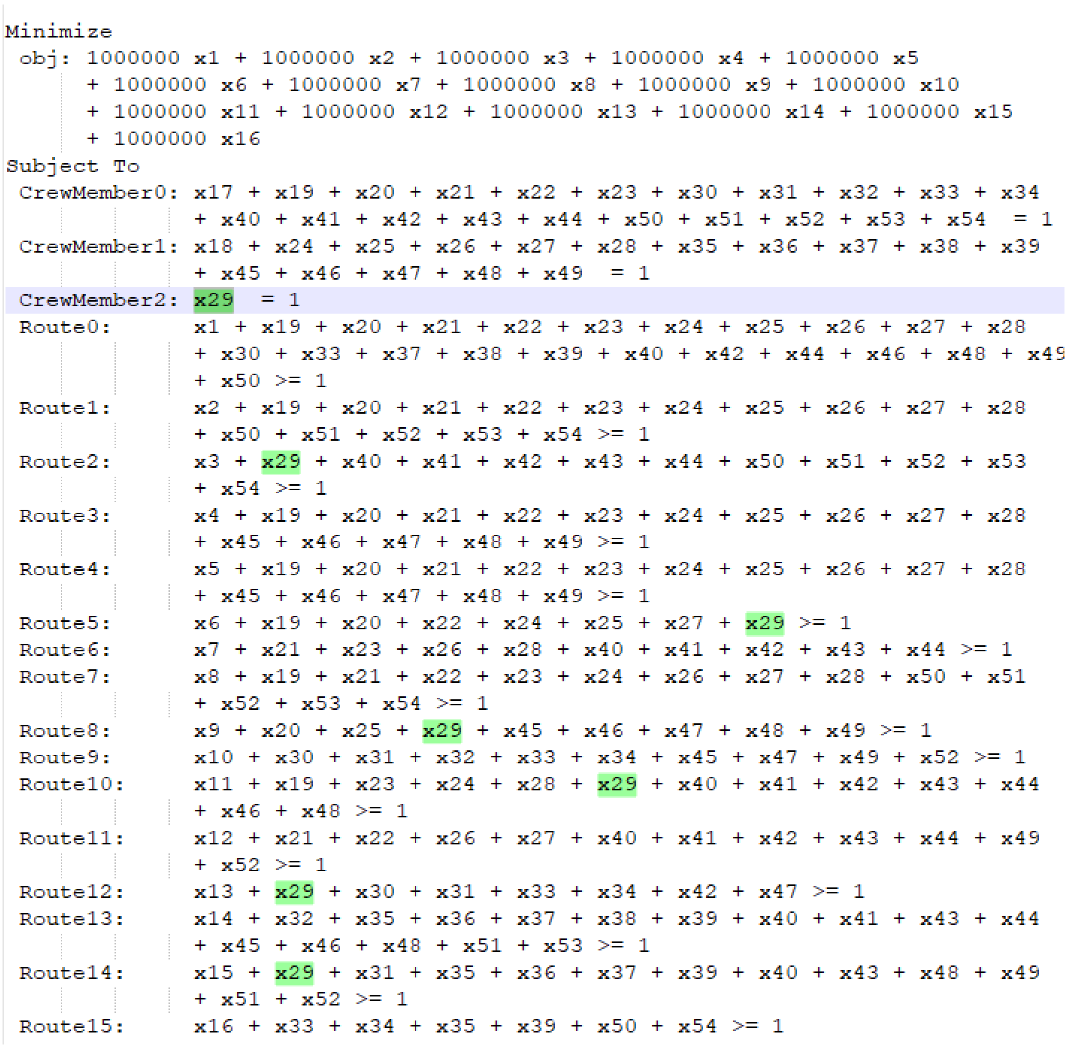
```
Minimize
 obj: 1000000 x1 + 1000000 x2 + 1000000 x3 + 1000000 x4 + 1000000 x5
      + 1000000 x6 + 1000000 x7 + 1000000 x8 + 1000000 x9 + 1000000 x10
      + 1000000 x11 + 1000000 x12 + 1000000 x13 + 1000000 x14 + 1000000 x15
      + 1000000 x16
Subject To
 CrewMember0: x17 + x20 + x21 + x22 + x23 + x24 + x34 + x35 + x36 + x37 + x38
              + x49 + x50 + x51 + x52 + x53 + x64 + x65 + x66 + x67 + x68  = 1
 CrewMember1: x18 + x25 + x26 + x27 + x28 + x29 + x39 + x40 + x41 + x42 + x43
              + x54 + x55 + x56 + x57 + x58  = 1
 CrewMember2: x19 + x30 + x31 + x32 + x33 + x44 + x45 + x46 + x47 + x48 + x59
              + x60 + x61 + x62 + x63  = 1
 Route0:      x1 + x20 + x21 + x22 + x23 + x24 + x25 + x26 + x27 + x28 + x29
              + x31 + x34 + x37 + x41 + x42 + x43 + x45 + x49 + x51 + x53 + x55
              + x57 + x58 + x61 + x64 >= 1
 Route1:      x2 + x20 + x21 + x22 + x23 + x24 + x25 + x26 + x27 + x28 + x29
              + x44 + x45 + x46 + x47 + x48 + x64 + x65 + x66 + x67 + x68 >= 1
 Route2:      x3 + x30 + x31 + x32 + x33 + x49 + x50 + x51 + x52 + x53 + x64
              + x65 + x66 + x67 + x68 >= 1
 Route3:      x4 + x20 + x21 + x22 + x23 + x24 + x25 + x26 + x27 + x28 + x29
              + x54 + x55 + x56 + x57 + x58 >= 1
 Route4:      x5 + x20 + x21 + x22 + x23 + x24 + x25 + x26 + x27 + x28 + x29
              + x54 + x55 + x56 + x57 + x58 >= 1
 Route5:      x6 + x20 + x21 + x23 + x25 + x26 + x28 + x30 + x31 + x32 + x33
              >= 1
 Route6:      x7 + x22 + x24 + x27 + x29 + x44 + x45 + x46 + x47 + x48 + x49
              + x50 + x51 + x52 + x53 + x59 + x60 + x61 + x62 + x63 >= 1
 Route7:      x8 + x20 + x22 + x23 + x24 + x25 + x27 + x28 + x29 + x59 + x60
              + x61 + x62 + x63 + x64 + x65 + x66 + x67 + x68 >= 1
 Route8:      x9 + x21 + x26 + x30 + x31 + x32 + x33 + x54 + x55 + x56 + x57
              + x58 >= 1
 Route9:      x10 + x33 + x34 + x35 + x36 + x37 + x38 + x45 + x46 + x48 + x54
              + x56 + x58 + x59 + x60 + x61 + x62 + x63 + x66 >= 1
 Route10:     x11 + x20 + x24 + x25 + x29 + x30 + x31 + x32 + x49 + x50 + x51
              + x52 + x53 + x55 + x57 >= 1
 Route11:     x12 + x22 + x23 + x27 + x28 + x33 + x49 + x50 + x51 + x52 + x53
              + x58 + x59 + x60 + x61 + x62 + x63 + x66 >= 1
 Route12:     x13 + x31 + x32 + x34 + x35 + x37 + x38 + x48 + x51 + x56 + x63
              >= 1
 Route13:     x14 + x30 + x36 + x39 + x40 + x41 + x42 + x43 + x44 + x46 + x47
              + x49 + x50 + x52 + x53 + x54 + x55 + x57 + x60 + x61 + x62 + x65
              + x67 >= 1
 Route14:     x15 + x33 + x35 + x39 + x40 + x41 + x43 + x44 + x45 + x49 + x52
              + x57 + x58 + x60 + x63 + x65 + x66 >= 1
 Route15:     x16 + x37 + x38 + x39 + x43 + x47 + x59 + x64 + x68 >= 1
```

Figure 4.18: Left sub-problem

In Figure 4.19, we present the right sub-problem. In this, we have "fixed" the value of *x34* to 1 and we erased all the other roster-variables of Crew Member 2. Once again, when we delete a variable, all variables with a greater index than the deleted variable have their index decreased by 1. Since we erased 5 variables with a smaller index than *x34*, this variable is represented as *x29* in the right sub-problem, while it still has the same meaning; it is a roster of Crew Member 2, covering pairings 2, 5, 8, 10, 12, 14, as is highlighted in Figure 4.19. The only thing that changes is its name.

It is important to note at this point that if we utilized the set-partition formulation we would also delete all rosters covering pairings 2, 5, 8, 10, 12, 14, even those belonging to crew members other than Crew Member 2. However, this is not needed in the set-cover formulation, where over-coverage of pairings is allowed.

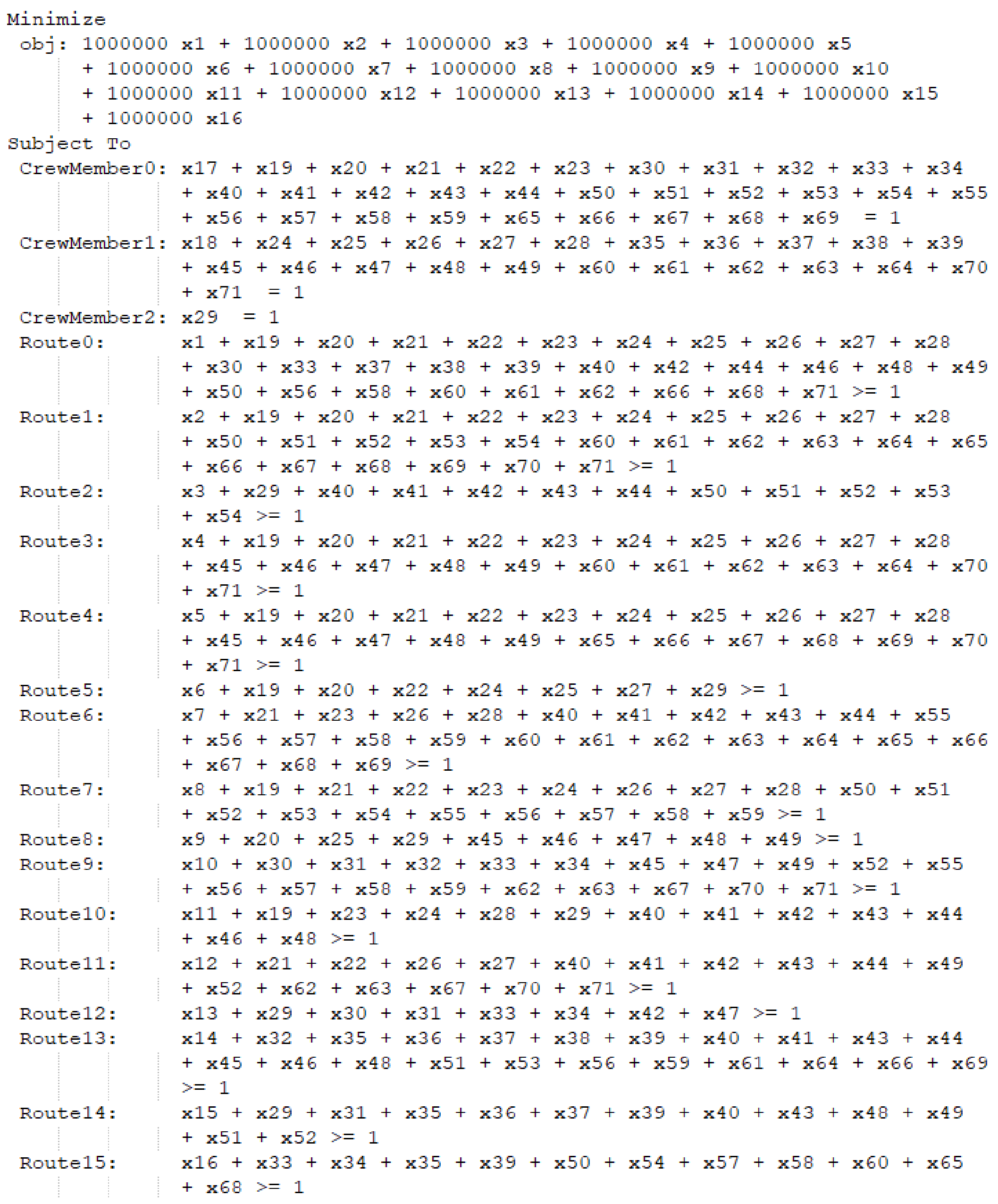
The changes that take place in the network of the right child is that the Crew Member 2 node and Pairing 2, 5, 8, 10, 12, 14 nodes are essentially removed.



```
Minimize
 obj: 1000000 x1 + 1000000 x2 + 1000000 x3 + 1000000 x4 + 1000000 x5
      + 1000000 x6 + 1000000 x7 + 1000000 x8 + 1000000 x9 + 1000000 x10
      + 1000000 x11 + 1000000 x12 + 1000000 x13 + 1000000 x14 + 1000000 x15
      + 1000000 x16
Subject To
 CrewMember0: x17 + x19 + x20 + x21 + x22 + x23 + x30 + x31 + x32 + x33 + x34
              + x40 + x41 + x42 + x43 + x44 + x50 + x51 + x52 + x53 + x54  = 1
 CrewMember1: x18 + x24 + x25 + x26 + x27 + x28 + x35 + x36 + x37 + x38 + x39
              + x45 + x46 + x47 + x48 + x49  = 1
 CrewMember2: x29  = 1
 Route0:      x1 + x19 + x20 + x21 + x22 + x23 + x24 + x25 + x26 + x27 + x28
              + x30 + x33 + x37 + x38 + x39 + x40 + x42 + x44 + x46 + x48 + x49
              + x50 >= 1
 Route1:      x2 + x19 + x20 + x21 + x22 + x23 + x24 + x25 + x26 + x27 + x28
              + x50 + x51 + x52 + x53 + x54 >= 1
 Route2:      x3 + x29 + x40 + x41 + x42 + x43 + x44 + x50 + x51 + x52 + x53
              + x54 >= 1
 Route3:      x4 + x19 + x20 + x21 + x22 + x23 + x24 + x25 + x26 + x27 + x28
              + x45 + x46 + x47 + x48 + x49 >= 1
 Route4:      x5 + x19 + x20 + x21 + x22 + x23 + x24 + x25 + x26 + x27 + x28
              + x45 + x46 + x47 + x48 + x49 >= 1
 Route5:      x6 + x19 + x20 + x22 + x24 + x25 + x27 + x29 >= 1
 Route6:      x7 + x21 + x23 + x26 + x28 + x40 + x41 + x42 + x43 + x44 >= 1
 Route7:      x8 + x19 + x21 + x22 + x23 + x24 + x26 + x27 + x28 + x50 + x51
              + x52 + x53 + x54 >= 1
 Route8:      x9 + x20 + x25 + x29 + x45 + x46 + x47 + x48 + x49 >= 1
 Route9:      x10 + x30 + x31 + x32 + x33 + x34 + x45 + x47 + x49 + x52 >= 1
 Route10:     x11 + x19 + x23 + x24 + x28 + x29 + x40 + x41 + x42 + x43 + x44
              + x46 + x48 >= 1
 Route11:     x12 + x21 + x22 + x26 + x27 + x40 + x41 + x42 + x43 + x44 + x49
              + x52 >= 1
 Route12:     x13 + x29 + x30 + x31 + x33 + x34 + x42 + x47 >= 1
 Route13:     x14 + x32 + x35 + x36 + x37 + x38 + x39 + x40 + x41 + x43 + x44
              + x45 + x46 + x48 + x51 + x53 >= 1
 Route14:     x15 + x29 + x31 + x35 + x36 + x37 + x39 + x40 + x43 + x48 + x49
              + x51 + x52 >= 1
 Route15:     x16 + x33 + x34 + x35 + x39 + x50 + x54 >= 1
```

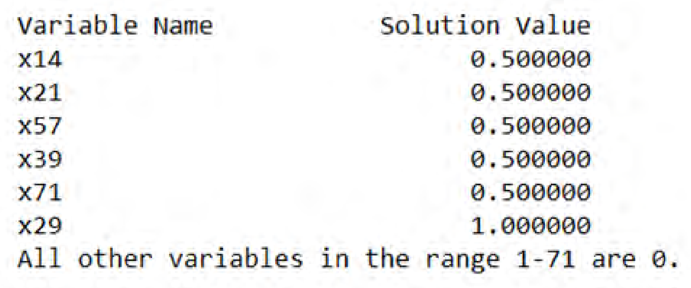
Figure 4.19: Right sub-problem

Our strategy to follow the right flow dictates that we continue to the right child. Therefore, we generate columns for the right sub-problem. This time however, we do not generate rosters for Crew Member 2, since this member has already been assigned a roster. In Figure 4.20 the reader can see the master problem formulation, after the column generation procedure that took place in Right Child 1 node. As one can notice, rosters were generated only for crew members 0 and 1 as expected.



```
Minimize
 obj: 1000000 x1 + 1000000 x2 + 1000000 x3 + 1000000 x4 + 1000000 x5
      + 1000000 x6 + 1000000 x7 + 1000000 x8 + 1000000 x9 + 1000000 x10
      + 1000000 x11 + 1000000 x12 + 1000000 x13 + 1000000 x14 + 1000000 x15
      + 1000000 x16
Subject To
 CrewMember0: x17 + x19 + x20 + x21 + x22 + x23 + x30 + x31 + x32 + x33 + x34
              + x40 + x41 + x42 + x43 + x44 + x50 + x51 + x52 + x53 + x54 + x55
              + x56 + x57 + x58 + x59 + x65 + x66 + x67 + x68 + x69  = 1
 CrewMember1: x18 + x24 + x25 + x26 + x27 + x28 + x35 + x36 + x37 + x38 + x39
              + x45 + x46 + x47 + x48 + x49 + x60 + x61 + x62 + x63 + x64 + x70
              + x71  = 1
 CrewMember2: x29  = 1
 Route0:      x1 + x19 + x20 + x21 + x22 + x23 + x24 + x25 + x26 + x27 + x28
              + x30 + x33 + x37 + x38 + x39 + x40 + x42 + x44 + x46 + x48 + x49
              + x50 + x56 + x58 + x60 + x61 + x62 + x66 + x68 + x71 >= 1
 Route1:      x2 + x19 + x20 + x21 + x22 + x23 + x24 + x25 + x26 + x27 + x28
              + x50 + x51 + x52 + x53 + x54 + x60 + x61 + x62 + x63 + x64 + x65
              + x66 + x67 + x68 + x69 + x70 + x71 >= 1
 Route2:      x3 + x29 + x40 + x41 + x42 + x43 + x44 + x50 + x51 + x52 + x53
              + x54 >= 1
 Route3:      x4 + x19 + x20 + x21 + x22 + x23 + x24 + x25 + x26 + x27 + x28
              + x45 + x46 + x47 + x48 + x49 + x60 + x61 + x62 + x63 + x64 + x70
              + x71 >= 1
 Route4:      x5 + x19 + x20 + x21 + x22 + x23 + x24 + x25 + x26 + x27 + x28
              + x45 + x46 + x47 + x48 + x49 + x65 + x66 + x67 + x68 + x69 + x70
              + x71 >= 1
 Route5:      x6 + x19 + x20 + x22 + x24 + x25 + x27 + x29 >= 1
 Route6:      x7 + x21 + x23 + x26 + x28 + x40 + x41 + x42 + x43 + x44 + x55
              + x56 + x57 + x58 + x59 + x60 + x61 + x62 + x63 + x64 + x65 + x66
              + x67 + x68 + x69 >= 1
 Route7:      x8 + x19 + x21 + x22 + x23 + x24 + x26 + x27 + x28 + x50 + x51
              + x52 + x53 + x54 + x55 + x56 + x57 + x58 + x59 >= 1
 Route8:      x9 + x20 + x25 + x29 + x45 + x46 + x47 + x48 + x49 >= 1
 Route9:      x10 + x30 + x31 + x32 + x33 + x34 + x45 + x47 + x49 + x52 + x55
              + x56 + x57 + x58 + x59 + x62 + x63 + x67 + x70 + x71 >= 1
 Route10:     x11 + x19 + x23 + x24 + x28 + x29 + x40 + x41 + x42 + x43 + x44
              + x46 + x48 >= 1
 Route11:     x12 + x21 + x22 + x26 + x27 + x40 + x41 + x42 + x43 + x44 + x49
              + x52 + x62 + x63 + x67 + x70 + x71 >= 1
 Route12:     x13 + x29 + x30 + x31 + x33 + x34 + x42 + x47 >= 1
 Route13:     x14 + x32 + x35 + x36 + x37 + x38 + x39 + x40 + x41 + x43 + x44
              + x45 + x46 + x48 + x51 + x53 + x56 + x59 + x61 + x64 + x66 + x69
              >= 1
 Route14:     x15 + x29 + x31 + x35 + x36 + x37 + x39 + x40 + x43 + x48 + x49
              + x51 + x52 >= 1
 Route15:     x16 + x33 + x34 + x35 + x39 + x50 + x54 + x57 + x58 + x60 + x65
              + x68 >= 1
```

Figure 4.20: *Optimal Restricted Master Problem (LP relaxation) at Right Child 1

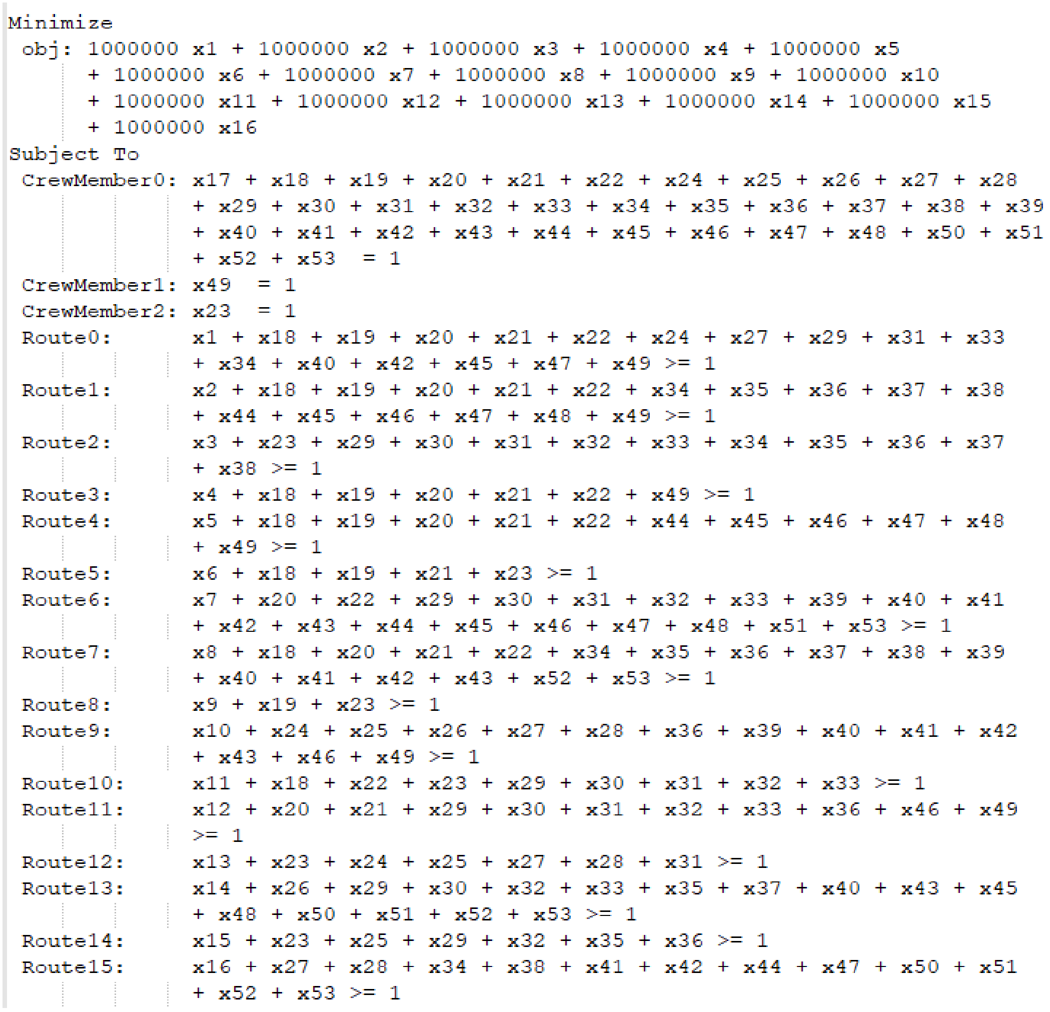


```
Variable Name              Solution Value
x14                              0.500000
x21                              0.500000
x57                              0.500000
x39                              0.500000
x71                              0.500000
x29                              1.000000
All other variables in the range 1-71 are 0.
```

Figure 4.21: Optimal solution of the lp relaxation in the Right Child 1

The solution of this problem is presented in Figure 4.21. Since more than one roster-variables share the largest fractional value in the optimal solution we chose to branch on one arbitrarily. In our case we choose to branch on *x71*, which is a roster variable of Crew Member 1.

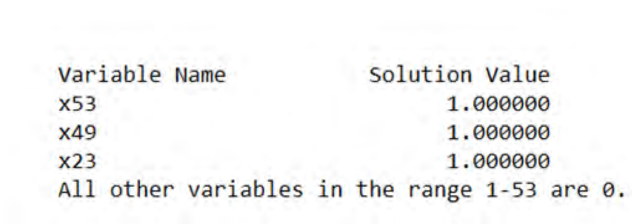
In a similar way as presented previously, we branch on this variable and create two children-nodes. Once again, we continue to the right node, where we generate columns; but this time, just for Crew Member 0. The master problem formulation after the column generation procedure of Right Child 2 node is presented in Figure 4.22.



```
Minimize
 obj: 1000000 x1 + 1000000 x2 + 1000000 x3 + 1000000 x4 + 1000000 x5
      + 1000000 x6 + 1000000 x7 + 1000000 x8 + 1000000 x9 + 1000000 x10
      + 1000000 x11 + 1000000 x12 + 1000000 x13 + 1000000 x14 + 1000000 x15
      + 1000000 x16
Subject To
 CrewMember0: x17 + x18 + x19 + x20 + x21 + x22 + x24 + x25 + x26 + x27 + x28
              + x29 + x30 + x31 + x32 + x33 + x34 + x35 + x36 + x37 + x38 + x39
              + x40 + x41 + x42 + x43 + x44 + x45 + x46 + x47 + x48 + x50 + x51
              + x52 + x53  = 1
 CrewMember1: x49  = 1
 CrewMember2: x23  = 1
 Route0:      x1 + x18 + x19 + x20 + x21 + x22 + x24 + x27 + x29 + x31 + x33
              + x34 + x40 + x42 + x45 + x47 + x49 >= 1
 Route1:      x2 + x18 + x19 + x20 + x21 + x22 + x34 + x35 + x36 + x37 + x38
              + x44 + x45 + x46 + x47 + x48 + x49 >= 1
 Route2:      x3 + x23 + x29 + x30 + x31 + x32 + x33 + x34 + x35 + x36 + x37
              + x38 >= 1
 Route3:      x4 + x18 + x19 + x20 + x21 + x22 + x49 >= 1
 Route4:      x5 + x18 + x19 + x20 + x21 + x22 + x44 + x45 + x46 + x47 + x48
              + x49 >= 1
 Route5:      x6 + x18 + x19 + x21 + x23 >= 1
 Route6:      x7 + x20 + x22 + x29 + x30 + x31 + x32 + x33 + x39 + x40 + x41
              + x42 + x43 + x44 + x45 + x46 + x47 + x48 + x51 + x53 >= 1
 Route7:      x8 + x18 + x20 + x21 + x22 + x34 + x35 + x36 + x37 + x38 + x39
              + x40 + x41 + x42 + x43 + x52 + x53 >= 1
 Route8:      x9 + x19 + x23 >= 1
 Route9:      x10 + x24 + x25 + x26 + x27 + x28 + x36 + x39 + x40 + x41 + x42
              + x43 + x46 + x49 >= 1
 Route10:     x11 + x18 + x22 + x23 + x29 + x30 + x31 + x32 + x33 >= 1
 Route11:     x12 + x20 + x21 + x29 + x30 + x31 + x32 + x33 + x36 + x46 + x49
              >= 1
 Route12:     x13 + x23 + x24 + x25 + x27 + x28 + x31 >= 1
 Route13:     x14 + x26 + x29 + x30 + x32 + x33 + x35 + x37 + x40 + x43 + x45
              + x48 + x50 + x51 + x52 + x53 >= 1
 Route14:     x15 + x23 + x25 + x29 + x32 + x35 + x36 >= 1
 Route15:     x16 + x27 + x28 + x34 + x38 + x41 + x42 + x44 + x47 + x50 + x51
              + x52 + x53 >= 1
```

Figure 4.22: *Optimal Restricted Master Problem (LP relaxation) at Right Child 2

When we solve the above master problem an integer solution is obtained (see Figure 4.23). As you can see, the objective function of the optimal solution is 0, hence the MIP gap tolerance is not violated and the program does not search for another integer solution. Therefore, the set-cover phase is over. In Figure 4.24, the whole search tree, until the end of the set-cover phase, is presented.



```
Variable Name          Solution Value
x53                          1.000000
x49                          1.000000
x23                          1.000000
All other variables in the range 1-53 are 0.
```

Figure 4.23: Optimal solution of the lp relaxation in the Right Child 2

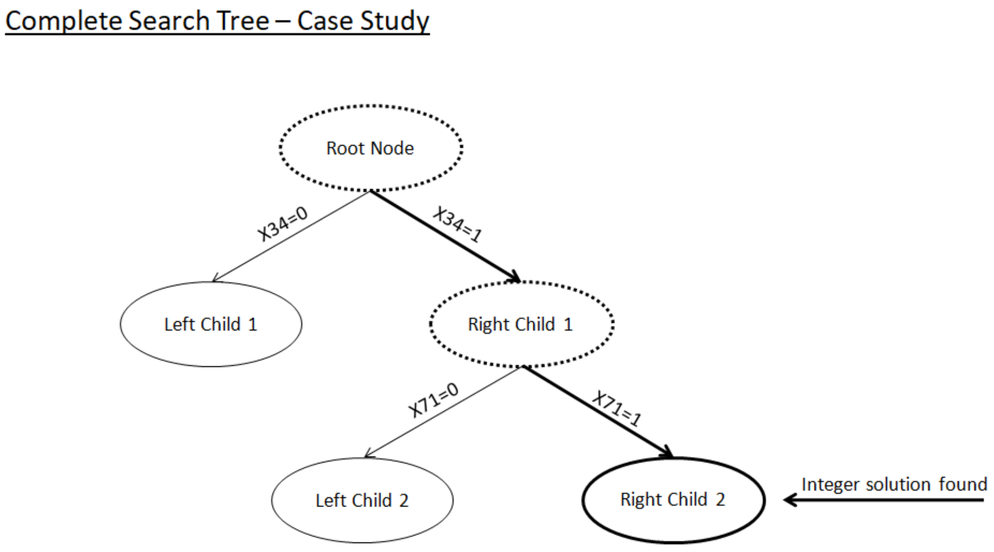
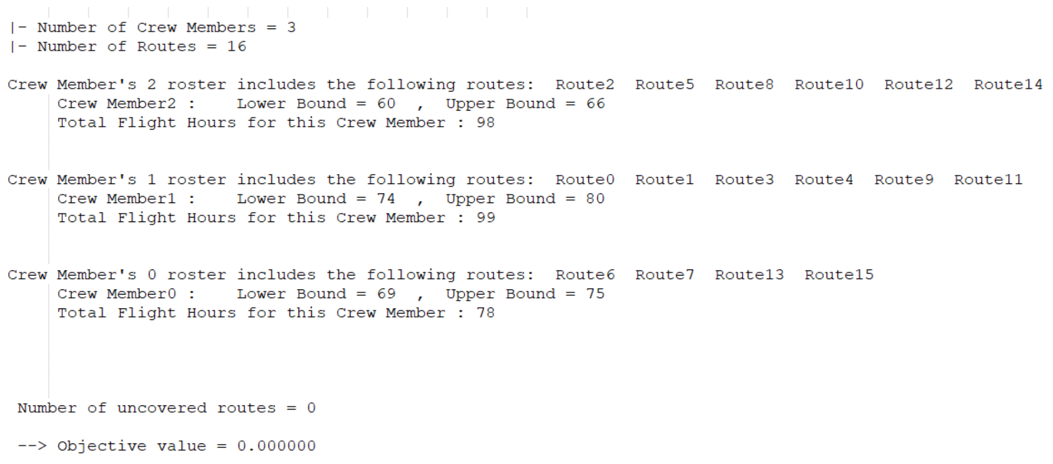


Figure 4.24: Complete search tree

Results

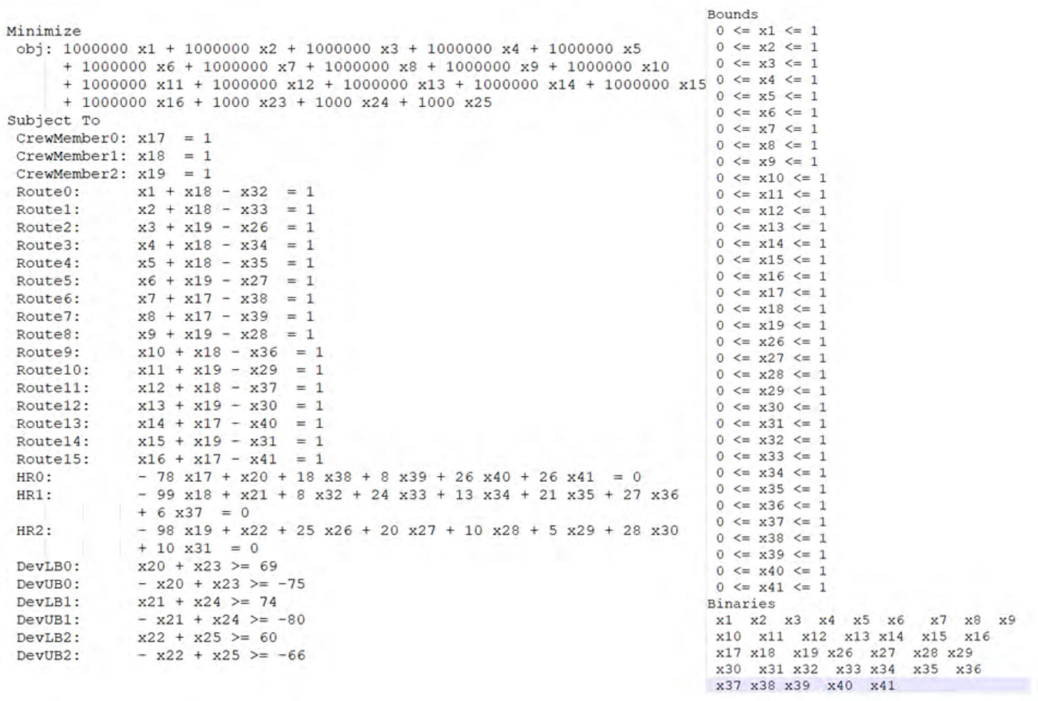
Finally, we demonstrate the solution of the set-cover phase as provided by our program in Figure 4.25. In this solution all pairings are covered and there is no over-covered pairing, since all pairings are covered exactly once, as needed.



```
|- Number of Crew Members = 3
|- Number of Routes = 16

Crew Member's 2 roster includes the following routes:  Route2  Route5  Route8  Route10  Route12  Route14
     Crew Member2 :    Lower Bound = 60  ,  Upper Bound = 66
     Total Flight Hours for this Crew Member : 98


Crew Member's 1 roster includes the following routes:  Route0  Route1  Route3  Route4  Route9  Route11
     Crew Member1 :    Lower Bound = 74  ,  Upper Bound = 80
     Total Flight Hours for this Crew Member : 99


Crew Member's 0 roster includes the following routes:  Route6  Route7  Route13  Route15
     Crew Member0 :    Lower Bound = 69  ,  Upper Bound = 75
     Total Flight Hours for this Crew Member : 78



 Number of uncovered routes = 0

 --> Objective value = 0.000000
```

Figure 4.25: Integer solution at the end of the set-cover phase

However, we still need to calculate the final cost of the schedules, since the set-cover formulation does not take into account the roster quality cost. In our small case, this can be done manually, but in larger cases that would be difficult. Therefore, even though there is no need for pairing deletion from a roster, we utilize the "converting" problem to calculate the final cost of the schedules we created (see Figure 4.26).

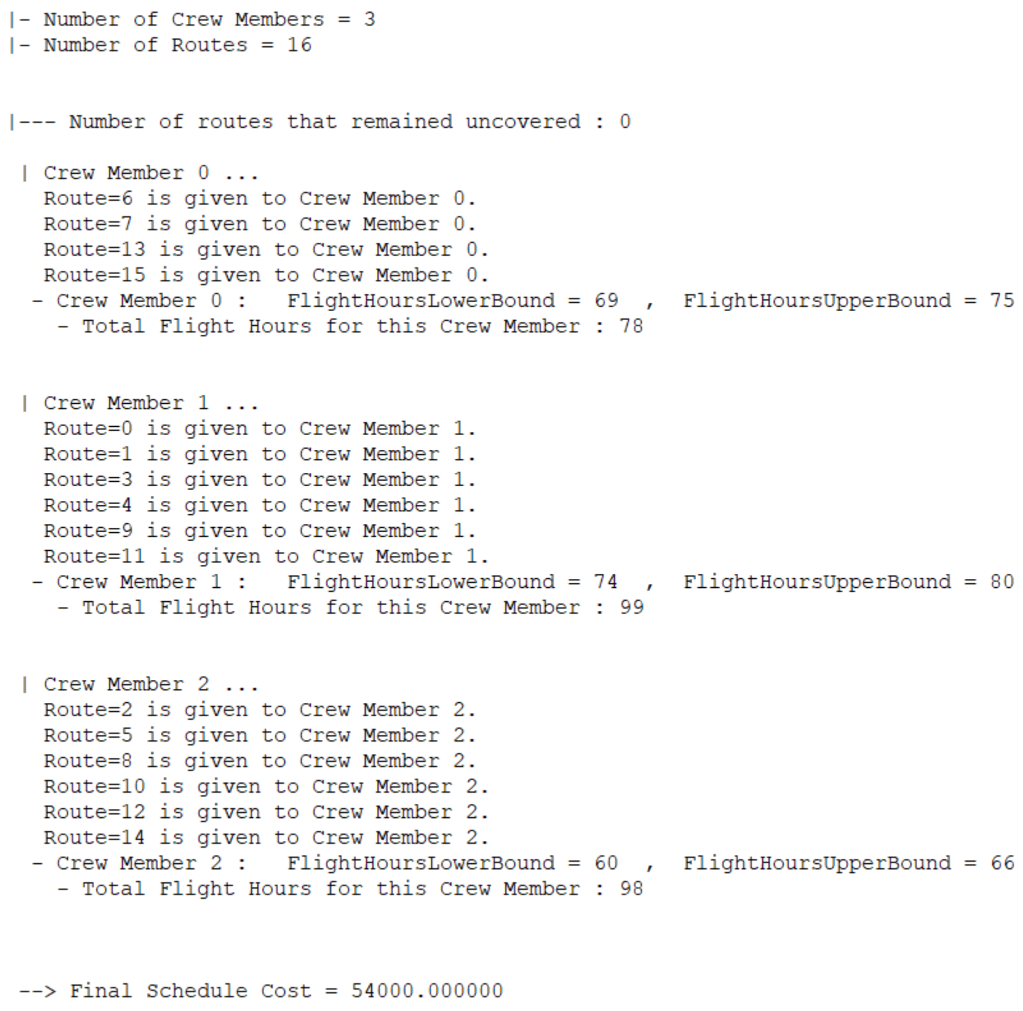


```
Minimize
 obj: 1000000 x1 + 1000000 x2 + 1000000 x3 + 1000000 x4 + 1000000 x5
      + 1000000 x6 + 1000000 x7 + 1000000 x8 + 1000000 x9 + 1000000 x10
      + 1000000 x11 + 1000000 x12 + 1000000 x13 + 1000000 x14 + 1000000 x15
      + 1000000 x16 + 1000 x23 + 1000 x24 + 1000 x25
Subject To
 CrewMember0: x17  = 1
 CrewMember1: x18  = 1
 CrewMember2: x19  = 1
 Route0:      x1 + x18 - x32  = 1
 Route1:      x2 + x18 - x33  = 1
 Route2:      x3 + x19 - x26  = 1
 Route3:      x4 + x18 - x34  = 1
 Route4:      x5 + x18 - x35  = 1
 Route5:      x6 + x19 - x27  = 1
 Route6:      x7 + x17 - x38  = 1
 Route7:      x8 + x17 - x39  = 1
 Route8:      x9 + x19 - x28  = 1
 Route9:      x10 + x18 - x36  = 1
 Route10:     x11 + x19 - x29  = 1
 Route11:     x12 + x18 - x37  = 1
 Route12:     x13 + x19 - x30  = 1
 Route13:     x14 + x17 - x40  = 1
 Route14:     x15 + x19 - x31  = 1
 Route15:     x16 + x17 - x41  = 1
 HR0:         - 78 x17 + x20 + 18 x38 + 8 x39 + 26 x40 + 26 x41  = 0
 HR1:         - 99 x18 + x21 + 8 x32 + 24 x33 + 13 x34 + 21 x35 + 27 x36
              + 6 x37  = 0
 HR2:         - 98 x19 + x22 + 25 x26 + 20 x27 + 10 x28 + 5 x29 + 28 x30
              + 10 x31  = 0
 DevLB0:      x20 + x23 >= 69
 DevUB0:      - x20 + x23 >= -75
 DevLB1:      x21 + x24 >= 74
 DevUB1:      - x21 + x24 >= -80
 DevLB2:      x22 + x25 >= 60
 DevUB2:      - x22 + x25 >= -66
Bounds
 0 <= x1 <= 1
 0 <= x2 <= 1
 0 <= x3 <= 1
 0 <= x4 <= 1
 0 <= x5 <= 1
 0 <= x6 <= 1
 0 <= x7 <= 1
 0 <= x8 <= 1
 0 <= x9 <= 1
 0 <= x10 <= 1
 0 <= x11 <= 1
 0 <= x12 <= 1
 0 <= x13 <= 1
 0 <= x14 <= 1
 0 <= x15 <= 1
 0 <= x16 <= 1
 0 <= x17 <= 1
 0 <= x18 <= 1
 0 <= x19 <= 1
 0 <= x26 <= 1
 0 <= x27 <= 1
 0 <= x28 <= 1
 0 <= x29 <= 1
 0 <= x30 <= 1
 0 <= x31 <= 1
 0 <= x32 <= 1
 0 <= x33 <= 1
 0 <= x34 <= 1
 0 <= x35 <= 1
 0 <= x36 <= 1
 0 <= x37 <= 1
 0 <= x38 <= 1
 0 <= x39 <= 1
 0 <= x40 <= 1
 0 <= x41 <= 1
Binaries
 x1  x2  x3  x4  x5  x6   x7  x8  x9
 x10  x11  x12  x13 x14  x15  x16
 x17 x18  x19 x26  x27  x28 x29
 x30  x31 x32  x33 x34  x35  x36
 x37 x38 x39  x40  x41
```

Figure 4.26: "Converting" problem formulation

The solution of the "converting" problem is presented below (see Figure 4.27). Apparently, no route is deleted from a roster. Each flight hour deviation from a crew members' balance window costs 1000. The total flight hours deviation of all crew member is 54 hours, hence the cost is 54000.

At this point, we would like to comment that Crew Member 3 is assigned a much heavier flight load than his/her upper bound, while Crew Member 0 is given just 3 flight hours more than his/her upper limit. If the master problem considered the roster quality cost then, probably, it would have assigned the most heavy roster to Crew Member 1, since his/her upper limit is the largest among the three [80 flight hours] and the lightest schedule to Crew Member 2, whose upper limit is the lowest [66 flight hours]. Such an assignment would reduce the final schedule cost, but it would require more computational effort to be produced.



```
|- Number of Crew Members = 3
|- Number of Routes = 16


|--- Number of routes that remained uncovered : 0

 | Crew Member 0 ...
   Route=6 is given to Crew Member 0.
   Route=7 is given to Crew Member 0.
   Route=13 is given to Crew Member 0.
   Route=15 is given to Crew Member 0.
  - Crew Member 0 :   FlightHoursLowerBound = 69  ,  FlightHoursUpperBound = 75
    - Total Flight Hours for this Crew Member : 78


 | Crew Member 1 ...
   Route=0 is given to Crew Member 1.
   Route=1 is given to Crew Member 1.
   Route=3 is given to Crew Member 1.
   Route=4 is given to Crew Member 1.
   Route=9 is given to Crew Member 1.
   Route=11 is given to Crew Member 1.
  - Crew Member 1 :   FlightHoursLowerBound = 74  ,  FlightHoursUpperBound = 80
    - Total Flight Hours for this Crew Member : 99


 | Crew Member 2 ...
   Route=2 is given to Crew Member 2.
   Route=5 is given to Crew Member 2.
   Route=8 is given to Crew Member 2.
   Route=10 is given to Crew Member 2.
   Route=12 is given to Crew Member 2.
   Route=14 is given to Crew Member 2.
  - Crew Member 2 :   FlightHoursLowerBound = 60  ,  FlightHoursUpperBound = 66
    - Total Flight Hours for this Crew Member : 98



 --> Final Schedule Cost = 54000.000000
```

Figure 4.27: "Converting" problem's solution

Chapter 5

Computational Results

In our program, we developed 2 alternative designs. In Design 1, we utilize the proposed set-cover master problem formulation. When we find an integer, set-covering solution, we employ the ”converting” problem to go from the set-covering to a set-partitioning solution, which has practical meaning, as explained in previous chapters (see Figure 5.1). In Design 2, we use the traditional set-partition master problem formulation and we arrive to a set-partitioning solution (see Figure 5.1). It is important to note, although it will become apparent soon, that the set-partitioning solutions of the 2 lines are different.

We test our program in three different flight load conditions, we present the results of our tests in tables and demonstrate the comparative performance of the alternative designs in line and bar charts. We compare the two designs in each flight load condition separately through a number of different cases, and then we discuss their overall performance.

All the experiments were performed on an Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz processor with 8GB system memory.

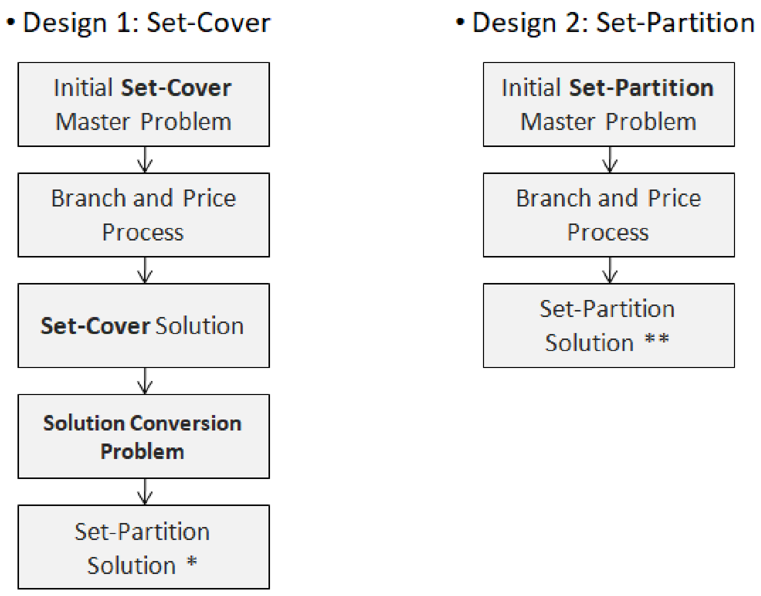
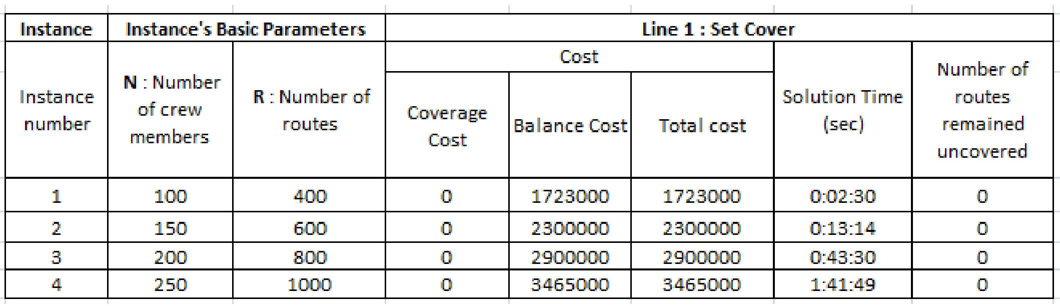

• Design 1: Set-Cover
Initial Set-Cover Master Problem
Branch and Price Process
Set-Cover Solution
Solution Conversion Problem
Set-Partition Solution *
• Design 2: Set-Partition
Initial Set-Partition Master Problem
Branch and Price Process
Set-Partition Solution **

Figure 5.1: Program’s Lines

5.1 Condition 1 [C1]: Light Flight Load

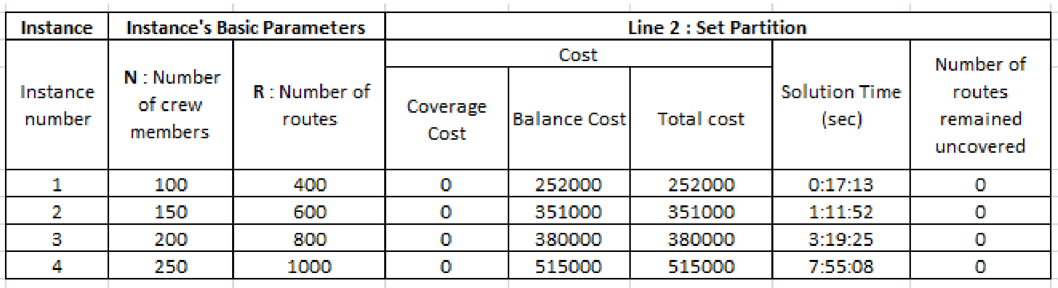
In the first condition, which is called ”Light”, each crew member has to cover 4 pairings on average. It was found that in this condition, in order to provide full pairing coverage, almost all crew member could fly less than or equal to the lower bound of their respective balance window. As a result, we could say that the crew members are ”under-utilized” in this condition.

5.1.1 Line Results Comparison C1: Tables



Instance	**Instance's Basic Parameters**		**Line 1 : Set Cover**				
			Cost				
Instance number	**N** : Number of crew members	**R** : Number of routes	Coverage Cost	Balance Cost	Total cost	Solution Time (sec)	Number of routes remained uncovered
1	100	400	0	1723000	1723000	0:02:30	0
2	150	600	0	2300000	2300000	0:13:14	0
3	200	800	0	2900000	2900000	0:43:30	0
4	250	1000	0	3465000	3465000	1:41:49	0

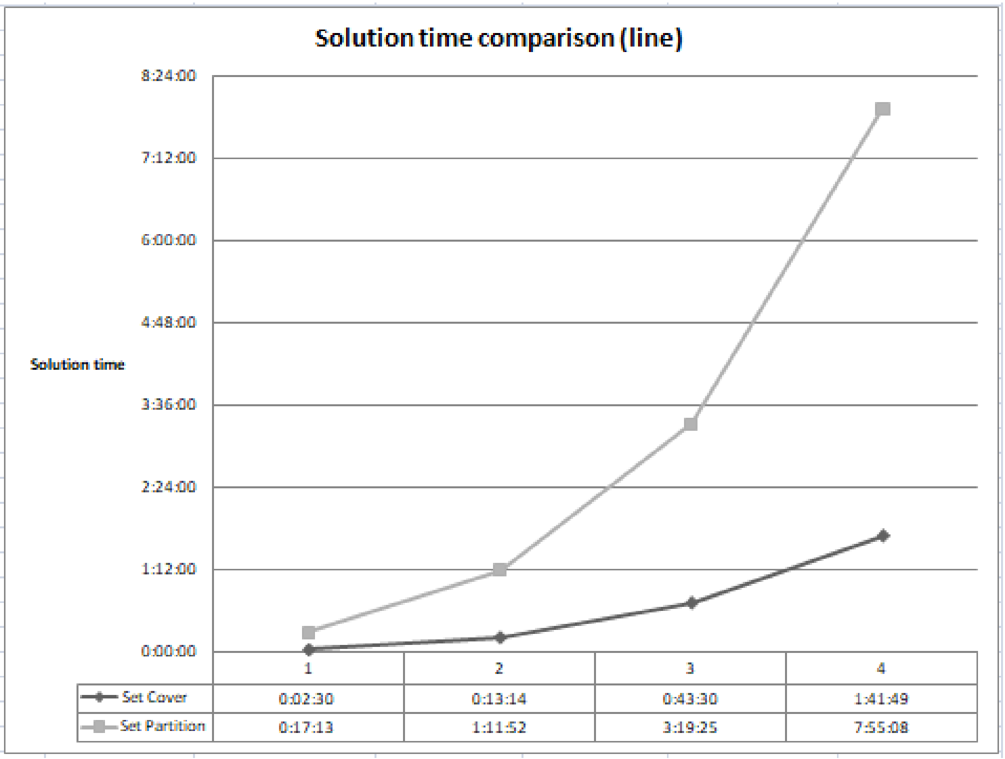
Figure 5.2: Computational results [C1] - Line 1: Set-Cover Formulation



Instance	**Instance's Basic Parameters**		**Line 2 : Set Partition**				
			Cost				
Instance number	**N** : Number of crew members	**R** : Number of routes	Coverage Cost	Balance Cost	Total cost	Solution Time (sec)	Number of routes remained uncovered
1	100	400	0	252000	252000	0:17:13	0
2	150	600	0	351000	351000	1:11:52	0
3	200	800	0	380000	380000	3:19:25	0
4	250	1000	0	515000	515000	7:55:08	0

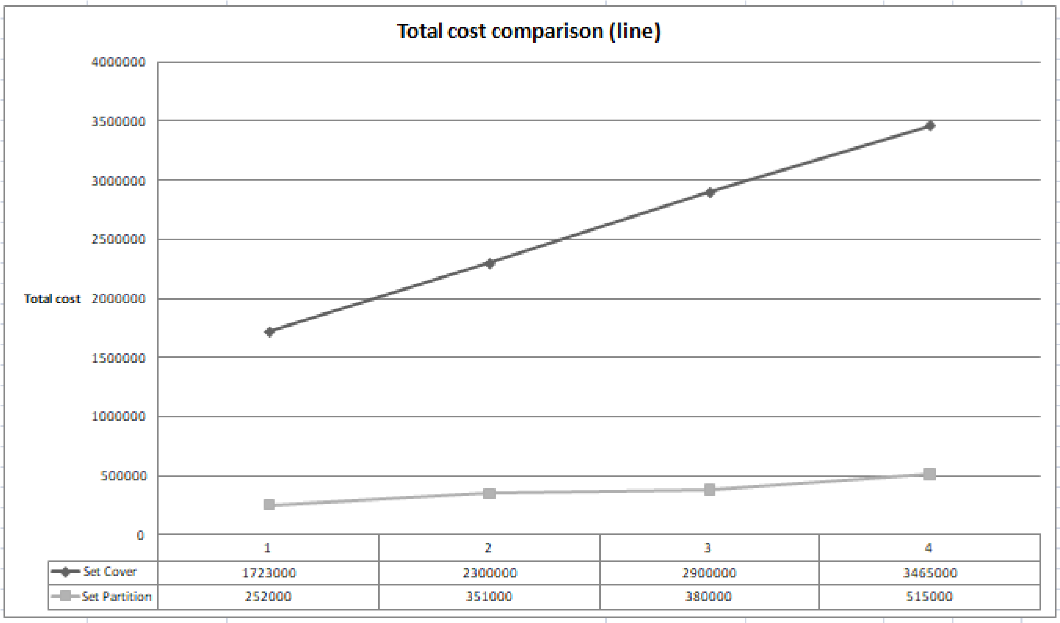
Figure 5.3: Computational results [C1] - Line 2: Set-Partition Formulation

5.1.2 Line Results Comparison C1: Line charts



	1	2	3	4
Set Cover	0:02:30	0:13:14	0:43:30	1:41:49
Set Partition	0:17:13	1:11:52	3:19:25	7:55:08

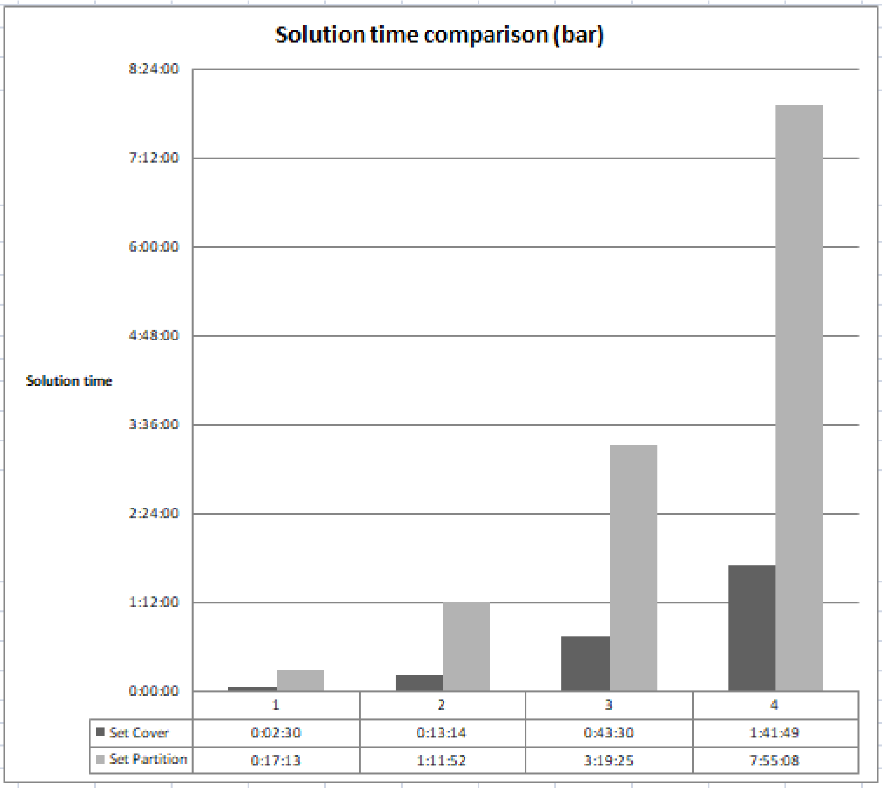
Figure 5.4: Solution time comparison [C1] - Line chart



	1	2	3	4
Set Cover	1723000	2300000	2900000	3465000
Set Partition	252000	351000	380000	515000

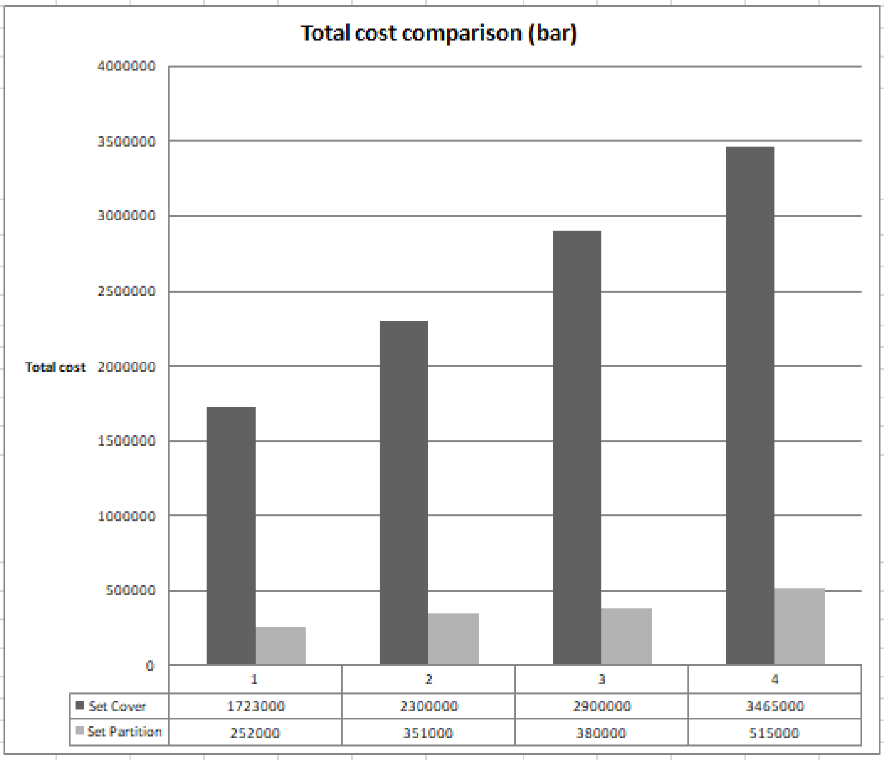
Figure 5.5: Total cost comparison [C1] - Line chart

5.1.3 Line Results Comparison C1: Bar charts



	1	2	3	4
■ Set Cover	0:02:30	0:13:14	0:43:30	1:41:49
■ Set Partition	0:17:13	1:11:52	3:19:25	7:55:08

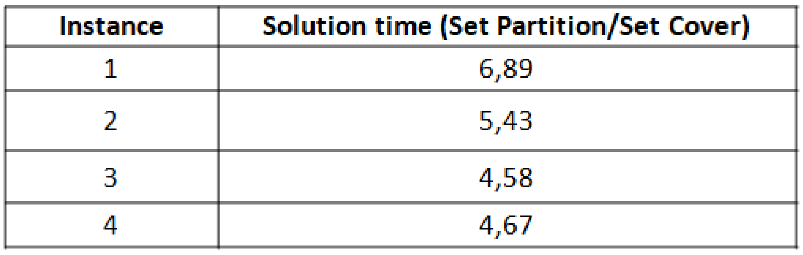
Figure 5.6: Solution time comparison [C1] - Bar chart



	1	2	3	4
■ Set Cover	1723000	2300000	2900000	3465000
■ Set Partition	252000	351000	380000	515000

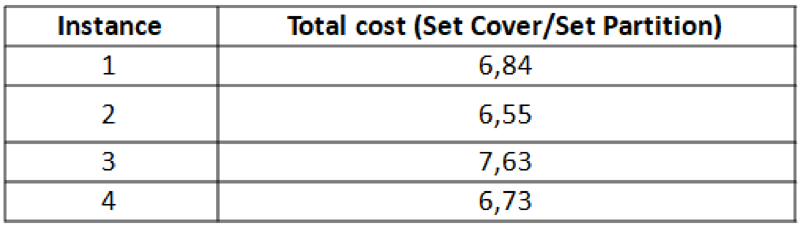
Figure 5.7: Total cost comparison [C1] - Bar chart

5.1.4 Line Results Comparison C1: Comparative Tables



Instance	Solution time (Set Partition/Set Cover)
1	6,89
2	5,43
3	4,58
4	4,67

Figure 5.8: Solution times [C1] - Comparative table



Instance	Total cost (Set Cover/Set Partition)
1	6,84
2	6,55
3	7,63
4	6,73

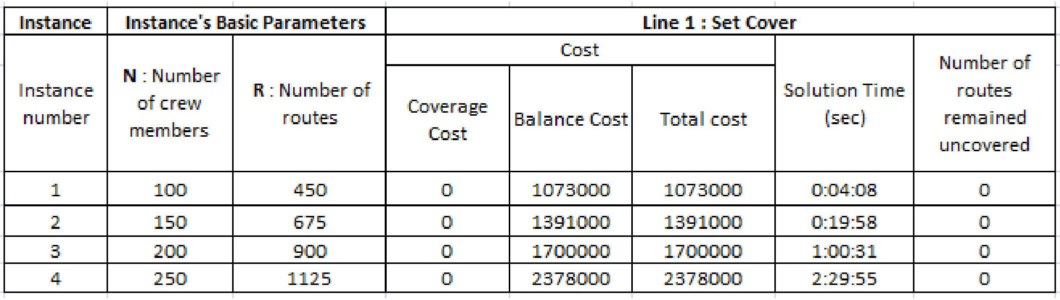
Figure 5.9: Total cost [C1] - Comparative table

The solution time is 5.39 times greater in Line 2 (Set-Partition) on average, while the total cost is 6.94 times greater in Line 1 (Set-Cover) on average, for the light flight load condition.

5.2 Condition 2 [C2]: Normal Flight Load

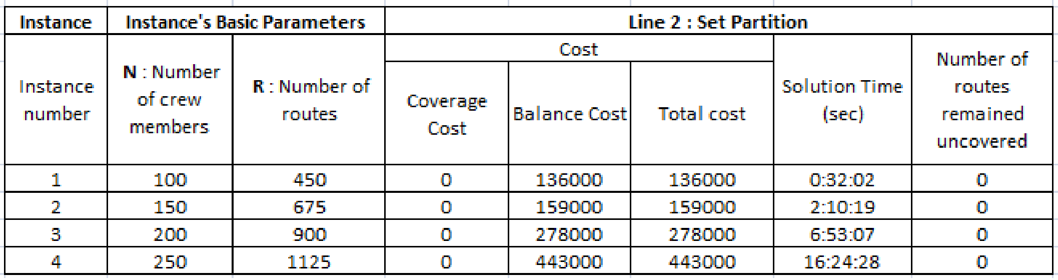
In the second condition, which is called ”Normal”, each crew member has to cover 4.5 pairings on average. It was found that in this condition, in order to provide full pairing coverage, many crew member could undertake flight load that lies within the bounds of their respective balance window. As a result, we could say that the crew members are ”normally-utilized” in this condition.

5.2.1 Line Results Comparison C2: Tables



Instance	Instance's Basic Parameters		Line 1 : Set Cover				
			Cost				
Instance number	**N** : Number of crew members	**R** : Number of routes	Coverage Cost	Balance Cost	Total cost	Solution Time (sec)	Number of routes remained uncovered
1	100	450	0	1073000	1073000	0:04:08	0
2	150	675	0	1391000	1391000	0:19:58	0
3	200	900	0	1700000	1700000	1:00:31	0
4	250	1125	0	2378000	2378000	2:29:55	0

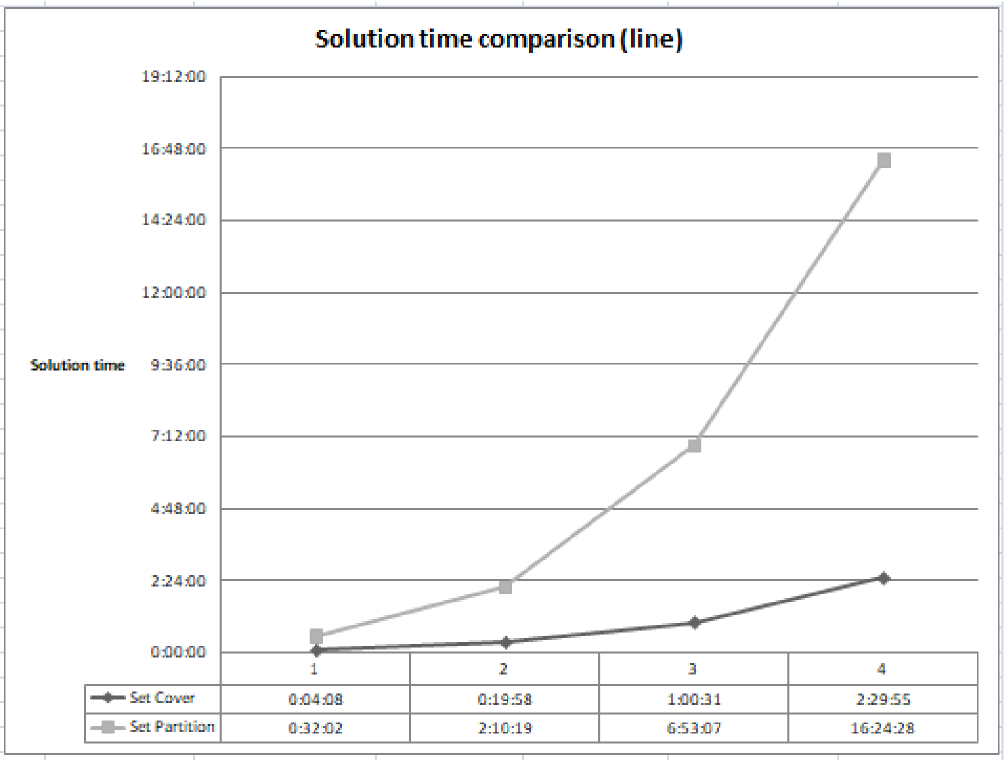
Figure 5.10: Computational results [C2] - Line 1: Set-Cover Formulation



Instance	Instance's Basic Parameters		Line 2 : Set Partition				
			Cost				
Instance number	**N** : Number of crew members	**R** : Number of routes	Coverage Cost	Balance Cost	Total cost	Solution Time (sec)	Number of routes remained uncovered
1	100	450	0	136000	136000	0:32:02	0
2	150	675	0	159000	159000	2:10:19	0
3	200	900	0	278000	278000	6:53:07	0
4	250	1125	0	443000	443000	16:24:28	0

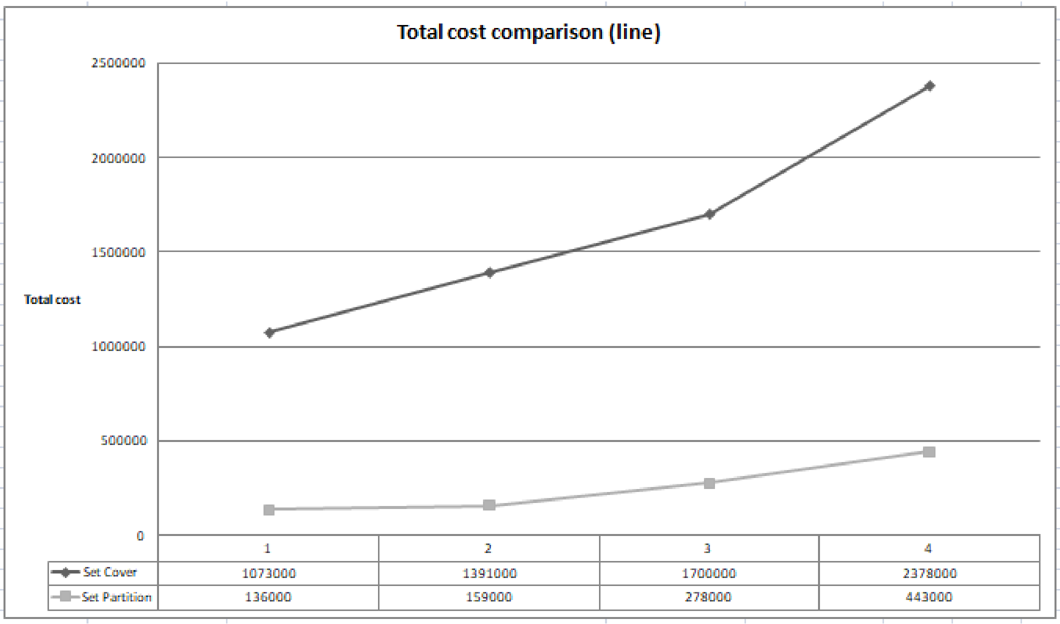
Figure 5.11: Computational results [C2] - Line 2: Set-Partition Formulation

5.2.2 Line Results Comparison C2: Line charts



	1	2	3	4
Set Cover	0:04:08	0:19:58	1:00:31	2:29:55
Set Partition	0:32:02	2:10:19	6:53:07	16:24:28

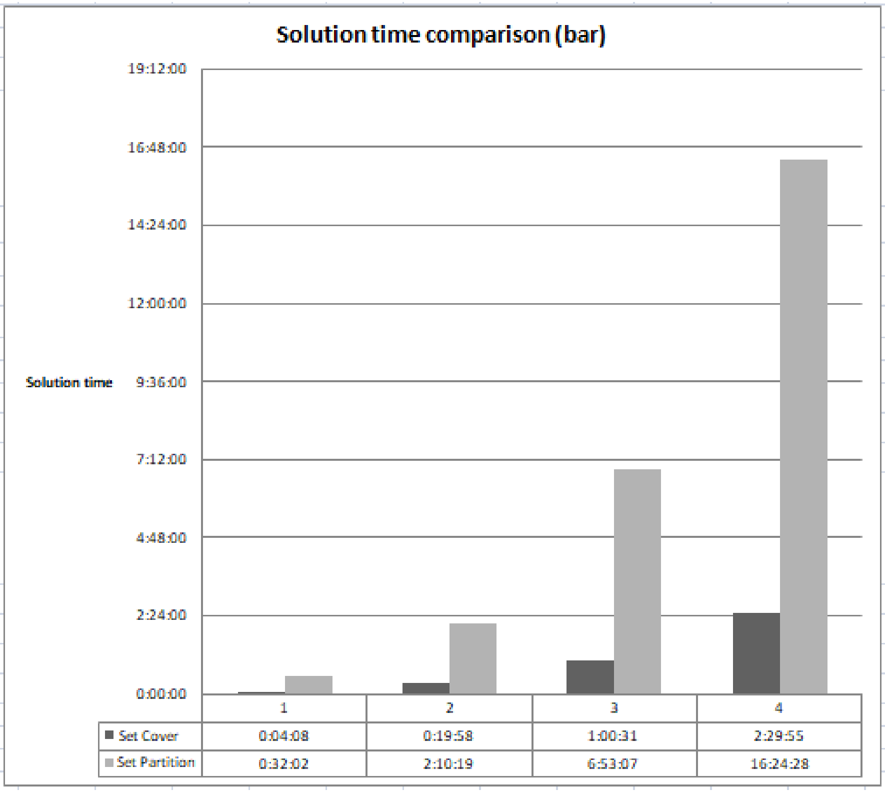
Figure 5.12: Solution time comparison [C2] - Line chart



	1	2	3	4
Set Cover	1073000	1391000	1700000	2378000
Set Partition	136000	159000	278000	443000

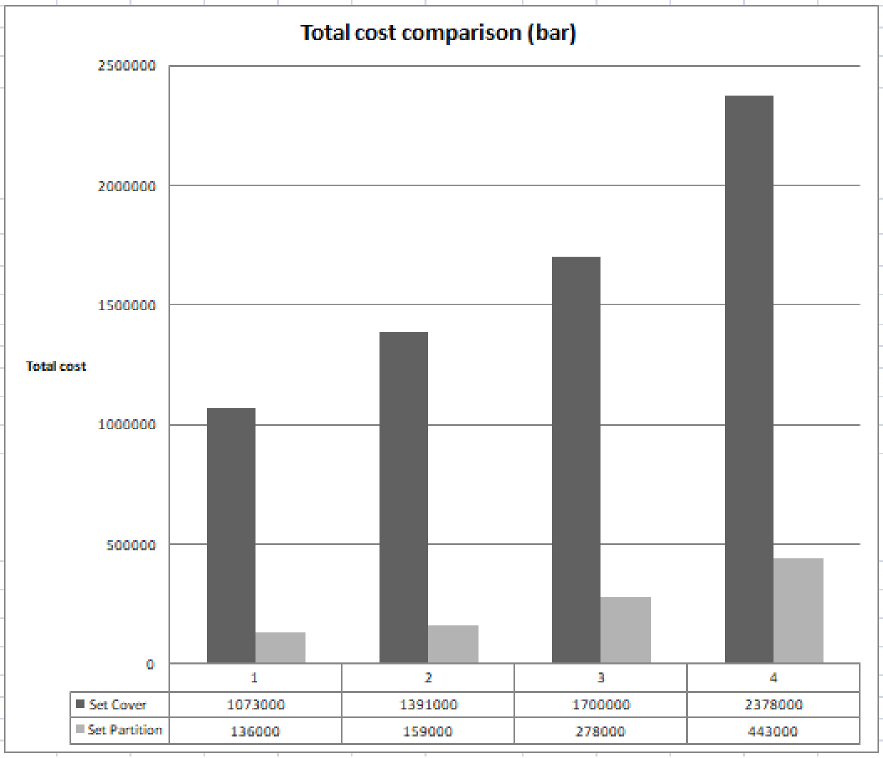
Figure 5.13: Total cost comparison [C2] - Line chart

5.2.3 Line Results Comparison C2: Bar charts



	1	2	3	4
■ Set Cover	0:04:08	0:19:58	1:00:31	2:29:55
■ Set Partition	0:32:02	2:10:19	6:53:07	16:24:28

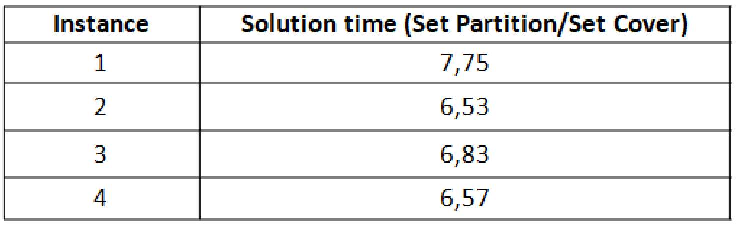
Figure 5.14: Solution time comparison [C2] - Bar chart



	1	2	3	4
■ Set Cover	1073000	1391000	1700000	2378000
■ Set Partition	136000	159000	278000	443000

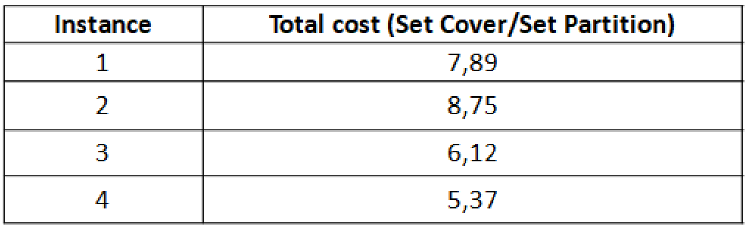
Figure 5.15: Total cost comparison [C2] - Bar chart

5.2.4 Line Results Comparison C2: Comparative Tables



Instance	Solution time (Set Partition/Set Cover)
1	7,75
2	6,53
3	6,83
4	6,57

Figure 5.16: Solution times [C2] - Comparative table



Instance	Total cost (Set Cover/Set Partition)
1	7,89
2	8,75
3	6,12
4	5,37

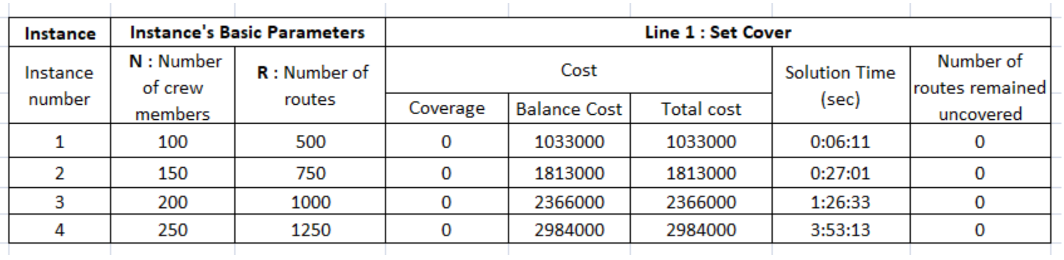
Figure 5.17: Total cost [C2] - Comparative table

The solution time is 6.92 times greater in Line 2 (Set-Partition) on average, while the total cost is 7.03 times greater in Line 1 (Set-Cover) on average, for the normal flight load condition.

5.3 Condition 3 [C3]: Heavy Flight Load

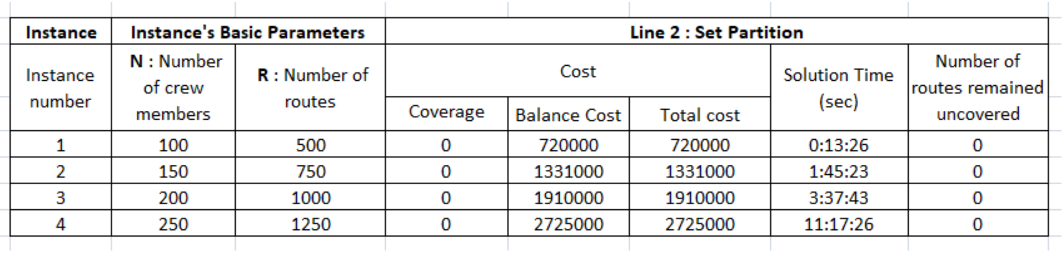
In the third condition, which is called ”Heavy”, each crew member has to cover 5 pairings, on average. It was found that in this condition, in order to provide full pairing coverage, almost all crew member had to fly equal to or more than the upper bound of their respective balance window. Therefore, we could say that the crew members are ”over-utilized” in this condition.

5.3.1 Line Results Comparison C3: Tables



Instance	**Instance's Basic Parameters**		**Line 1 : Set Cover**				
Instance number	**N** : Number of crew members	**R** : Number of routes	Cost			Solution Time (sec)	Number of routes remained uncovered
			Coverage	Balance Cost	Total cost		
1	100	500	0	1033000	1033000	0:06:11	0
2	150	750	0	1813000	1813000	0:27:01	0
3	200	1000	0	2366000	2366000	1:26:33	0
4	250	1250	0	2984000	2984000	3:53:13	0

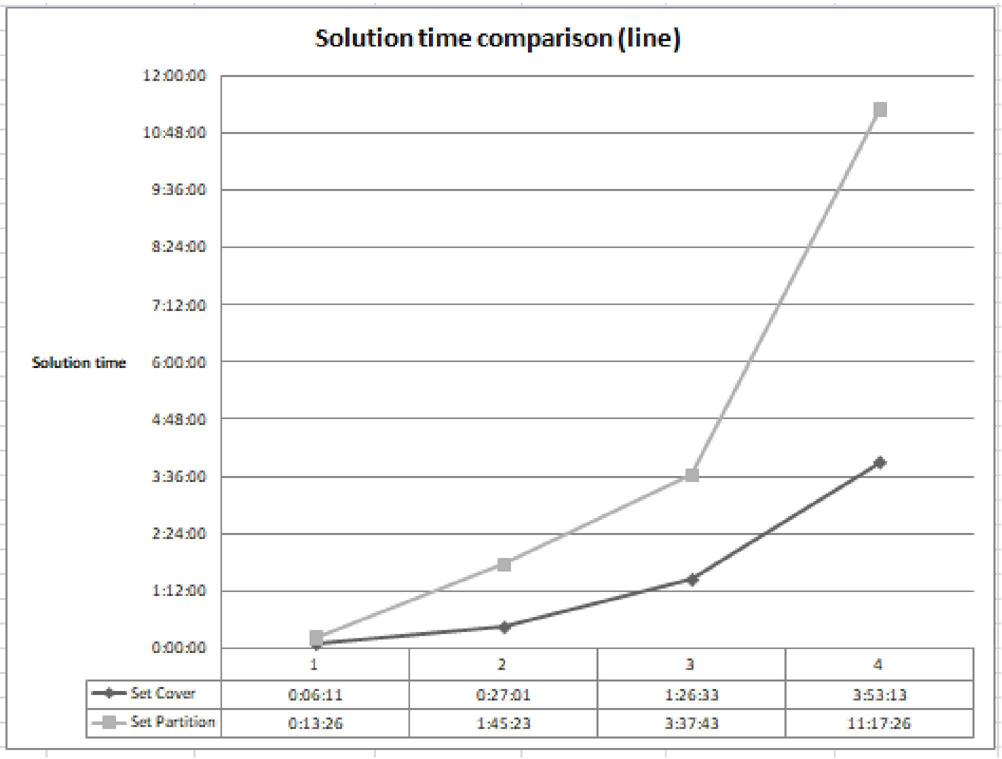
Figure 5.18: Computational results [C3] - Line 1: Set-Cover Formulation



Instance	**Instance's Basic Parameters**		**Line 2 : Set Partition**				
Instance number	**N** : Number of crew members	**R** : Number of routes	Cost			Solution Time (sec)	Number of routes remained uncovered
			Coverage	Balance Cost	Total cost		
1	100	500	0	720000	720000	0:13:26	0
2	150	750	0	1331000	1331000	1:45:23	0
3	200	1000	0	1910000	1910000	3:37:43	0
4	250	1250	0	2725000	2725000	11:17:26	0

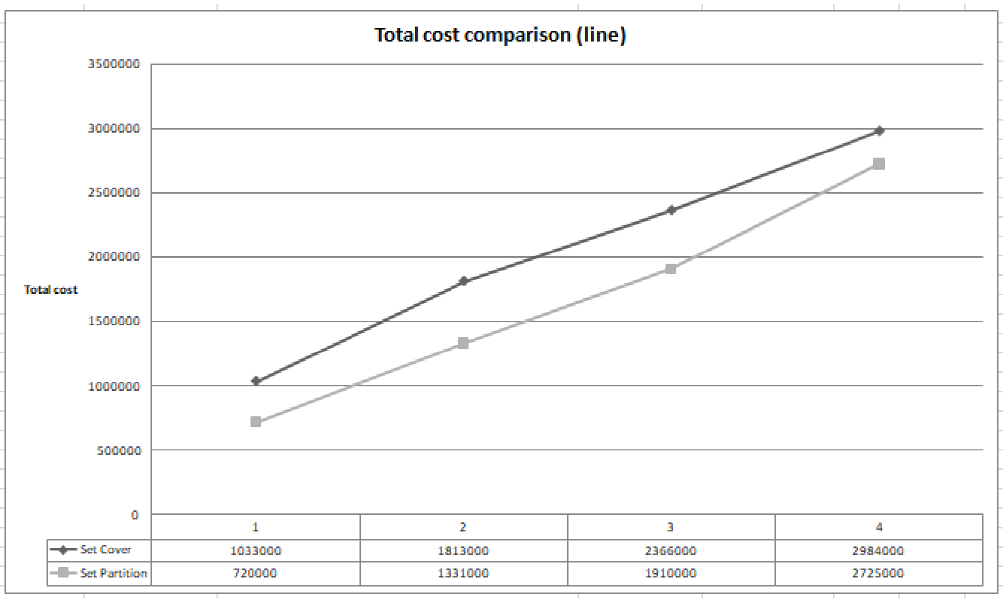
Figure 5.19: Computational results [C3] - Line 2: Set-Partition Formulation

5.3.2 Line Results Comparison C3: Line charts



	1	2	3	4
Set Cover	0:06:11	0:27:01	1:26:33	3:53:13
Set Partition	0:13:26	1:45:23	3:37:43	11:17:26

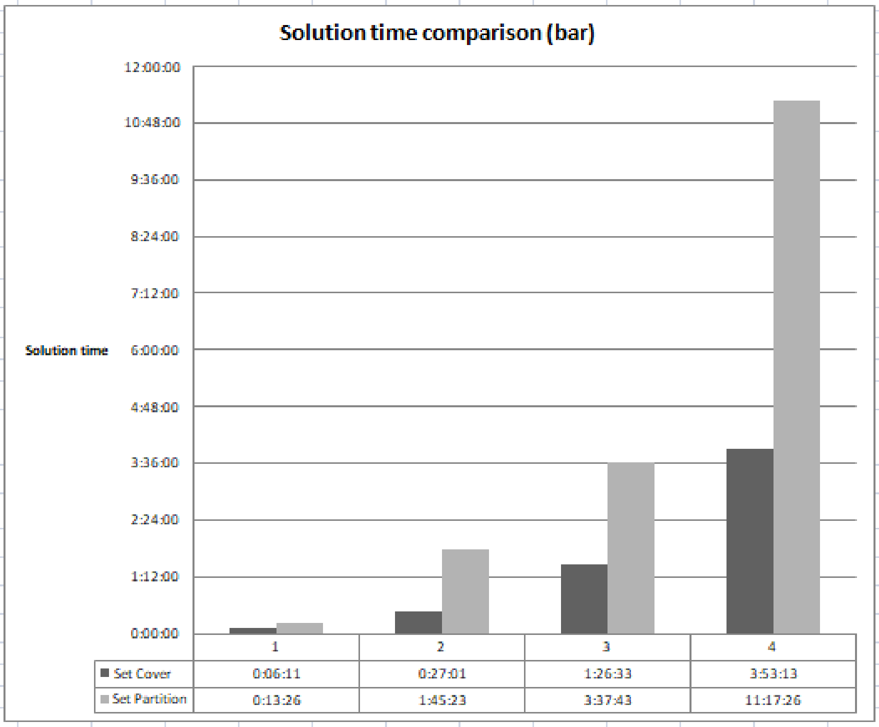
Figure 5.20: Solution time comparison [C3] - Line chart



	1	2	3	4
Set Cover	1033000	1813000	2366000	2984000
Set Partition	720000	1331000	1910000	2725000

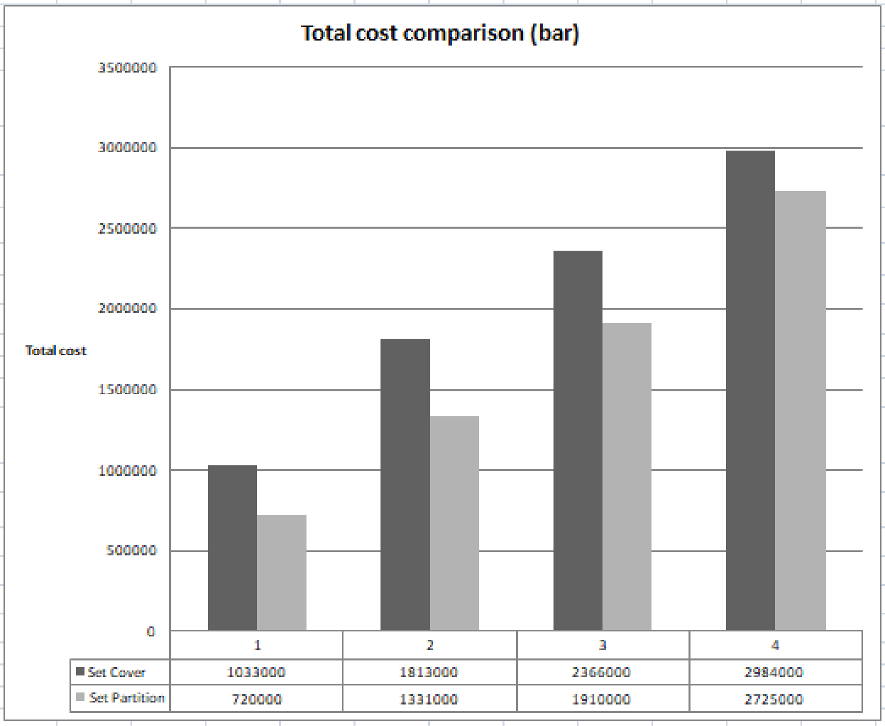
Figure 5.21: Total cost comparison [C3] - Line chart

5.3.3 Line Results Comparison C3: Bar charts



	1	2	3	4
■ Set Cover	0:06:11	0:27:01	1:26:33	3:53:13
▨ Set Partition	0:13:26	1:45:23	3:37:43	11:17:26

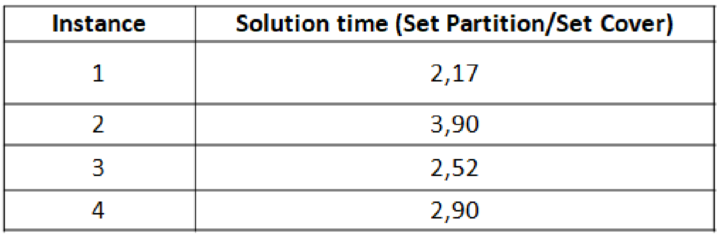
Figure 5.22: Solution time comparison [C3] - Bar chart



	1	2	3	4
■ Set Cover	1033000	1813000	2366000	2984000
▨ Set Partition	720000	1331000	1910000	2725000

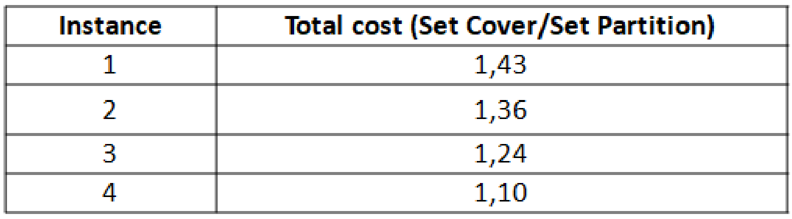
Figure 5.23: Total cost comparison [C3] - Bar chart

5.3.4 Line Results Comparison C3: Comparative Tables



Instance	Solution time (Set Partition/Set Cover)
1	2,17
2	3,90
3	2,52
4	2,90

Figure 5.24: Solution times [C3] - Comparative table



Instance	Total cost (Set Cover/Set Partition)
1	1,43
2	1,36
3	1,24
4	1,10

Figure 5.25: Total cost [C3] - Comparative table

The solution time is 2.87 times greater in Line 2 (Set-Partition) on average, while the total cost is 1.28 times greater in Line 1 (Set-Cover) on average, for the heavy flight load condition.

5.4 Results' Discussion

To begin with, as anticipated, Line 1, which utilizes the set-cover formulation for the master problem, solves the scheduling problem faster than Line 2. The difference is greater in the normal flight load condition [C2], in which the solution time of Line 2 is 6.92 times greater than that of Line 1, on average. The smaller difference is seen in the heavy load condition [C3], in which the solution time of Line 2 is 2.87 times greater than that of Line 1, on average.

However, when it comes to the total cost of the final solution, Line 2, which utilizes the set-partition formulation for the master problem, produces solutions of superior quality than that of Line 1 in every condition. The difference is greater in the normal flight load condition [C2], in which the total cost of Line 1 solution is 7.03 times larger than that of Line 2, on average. The smaller difference is found in the heavy load condition [C3], in which the total cost of Line 1 is 1.28 times greater than that of Line 2, on average.

The reasons behind these variations in the performance of the two lines, are the two differences in the master problem formulations; the absence of the roster quality cost in the objective function and the use of inequalities instead of equalities in the pairing constraint set of the set-cover formulation. To better illustrate, we will try to draw a parallel to our situation. Imagine that there are many balls of different sizes and colors in front of you and you have a bucket. In one case, you are asked to fill your bucket with red balls. In another case, you are asked to put 10 red balls with diameter between 3 and 5 centimeters in your bucket. In the second case, you would have to spot the red balls, measure the balls' diameter and also you would have to count the balls that you put in your bucket, whereas in the first one you would just have to spot the red balls and place them in your bucket until the bucket is full. It is easy to understand that the second case would require much more time than the first but the product would be more refined. Apparently, the first case in our parallel is Line 1, with the set-cover master problem formulation, while the second is Line 2, with the set-partition master problem formulation.

The small difference in the final cost between the two Lines in the heavy load condition [C3] is remarkable. One possible explanation is that, since the flight load is so heavy and many crew members have to fly for the maximum time permitted by the rules, which for most crew members is much more than the upper bound of their respective balance window, Line 2 does not have much "room for optimization". Another indication for that is the much higher total cost of the solutions produced by Line 2 in the heavy load in comparison to the solutions produced by the same design in the other two conditions (see Figure 5.26).

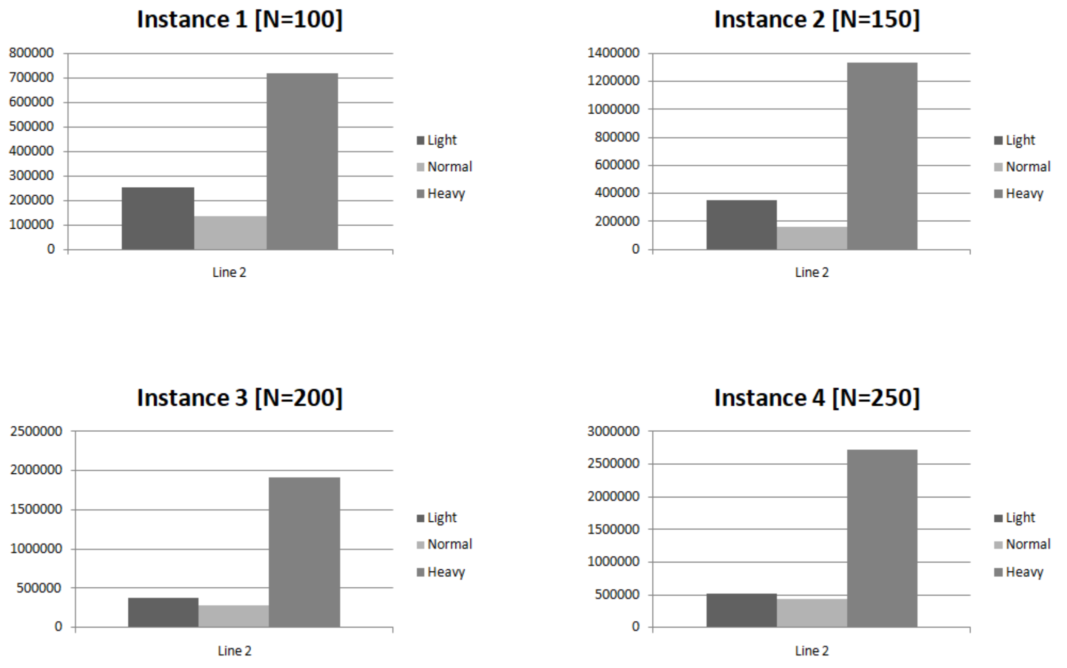


Figure 5.26: Total cost comparison between conditions - Line 2 [Set-partition]

Another thing that the reader can observe in Figure 5.26, is that the total cost in the light load condition for Line 2 is smaller than in the normal load condition, which may seem strange, since the pairings are less in every instance of light load while the crew members that have to cover them are the same. At this point, we remind the reader that the quality cost or, in our case, the balance cost is imposed in case of deviation of the actual flight hours of a crew member from his/her associated balance window. The actual flight hours may be above the upper bound of that window, or below the lower bound, in case of deviation. In the light load condition, many crew member are under-utilized, i.e. their assigned flight hours are below the lower bound of their balance window, and thus a penalty is imposed. In the normal load condition, the actual flight hours of many crew members can be within the balance window. As a result, there is no penalty for many of them and the total cost is lower compared to the other conditions.

In the light flight load condition, the total cost of Line 1 is larger in all cases (see Figure 5.27) compared to the solutions of Line 1 for the other two conditions. A possible explanation for is the fact that it assigns many pairings to some crew members, whereas some are left without a pairing (they are assigned an empty roster). This can be explained by the fact that the quality of rosters, which in our case is the time balance, is not taken into consideration in the set-cover formulation and only coverage of all pairings is the objective. As a result, due to the design of the solution methodology, the program covers all the pairings with a few crew members, who are over-utilized and lets others under-utilized or even without a duty. This leads to unbalanced solutions which translate to large quality costs.

An empty roster has generally a higher quality cost than a heavily loaded roster in our cases because, in general, the flight hours target, which is the center of the balance windows, is set around 75 flight hours, which is closer to the upper legal limit of flight hours a crew member can operate in a planning horizon (which is set to 100) than to zero, which are the flight hours of an empty roster. In the light condition solutions, there are many crew members with an empty roster and thus the total cost is higher in this condition than the heavy load condition, in which almost all crew members have a "heavy" schedule (or are over-utilized).

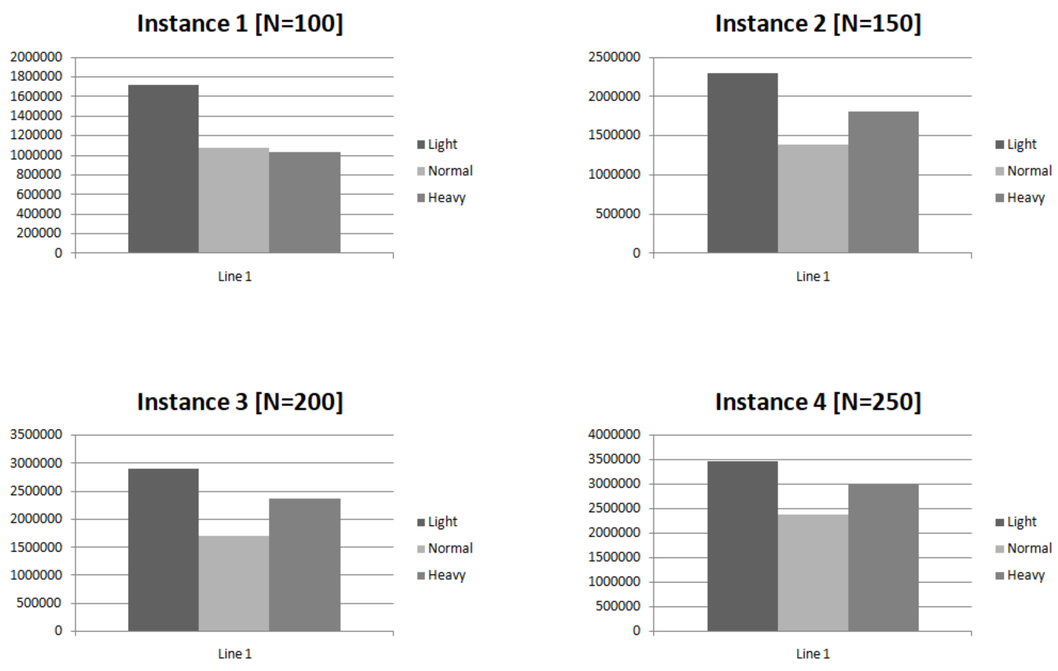


Figure 5.27: Total cost comparison between conditions - Line 1 [Set-cover]

The fact that the actual flight hours of crew members can be within their balance window in the normal load condition translates into much "room for optimization" for Line 2. This becomes obvious in Figure 5.28. Although the problem is smaller in the normal condition than in the heavy load condition, since there are more pairings in the latter and the number of crew members is the same, the solution time of the normal load condition is greater for all equivalent cases.

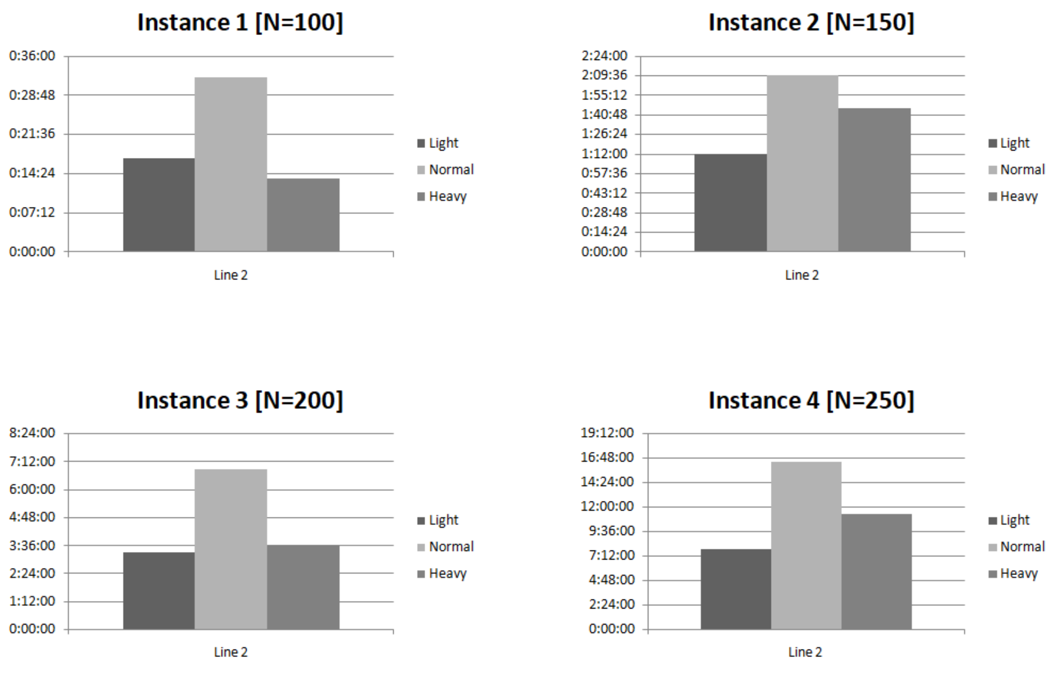


Figure 5.28: Solution time comparison between conditions - Line 2 [Set-partition]

In Line 1, solution time increases as the problems become larger (see Figure 5.29).

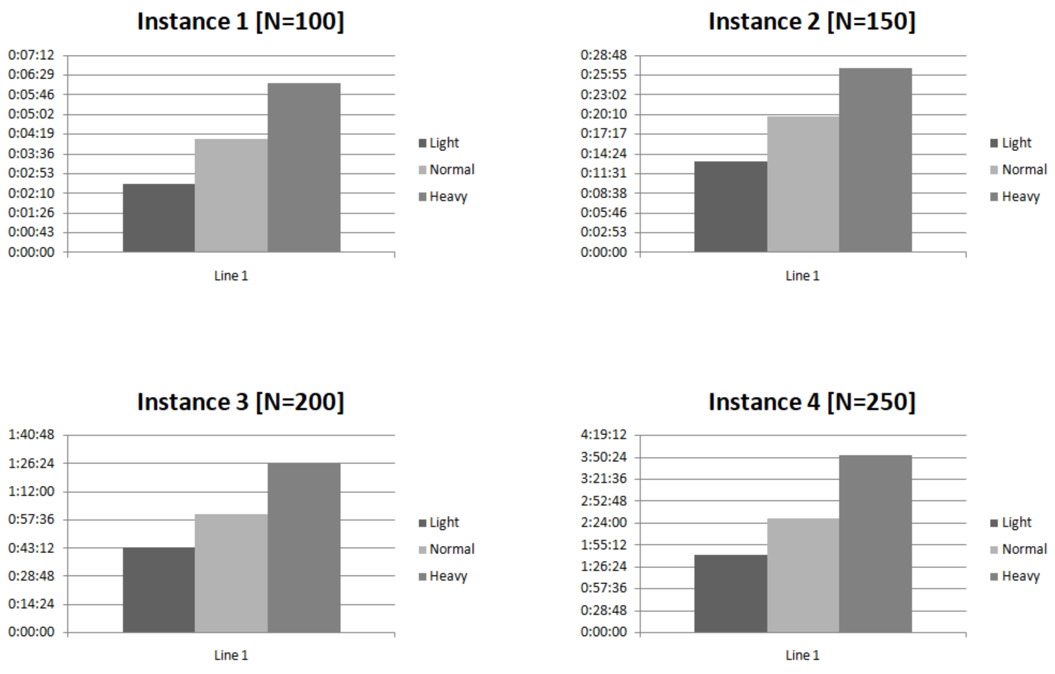


Figure 5.29: Solution time comparison between conditions - Line 1 [Set-cover]

5.5 Conclusion

We have seen that the total cost of the solutions produced by Line 1 is much greater than that of those produced by Line 2, with the exception of the heavy load condition, in which the gap is much smaller. As a consequence, we would recommend the utilization of Line 1, only in heavy loaded cases and only if a fast answer is required.

The quality of rosters is very important for the airlines as explained numerous times in the present thesis. As a result, the final schedules should be as good as possible and such schedules can be produced by Line 2.

The substantial difference in the solution times of the two alternative designs could be utilized in occasions that the roster quality is irrelevant. A situation like this is encountered in the first phases of crew rostering, where schedule planners have a set of pairings generated previously in the pairing generation phase and wish to know just if a given set of crew members can operate all the pairings (or provide full coverage of the flight duties). In this occasion, our approach could help practitioners identify the best possible coverage very fast, and thus in case of uncovered duties they would have more time to make the necessary corrective moves.

Chapter 6

Future Work

6.1 Getting Closer to Reality

One of the future goals is to test our program in more realistic environments. That would include the input of real-life data sets of crew members and pairings, the scheduling of which is required for the operation of a real airline firm. That could happen in collaboration with an real industry actor, such as an airline or an airline consulting firm.

An extension that would also make our problem more realistic would be the addition of more regulations regarding the legality of rosters. In our current application, only a handful of those rules are included. However, in reality there are much more.

Moreover, another addition that would make our program more realistic would be the incorporation of more criteria associated with schedules' quality. In our application only the time balance is determining the quality of a roster. Other quality costs that could be included are a penalty for short rest time and a penalty for early start after days off.

In reality, crew members often declare their preferences concerning their schedule. This can be done in various ways and the approach is different in the U.S.A. and Europe. They can either make bids for already created schedules, or they can state their work related preferences for the planning horizon before the construction of rosters, in order to be taken into consideration by the schedulers. The inclusion of the preferences of the crew members into the construction of schedules would add more value to our program since it would make it more realistic.

6.2 Experimentation: Program's Parameters

To begin with, we could experiment with the performance parameters of our program and identify the optimal combination of their values for different flight loads, if we find a relation between them. Such parameters are the backtrack and MIP gap tolerances, the upper limit of nodes in the search tree and of backtracks allowed and also the number of network pairing and fictitious nodes' labels.

Another experimentation would be to test alternative branching strategies. In our program we utilize a single variable branching approach. In the future, we could test whether branching on a set of variables could produce better results.

Instead of utilizing a shortest path algorithm for acyclic networks for solving each column generation sub-problem, we could use commercial optimization software. In that case, the column generation sub-problem would be formulated as an optimization problem, rather than a Shortest Path Problem with Side-Constraints. In that way, we would be able to compare the results of these two alternative ways to model and solve the column generation problem.

Another idea to be tested would be the combination of the set-partition and the set-cover formulations. In our program, we developed two alternative designs or lines. Line 1 utilizes the set-cover, while Line 2 the set-partition formulation. In a third design, or Line 3, we could start with the set-cover formulation, and then use the final solution as a seed for the set-partition formulation (see Fig. 6.1). In other words, we would do a "warm" start for Line 2, by first following Line 1.

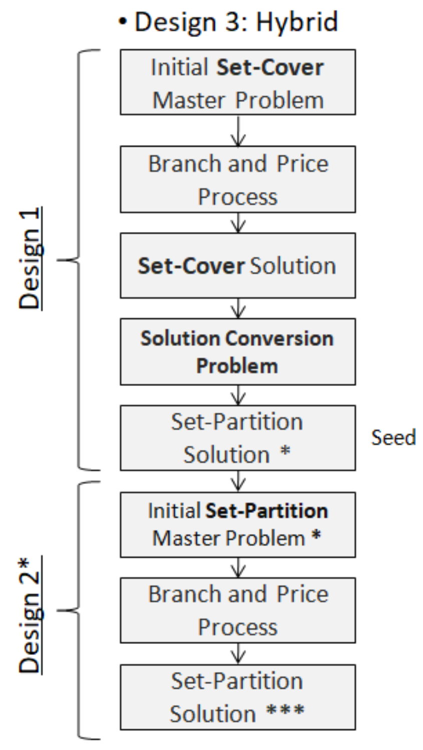


Figure 6.1: Proposed Hybrid Design

6.3 Further Exploration of Potential Utility

Motivated by the airlines' desire to maximize the flight duty coverage that can be achieved with a given set of crew members, crew planning practitioners quite often solve heavily unbalanced (and in fact, overloaded) crew scheduling problems, aiming to estimate the extent of anticipated uncovered duties with the intention to increase crew capacity for covering the extra duties at a later stage. The alternative set-cover formulation that we propose may well help them identify the achievable coverage much faster than the traditional set-partition formulation that is utilized. In the future, we would like to test cases in which the problem is overloaded, i.e. the crew members set can not cover all the pairings that is required to be operated. The results produced in this thesis are promising and our approach could be proved a valuable tool in the hands of airline managers, which could be used in the early stages of crew scheduling, saving them much time. However, special tests of overloaded cases are needed to prove the usefulness of our method.

6.4 Program's Extensions and New Applications

In the future, we would like to further research the potential benefits that our approach could bring to schedule planners in their quest to maximize crew utilization as well. A possible application based on our method could provide planners a recommendation of the crew set that most efficiently, fully covers a given set of pairings, by identifying its composition from a pool of crew members

Finally, we could integrate our program in a platform that would address the crew scheduling problem in its entirety. To do that, we should first develop another program to tackle the pairing generation problem and then integrate the two in a single platform which would take as an input a set of flights and a set of crew members and would produce the final schedules of each crew member.

Chapter 7

Summary

In the present thesis we addressed the crew rostering problem, which is a part of the crew scheduling problem of the airline industry. In this problem, monthly schedules are constructed for the crew members of an airline for a planning horizon of a month. These schedules are comprised of blocks of flight legs, called pairings, interrupted by rest intervals in the home base of the crew.

The crew rostering problem is particularly challenging due to the sheer size of it and because of a set of tight regulations that dictate the legality of crew rosters. As a result, advanced solution methodologies have to be employed in order to solve this problem efficiently, which translates to producing cost efficient schedules for the crew. Low cost rosters, usually correspond to lower labor costs for airlines and more content crew members, both vital for the smooth operation and profitability of every airline firm.

In our application, we implemented a column generation-based solution methodology modified to address integer programming problems with a huge number of decision variables, such as the crew rostering problem, called branch and price method. The implementation of this solution framework was done in a program developed in C programming language, with the utilization of IBM ILOG CPLEX, which is a commercial optimization software, in parts of our program.

The implementation of the proposed solution framework proved to be a difficult task, mainly because of the many algorithmic parts that compose it. To ensure computational efficiency we took advantage of the dynamic memory allocation features provided by C and also utilized linked lists, which are an example of dynamic data structure that uses pointers for its implementation. Although barely a reference was made here, the largest part of our work was coding and finding out ways to implement the algorithmic framework in a computer program. It was a strenuous experience in many points but the author found it very didactic and learned many things about coding and more importantly about developing an elaborate program, which is actually a project.

The code developed will be available to students of the Systems Optimization Laboratory in the future, so that they can learn from it and also improve it. This is considered a significant contribution on the author's part, because the computer program developed is a product of hundreds of hours of work.

Returning to our subject, the solution framework was developed in order to test alternative master problem formulations and how these affect the final solution of the program. In our work, two alternative formulations were tested; the standard, set-partition formulation and a "novel", set-cover formulation.

The results from our tests showed that, although the set-cover formulation produces inferior quality solutions compared to the set-partition formulation and thus it is not recommended for the construction of the final rosters assigned to crew members, its utilization can potentially prove useful to the schedule planners in earlier stages of the scheduling process. More specifically, due to its speed, our proposed formulation could inform the practitioners whether a given set of crew members could fully cover a particular set of pairings, much faster than the traditional set-partition approach. As a result, the schedule planners would have more time to make the appropriate corrective moves, such as utilizing a different set of crew members to cover the same set of pairings, with the aim to provide full flight coverage.

The planners could feed the program with overloaded cases, i.e. cases in which the set of crew members can not operate all the flights required, in their quest to maximize crew utilization. Since the crew costs are the second highest operating cost of an airline firm, behind fuel, and taking into consideration that profit margins are relatively low for the airline compared to other industries, mainly due to the fierce competition in the air transportation market, maximization of crew utilization could prove a tremendous competitive advantage for airlines and in some cases could decide their growth or their demise.

We hope that further exploration and development of the current approach could lead to a useful tool for real-life airlines and consultancy firms. We also hope that the computer program developed for the needs of the present thesis will be finally a part of a platform to tackle the crew scheduling program at first, and the scheduling of airline operations subsequently, in their entirety. Although rather ambitious, this target is possible.

I would like to close this thesis with a personal message to the reader; many things in our lives are difficult. They require much time and patience. There are some points, when in the middle of a project or a race, that we may feel that we can not go further. The abandonment of our cause seems very tempting at such moments. But in these dark times, I believe, we should learn to value ourselves, our time, our sacrifices for our cause. I believe that we should focus on the end, on the finish line and project ourselves as winners, as those that managed to get through the difficulties and finish what they started. And it is these images that will give us strength and that will lead us out of the dead end that was previously in front of us.

Endure - Advance - Excel

Chapter 8

Bibliography

[1] Mohan Akella, Avijit Sarkar, and Sharad Gupta. Branch and price: Column generation for solving huge integer programs. *University at Buffalo, created Oct*, 29:1–20, 2011.

[2] Cynthia Barnhart, Peter Belobaba, and Amedeo R. Odoni. Applications of operations research in the air transport industry. 37:368–391, 11 2003.

[3] Cynthia Barnhart, Amy M Cohn, Ellis L Johnson, Diego Klabjan, George L Nemhauser, and Pamela H Vance. Airline crew scheduling. In *Handbook of transportation science*, pages 517–560. Springer, 2003.

[4] Cynthia Barnhart, Ellis L Johnson, George L Nemhauser, Martin WP Savelsbergh, and Pamela H Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations research*, 46(3):316–329, 1998.

[5] IBM Knowledge Center. IBM ILOG CPLEX Optimization Studio CPLEX User's Manual. `https://www.ibm.com/support/knowledgecenter/SSSA5P_12.7.0/ilog.odms.studio.help/pdf/usrcplex.pdf`. [Online; accessed 18 Jun, 2018].

[6] IBM Knowledge Center. What is Column Generation? `https://www.ibm.com/support/knowledgecenter/SSSA5P_12.6.2/ilog.odms.cplex.help/CPLEX/UsrMan/topics/discr_optim/eg_col_gen/02_col_gen_defn.html`. [Online; accessed 12 Jun, 2018].

[7] Guy Desaulniers, Jacques Desrosiers, and Marius M Solomon. *Column generation*, volume 5. Springer Science & Business Media, 2006.

[8] Martin Desrochers and François Soumis. A generalized permanent labelling algorithm for the shortest path problem with time windows. *INFOR: Information Systems and Operational Research*, 26(3):191–212, 1988.

[9] Jacques Desrosiers and Marco E Lübbecke. A primer in column generation. In *Column generation*, pages 1–32. Springer, 2005.

[10] Michel Gamache, François Soumis, Gérald Marquis, and Jacques Desrosiers. A column generation approach for large-scale aircrew rostering problems. *Operations research*, 47(2):247–263, 1999.

[11] Balaji Gopalakrishnan and Ellis L Johnson. Airline crew scheduling: state-of-the-art. *Annals of Operations Research*, 140(1):305–337, 2005.

[12] Industry High Level Group (IHLG). *Aviation Benefits 2017*. URL: https://www.icao.int/sustainability/Documents/AVIATION-BENEFITS-2017-web.pdf.

[13] Stefan Irnich and Guy Desaulniers. Shortest path problems with resource constraints. In *Column generation*, pages 33–65. Springer, 2005.

[14] Hamid Kharraziha, Marek Ozana, and Sami Spjuth. Large scale crew rostering. *Carmen Research and Technology Report CRTR-0305, Carmen Systems AB, Gothenburg, Sweden*, 47, 2003.

[15] Diego Klabjan. Large-scale models in the airline industry. In *Column generation*, pages 163–195. Springer, 2005.

[16] Niklas Kohl and Stefan E Karisch. Airline crew rostering: Problem types, modeling, and optimization. *Annals of Operations Research*, 127(1-4):223–257, 2004.

[17] George Kozanidis. Branch and price for covering shipments in a logistic distribution network with a fleet of aircraft. *Optimization Methods and Software*, 33(2):221–248, 2018.

[18] Sylvie Lavoie, Michel Minoux, and Edouard Odier. A new approach for crew pairing problems by column generation with an application to air transportation. *European Journal of Operational Research*, 35(1):45–58, 1988.

[19] Ernesto de Queiros Vieira Martins. An algorithm for ranking paths that may contain cycles. *European Journal of Operational Research*, 18(1):123–130, 1984.

[20] Jaroslav Moravek. A note upon minimal path problem. *Journal of Mathematical Analysis and Applications*, 30(3):702–717, 1970.

[21] David R Morrison, Sheldon H Jacobson, Jason J Sauppe, and Edward C Sewell. Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. *Discrete Optimization*, 19:79–102, 2016.

[22] Air Transport Association of America. *The Airline Handbook*. Air Transport Association of America, 2001.

[23] Jay M Rosenberger, Hee S Hwang, Ratna P Pallerla, Adnan Yucel, Ron L Wilson, and Ed G Brungardt. The generalized weapon target assignment problem. Technical report, TEXAS UNIV AT ARLINGTON, 2005.

[24] Moritz Ferdinand Scharpenseel. Consequences of eu airline deregulation in the context of the global aviation market. *Nw. J. Int'l L. & Bus.*, 22:91, 2001.

[25] Gang Yu. *Operations research in the airline industry*, volume 9. Springer Science & Business Media, 2012.