LAGRANGIAN RELAXATION FOR AIRPORT GATE ASSIGNMENT PROBLEM

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
MASTER OF SCIENCE
IN
INDUSTRIAL ENGINEERING

By
Göksu Ece Okur
July 2023

Lagrangian Relaxation for Airport Gate Assignment Problem
By Göksu Ece Okur
July 2023

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Özlem Karsu(Advisor)

Oğuz Solyalı(Co-Advisor)

Sakine Batun

Taghi Khaniyev

Approved for the Graduate School of Engineering and Science:

Orhan Arıkan
Director of the Graduate School

ABSTRACT

LAGRANGIAN RELAXATION FOR AIRPORT GATE ASSIGNMENT PROBLEM

Göksu Ece Okur
M.S. in Industrial Engineering
Advisor: Özlem Karsu
July 2023

In this study we focus on the Airport Gate Assignment Problem that minimizes the total walking distance of passengers while ensuring that the number of aircraft assigned to apron is at its minimum. We utilize an alternative formulation for the problem compared to the ones in the literature and propose approaches based on Lagrangian Relaxation so as to obtain tight lower bounds. The method also harnesses the power of a good initial upper bound and provides good quality solutions. To the best of our knowledge, the current studies in the literature rely only on upper bounds or the linear relaxation lower bounds to assess the quality of heuristic solutions. We propose using the tighter Lagrangian Relaxation based bounds as a better reference to assess solution quality. Our computational experiments demonstrate that our Lagrangian relaxation based method returns strong lower bounds and good quality upper bounds that are comparable to the state-of-the art results from the literature.

Keywords: Airport Gate Assignment Problem, Lagrangian relaxation, subgradient method, bundle method.

ÖZET

HAVAALANI KAPI ATAMA PROBLEMİNDE LAGRANGE GEVŞETMESİ

Göksu Ece Okur
Endüstri Mühendisliği, Yüksek Lisans
Tez Danışmanı: Özlem Karsu
Temmuz 2023

Bu çalışmada aprona atanan uçak sayısının en az sayıda tutulmasını sağlayarak yolcuların toplam yürüme mesafesini en küçükleyen havaalanı kapı atama problemi ele alınmaktadır. Literatürde kullanılan formülasyonlara alternatif bir formülasyon kullanılmakta ve sıkı alt sınırlar elde edebilmek için Lagrange gevşetmesine dayalı yöntemler önerilmektedir. Yöntem, aynı zamanda iyi başlangıç üst sınırlarının gücünden faydalanarak iyi kalitede çözümler vermektedir. Bildiğimiz kadarıyla literatürdeki çalışmaların çoğu, sezgisel sonuçlarını performans bazlı üst sınır karşılaştırması ya da doğrusal programlama gevşetmesinden elde edilen alt sınırlardan faydalanarak ölçmektedir. Deneysel sonuçlar, kullanılan Lagrange gevşetmesine dayalı yöntemlerin sıkı alt sınırlar ve iyi kalitede üst sınırlar verdiğini göstermektedir. Elde edilen değerler literatürde var olan sezgisel sonuçların performansını ölçmede kullanilabilir.

Anahtar sözcükler: Havaalanı Kapı Atama Problemi, Lagrange Gevşetmesi, alt gradyan yöntemi, yığın yöntemi.

Acknowledgement

First of all, I would like to express my sincere gratitude to my advisors, Assoc. Prof. Özlem Karsu and Prof. Oğuz Solyalı for their exceptional expertise and valuable guidance. I have learned so much from them and, I am also grateful for their valuable support.

I would like to thank Asst. Prof. Taghi Khaniyev and Asst. Prof. Sakine Batun, for accepting to read and review my thesis. I am grateful for their valuable comments.

I am thankful to my parents Neşe Okur and Koray Okur, for their constant support. I am thankful to my brother Mert Okur for always cheering me on and never letting me feel alone. I am also thankful to my aunt Yeliz Kilis Yıldırım for making Ankara a home to me.

I would like to express my gratitude to Yüce Hasan Kılıç for being my best friend and for his trust and encouragement. I would like to thank Ecenur Oğuz, Deniz Şahin, Pelin Erdil, and Ece Ünal for being such great friends and always being there for me.

I would like to thank my dear friends Elif Rana Yöner, Ali Eren Demir, Deniz Akkaya, Elif Sena Işık and Ali İlhan Haliloğlu for their great friendship and support. I am also grateful to them for making my days fun and always motivating me. I can not imagine a Bilkent without them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Increase in the use of airport services led to the search for more efficient ways to provide this service [3]. Therefore, airport operations has been a widely studied area of research. Airport slot allocation, airport runway scheduling, and aircraft baggage allocation are some of the problems studied in airport operations. One of the major problems in this area is the assignment of aircraft to gates, which is known as the Airport Gate Assignment Problem (AGAP) in the literature. This problem is challenging as one has to consider gate availability and flight conflicts when arranging the assignments, as well as the utilities and preferences of multiple parties involved such as airline operators, passengers and airport management.

Airports are generally divided into categories according to the type of flights (domestic/international). Some minor airports only provide domestic flights, whereas some larger airports provide both domestic and international airports. Domestic and international gates are separated in those airports as international flights require identity and passport control operations. When making the assignments, such dynamics and problem specific settings should be taken into account.

There are many different objectives considered for the AGAP. Daş et al. [3] separate the objectives into four main groups. The first group is the passenger-oriented objective functions such as minimizing total passenger walking distance [4, 5] and objectives considering baggage transport distance [6]. The second group is airport/airline-oriented objective functions such as minimizing the number of aircraft assigned to the apron [7], objectives considering the preferences of airlines for specific gates [8, 9], and maximizing number of passengers at gates close to shopping facilities [10]. The third group is

robustness-oriented objective functions such as minimizing expected number of gate conflicts [11, 12]. The last group is multi-objective models [13]. Our study combines the airport and passenger-oriented concerns and focuses on a hierarchical AGAP, where we aim to minimize the total passenger walking distance while first ensuring that the number of aircrafts assigned to the apron is at its minimum.

Since the problem is challenging to solve, especially for real-life based larger instances, a vast majority of the studies in the literature propose (meta)-heuristic solution approaches. However, such approaches do not have an optimality or a quality guarantee, making it difficult to see how good the returned solutions are. In this work, we focus on finding good quality lower bounds for the AGAP considered in [14, 15]. In particular, we utilize Lagrangian Relaxation for the AGAP that aims to minimize the total walking distance of passengers while keeping the number of aircrafts assigned to the apron at its minimum. We consider two methods for solving the Lagrangian Dual problem that arises: subgradient optimization and bundle method.

The remainder of the paper is structured as follows. Literature review can be found in Chapter 2, Details of the mathematical model of the AGAP formulation and Lagrangian relaxation can be found in Chapter 3 and Chapter 4, respectively. Solution approaches for the Lagrangian Dual are presented in Chapter 5. Computational results are provided in Chapter 6.

Chapter 2

Literature Review

In this work we apply Lagrangian relaxation-based approaches on the AGAP. As our study lies in the intersection of Lagrangian relaxation and AGAP, we will discuss the related literature in two parts: we will first mention the relevant studies on various AGAP variants and then discuss some recent successful implementations of Lagrangian relaxation-based approaches.

2.1 AGAP

Cheng,Ho & Kwan [16] classify the studies focusing on AGAP into two categories with respect to the type of the models used as static models and stochastic & robust models. Static AGAP only considers deterministic factors whereas stochastic & robust AGAP considers stochastic factors such as delays and flight cancellations. Some commonly used objectives are minimizing the walking distance of all the passengers, minimization of total passenger waiting time, number of unassigned flights, buffer time and gate conflicts.

Cheng, Ho & Kwan [16] also classify the AGAP literature according to the solution approaches. The solution methodologies are classified into three categories: expert system approaches, exact solution approaches, and heuristic approaches.

An expert system is a software system that is designed to simulate the behavior of human professionals. It makes decisions based on its database that consists of rules derived from human knowledge within a particular problem domain. Some studies that work on expert systems on AGAP are Brazile & Swigger [17] and Gosling [18].

A group of studies propose exact solution approaches for different AGAP variants. Mangoubi and Mathaisel [4] utilizes two methods: an LP relaxation and a heuristic approach with the objective of minimizing passenger walking distance. They use the data of Toronto International Airport. Passengers are divided in three groups: arrival passengers, departure passengers and transfer passengers. Transfer passenger walking distances are obtained from a uniform distribution assuming that moving to any gate is equally likely. LP and heuristic approach both result with near-optimal solutions while the heuristic requires less CPU time than the LP with no incumbent solution. They suggest using the solution of the heuristic as an incumbent solution for the LP relaxation to reduce the total CPU time used.

Hanghani & Chen [19] formulate a multi-slot gate assignment problem with the minimizing the total walking distance objective. Suggested MILP is solved using a brach &bound method and a heuristic.

Bolat [20] suggests a robust model, which aims to minimize the number of ungated flights and the disruptions when a deviation from the original schedule occurs. The objective of the proposed model is minimizing the range of the slack times, which is the time between departure of an preceding flight and the arrival of the next flight to the same gate. They use a branch bound algorithm and propose a heuristic. Computational experiments are conducted with the data of King Khaled International Airport. Proposed algorithm is superior to the current practice in terms of towed and ungated aircrafts.

Heuristic approaches are more often used due to the complexity of the problem. Yan & Huo [5] formulate the problem as a multi-objective 0-1 integer programming problem. They use weighted sum method, the column generation approach, the simplex method and the branch-and-bound algorithm in their solution approach. They combine their objectives, minimizing the total passenger walking distance and minimizing the total passenger waiting time by unifying them and taking their weighted sums. They conduct a case study on Chiang Kai-Shek (CKS) Airport, and the case study results are favorable.

Aktel et al. [21] utilize two metaheuristics, tabu search and simulated annealing in AGAP with the objectives: minimizing the number of ungated flights and minimizing the total walking distances. Simulated annealing and Tabu search give comparable results but Simulated annealing has better performance on average. Experimental results show that proposed algorithms are able to provide good quality solution for large scale problems within a reasonable time.

Karsu et al. [14] focus on AGAP with the minimizing the total walking distance objective. They develop bounding mechanisms and use the branch and bound, beam search and filtered beam search algorithms. Branch and bound algorithm can solve instances with up to 25 aircraft when there are 8 gates and up to 25 and 20 aircraft there are 10 and 12 gates, respectively within 1 hour time limit. The beam search algorithm provides good quality solutions to problem instances with up to 100 aircraft and 18 gates and 50 aircraft and 38 gates, while filtered beam search returns solutions for problems up to 200 aircraft and 38 gates.

Li et al.[22] propose a probability learning based heuristic algorithm to tackle the AGAP with the objective of minimizing the total passenger walking distance. They also separate the passengers into three groups. They introduce a reinforcement learning based approach and define their probability vectors as the likelihood of assigning an aircraft to a particular gate. Additionally, they utilize tabu search for local improvement.They test their algorithmic performance using data of Cheng et al. [16] (Incheon airport) and Karsu et al. [14] (Esenboğa airport & Atatürk airport). Computational experiments are favorable in terms of the quality of the upperbound within a reasonable time.

Karsu and Solyalı [15] formulate AGAP as a MILP model, and use a metheuristic approach. Objectives in the MILP model are minimizing the number of ungated flights and minimizing the total passenger walking distance objectives. They address these objective lexigrophically and minimizing the number of ungated fligts is their primary objective. Suggested matheuristic outperform other matheuristics in the literature.

2.2 Lagrangian Relaxation

Lagrangian relaxation has been extensively employed in the field of operations research for numerous years. Held and Karp [23, 24] are the pioneers in this technique with their adaptation on the travelling salesman problem. Since then, it has been adapted to generate bounds for a wide range of optimization problems and used in many areas including but not limited to scheduling problems, generalized assignment problems, set covering-partitioning problems, and location problems [25].

Since the Lagrangian Dual Problem (LD) is a non-differentiable optimization problem due to integer decision variables, we focus on nondifferentiable optimization techniques. In the literature they are classified according to their solution approaches: subgradient

type approaches and cutting plane type approaches [26]. Also, note that since our AGAP formulation has a minimization objective, Lagrangian dual problem has a maximization objective. Subgradient approaches are based on moving towards a direction with the magnitude determined by the step size at any iteration. The direction is computed using the subgradient and it is not necessarily an ascent direction. Due to the non-monotonicity of subgradient method, the steepest decent and ϵ-subgradient methods are designed [27]. Steepest decent methods move to the direction that gives the most decent (ascent for our problem) using the whole differential set, whereas ϵ-subgradient methods calculate the ascent direction using the ϵ-subdifferential set [27].

Subgradient-type approaches do not keep the memory of past iterations, whereas information obtained in past iterations are used in cutting plane methods. However, this may result in an excess use of memory and long computation times. Cutting plane type approaches are divided into three groups: Kelley's cutting plane algorithm, center based approaches [26, 27] and bundle type approaches. Drawbacks of the Kelley's classical cutting plane algorithm are its slow convergence and instability. The underlying cause of instability is the potential infeasibility of the dual problem (dual of LD), arising from the possible unboundedness of the primal problem (LD) in the initial iterations. This can lead to far away points in subsequent iterations and result in instability [28]. Center based approaches are based on choosing a center and using interior point concepts. We will not discuss them in detail as they are beyond the scope of this work. Bundle type approaches keep the information on the previously used Lagrangian coefficients in a set called Bundle. These approaches add a quadratic term in the objective function and penalize large steps to address instability. As this modification increases the solution time of the resulting models, the size of the bundle is kept small. Bundle approaches are ascent methods, if the bound found at the last iteration is worse than the best bound obtained, a null step is taken, new information is added to the bundle but a move is not made towards that direction. Hereby, we will mention a set of recent and relevant applications of Lagrangian relaxation to various Operations Research problems.

Contreras et. al. [29] apply Lagrangian relaxation on the capacitated hub location problem with single assignment. They use a four index formulation for their problem so as to obtain tighter bounds, and relax the constraints that link the assignment variables to the traffic variables, which allows decomposing the problem into smaller subproblems. They use the subgradient algorithm to find lower bounds and develop heuristics to obtain feasible solutions for their problem. They conduct their experiments with 10 to 200 nodes and obtain comparable results within a reasonable time.

Frangioni and Gallo [30] adopt the bundle method for the Linear Multi-commodity Min-Cost Flow Problem. They suggest a Lagrangian Variable Generation (LVG) Strategy in order to deal with less number of variables in their restricted problem. LVG strategy reduces the CPU time up to 20% of the CPU time without the LVG Strategy. Proposed Method is able to solve Multicommodity Min Cost Flow Problems with over one million variables within 15 minutes provided that problem has a Shortest Path Tree structure. Frangioni and Gorgone [31] use a bundle based method on the Dantzig-Wolfe Reformulation of the Fixed-Charge Multicommodity Min-Cost Flow problem. Kadri et al. [32] use the Lagrangian relaxation on the Multicommodity Capacitated Location Problem. They add copy variables and constraints on the interdepot flow variables and obtain subproblems that are decomposed by depot. They use the aggregated bundle method to solve the Lagrangian dual, and combine slope scaling with Lagrangian relaxation to obtain feasible solutions. Suggested feasible solutions were 1% away from the Lagrangian Dual computed utilizing bundle method. Computational results outperform Gendron at al. [33] that work on the same instance and is based on tabu search.

Crainic et al. [34] apply Lagrangian relaxation on the multicommodity capacitated fixed charge network design. They study two relaxations that allow them to obtain shortest path and knapsack problems after decomposition. They then use a bundle algorithm and the subgradient algorithm for solving the Lagrangian dual. Knapsack problem Relaxations are faster than Shortest Path Relaxations. However, shortest path relaxation with bundle method have the least gaps. Shibasaki et al. [35] implement the Volume Algorithm and the Bundle Method to obtain bounds on the large-scale Fixed-Charge Multicommodity Capacitated Network Design Problem. They obtain feasible solutions using the dual information in heuristic methods. Computational experiments on direct comparison of Volume algorithm and Bundle method demonstrate that Volume Algorithm outperforms bundle method in terms of average CPU Time and bound quality. This is explained by the faster iterations of Volume Algorithm. However, when used in the heuristic method suggested two methods were comparable in terms of CPU Time and solution quality.

Jena et al. [36] implement the subgradient and bundle methods on large-scale Dynamic Facility Location Problem with generalized modular capacities. They demonstrate that these methods outperform the state-of-the-art MIP solvers. They obtain feasible solutions at each iteration using the dual information, then they use upper bound improvement heuristics to increase the quality of their bound.

Similar to the above mentioned works, we investigate the performance of two approaches for solving the Lagrangian Dual that we obtain for the AGAP: the subgradient optimization and the bundle method.

Chapter 3

Problem Definition and Mathematical Model

AGAP consists of n planes that need to be assigned to one of m gates or apron. Apron has the longest distance to other gates and the entrance/exit of the airport. It also results in an additional cost of transportation to the airport. We take the arrival and departure times of planes as deterministic. The objective is to minimize the total distance traveled by the passengers while ensuring that the number of aircraft assigned to the apron is at its minimum. The minimum number of aircrafts assigned to the apron is found by solving a maximum cost network flow model for domestic and international flights separately [14]. This model provides the maximum number of aircraft assigned to regular gates, hence the minimum number assigned to apron. In this maximum cost network problem, nodes denote the aircraft and arcs represent the overlapping relations as follows: The arc cost is 1 if flights do not overlap and 0 otherwise. The total flow sent from the start node is the number of domestic/international gates. More details can be found in [14].

Passengers are divided into two groups: transfer passengers and non transfer passengers. Transfer passengers arrive at the airport by a flight and leave with another flight whereas non-transfer passengers use the entrance/exit before/after their flight. Distance that non-transfer passengers travel is the distance between the gate of their airplane and the airport entrance/exit, while the distance that transfer passengers travel is the distance between the gates of the aircraft they use.

We now give the proposed mixed integer linear programming formulation for the problem. For convenience, we use similar notation to previous papers [14, 15].

Sets	I_{Dt}: Set of domestic aircrafts overlapping in a time interval t. I_{It}: Set of international aircrafts overlapping in a time interval t. $I_t = I_{Dt} \cup I_{It}$. T: Set containing I_t for all t. $T = \{I_1, I_2, \ldots\}$ K'_D: Set of domestic gates. K'_I: Set of international gates. K_D: Set of domestic gates and apron. K_I: Set of international gates and apron.
Parameters	p_{ij} : Number of passengers between aircraft i and j (defined only for $(i, j) : i < j$). d_{kl} : Distance between gates k and l. g_i : Equal to D if aircraft i is domestic, equal to I if aircraft i is international $\forall i \in I$. NA^* : Number of planes assigned to apron. e_i : Number of passengers that are coming from entrance to aircraft i. f_i : Number of passengers that are going from aircraft i to exit. ed_k : Distance between gate k and the airport entrance/exit.
Decision Variables	x_{ik}: Equal to 1 if aircraft i is assigned to gate k, 0 otherwise. w_{ijkl}: Equal to 1 if aircrafts i and j are respectively assigned to gates k and l, 0 otherwise.

Overlapping flights are calculated by finding the maximal cliques with the polynomial time algorithm suggested in [37]. As the gates for domestic and international flights are separated, maximal cliques can be found separately. Therefore, we only give the pseudocode for maximal clique algorithm for domestic flights. Let I_D be the set of domestic aircrafts. Let a_{Di} (b_{Di}) be the arrival time (departure time) of domestic aircraft i, $\forall i \in I_D$.

Pseudocode for the maximal clique algorithm for domestic flights can be found in Algorithm 1 and more details can be found in [37].

Algorithm 1: Finding Maximal Cliques

```
p = 0, I_Dt = ∅,
s = min_{i: i∈I_D} a_Di, stop = 0;
while stop = 0 do
    f = min_{i:s<d_Di} b_i
    p = p+1
    K_p = {i ∈ I_D | a_Di < f ≤ b_Di}
    I_Dt = I_Dt ∪ {K_p}
    if ∃ a_Di ≥ f then
        s = min_{a_Di≥f} a_Di
    else
        stop = 1
```

MODEL MIP

$$\text{Min} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sum_{k \in K_{g(i)}} \sum_{l \in K_{g(j)}} p_{ij} d_{kl} w_{ijkl} + \sum_{i \in I} \sum_{k \in K_{g(i)}} (e_i + f_i) e d_k x_{ik} \tag{3.1}$$

$$\text{s.t.} \sum_{k \in K_{g(i)}} x_{ik} = 1 \qquad \forall i \in I \tag{3.2}$$

$$\sum_{i \in I_t} x_{ik} \leq 1 \qquad \forall I_t \in T, \forall k \in K \tag{3.3}$$

$$\sum_{i \in I} x_{i,m+1} = NA^* \tag{3.4}$$

$$\sum_{l \in K_{(g_j)}} w_{ijkl} = x_{ik} \qquad \forall i \in I \setminus \{n\}, j \in I, i < j, k \in K_{g(i)} \tag{3.5}$$

$$\sum_{l \in K_{(g_j)}} w_{jilk} = x_{ik} \qquad \forall i \in I, j \in I \setminus \{n\}, j < i, k \in K_{g(i)} \tag{3.6}$$

$$x_{ik} \in \{0, 1\} \qquad \forall i \in I, k \in K_{g(i)} \tag{3.7}$$

$$w_{ijkl} \in \{0, 1\} \qquad \forall i \in I \setminus \{n\}, j \in I, i < j, k \in K_{g(i)}, l \in K_{g(j)}. \tag{3.8}$$

(3.1) states that we minimize the total distance walked by passengers.

(3.2) states that each airplane needs to be assigned to a gate of its kind or apron.

(3.3) states that at most one airplane can use any given gate in any given interval.

(3.4) states that NA^* airplanes are assigned to apron.

(3.5) states that if airplane i is assigned to gate k then transfer passengers from airplane i to any airplane j (whose index is higher than i) must go through a gate l.

(3.6) states that if airplane i is assigned to gate k then transfer passengers from any airplane j(whose index is lower than i) to airplane i must come through a gate l.

(3.7) - (3.8) state that decision variables are binary.

There are also different linear formulations suggested for the AGAP in the literature. A four-indexed formulation is used by [38, 39, 40, 14] and a three-indexed formulation is used by [15]. We propose a different four-indexed formulation so as to obtain tighter lower bounds which we also verify by comparing three-indexed and four-indexed formulations in Chapter 6. Computational results show that our four-indexed formulation provides tighter lowerbounds in both MILP and LP relaxation models.

Note that problem is equivalent when constraint (3.8) is linearly relaxed due to constraints (3.5)-(3.7). The resulting MILP model is provided below:

MODEL MILP

$$\text{Min} \quad (3.1)$$

$$\text{s.t.} (3.2) - (3.7)$$

$$0 \leq w_{ijkl} \leq 1 \qquad \forall i \in I \setminus \{n\}, j \in I, i < j, k \in K_{g(i)}, l \in K_{g(j)}. \tag{3.9}$$

AGAP can also be modeled as a quadratic programming model as follows:

MODEL MIQP

$$\text{Min} \quad \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sum_{k \in K_{g(i)}} \sum_{l \in K_{g(j)}} p_{ij} d_{kl} x_{ik} x_{jl} + \sum_{i \in I} \sum_{k \in K_{g(i)}} (e_i + f_i) e d_k x_{ik} \tag{3.10}$$

$$\text{s.t.} (3.2) - (3.4), (3.7)$$

Provided models are for airports that have an apron, have symmetric distances ($d_{kl} = d_{lk}$) and have separate gates for domestic and interntional flights. For airports without an apron, constraint (3.4) is removed, and for models with a single type of gate a set K needs to be considered instead of sets K_I & K_D. For asymmetric layouts, all (i,j) pairs needs to be considered.

Chapter 4

Lagrangian Relaxation

The MILP formulation cannot be directly solved due to the number of decision variables it contains, but it may provide tight lower bounds through Lagrangian Relaxation. Constraints (3.5)-(3.6) can be relaxed using Lagrange multipliers α and β respectively. Then, the relaxed problem is as follows:

$$P(\alpha,\beta): \text{Min} \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\sum_{k\in K_{g(i)}}\sum_{l\in K_{g(j)}} p_{ij}d_{kl}w_{ijkl} + \sum_{i\in I}\sum_{k\in K_{g(i)}}(e_i+f_i)ed_k x_{ik} \tag{4.1}$$
$$+\sum_{i\in I\setminus\{n\}}\sum_{j\in I, i<j}\sum_{k\in K_{g(i)}}\alpha_{ijk}(x_{ik}-\sum_{l\in K_{(g_j)}} w_{ijkl})$$
$$+\sum_{i\in I}\sum_{j\in I\setminus\{n\}, j<i}\sum_{k\in K_{g(i)}}\beta_{jik}(x_{ik}-\sum_{l\in K_{(g_j)}} w_{jilk})$$
$$\text{s.t. } (3.2)-(3.8).$$

The relaxed problem can be divided into two subproblems, SP1 and SP2, as follows:

$$SP1(\alpha,\beta): \text{Min} \sum_{i\in I}\sum_{k\in K_{g(i)}}[(e_i+f_i)ed_k + \sum_{j\in I, j>i}\alpha_{ijk} + \sum_{j\in I, j<i}\beta_{jik}]x_{ik} \tag{4.2}$$
$$\text{s.t. } (3.2)-(3.4), (3.7).$$

Using a state-of-the-art commercial solver, SP1 can be solved to optimality quite fast. The solution of SP1(α,β) can be used to find feasible solutions (i.e., upper bounds) to

the AGAP. Note that the solution of SP1 is feasible to AGAP due to (3.2)-(3.4) and (3.7). Thus, knowing that we have a feasible solution defined by x variables, when the true objective function (3.1) where $w_{ijkl} = x_{ik}x_{jl}$ is computed, we obtain an upper bound for the AGAP.

$$SP2(\alpha,\beta) : \text{Min} \quad \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\sum_{k\in K_{g(i)}}\sum_{l\in K_{g(j)}}[p_{ij}d_{kl} - \alpha_{ijk} - \beta_{ijl}]w_{ijkl} \tag{4.3}$$

s.t. (3.8)

$$\sum_{k\in K_{g(i)}}\sum_{l\in K_{(g_j)}} w_{ijkl} = 1 \qquad \forall i \in I \setminus \{n\}, j \in I, i < j. \tag{4.4}$$

SP2(α,β) can easily be solved for each aircraft pair (i,j). Note that constraints (4.4) are redundant for the original problem but are useful in Lagrangian Relaxation to obtain better bounds. For each (i,j) pair:

$w_{ijwt} = 1$ for w,t such that $p_{ijwt} = \min\{\ p_{ij}d_{kl} - \alpha_{ijk} - \beta_{ijl} | k,l \in K\}$

$w_{ijwt} = 0$ for $k,l \in K \setminus \{w,t\}$

The sum of the objective functions of SP1 and SP2(α,β) gives a valid lower bound for the AGAP. In order to find the best lower bound, the following Lagrangian Dual should be solved.

$$Z_D = \text{Max}_{\alpha,\beta}(SP1(\alpha,\beta) + SP2(\alpha,\beta)) \tag{4.5}$$

We solve (4.5) via subgradient optimization and bundle methods, which will be explained in the following chapters.

Chapter 5

Solving the Lagrangian Dual

This chapter discusses two approaches for solving the Lagrangian Dual. The first approach is subgradient optimization, and the second approach is the bundle method.

Let $\mathcal{X}$ and $\mathcal{W}$ be the sets of points satisfying (3.2)-(3.4),(3.7) and (3.8),(4.4), respectively. Note that $\mathcal{X}$ and $\mathcal{W}$ contain finite but very large number of points $\{x^1, ..., x^{S_1}\}$ and $\{w^1, ..., w^{S_2}\}$.

$$Z_D = max_{\alpha,\beta}(SP1(\alpha,\beta) + SP2(\alpha,\beta)) \tag{5.1}$$

$$\begin{aligned} =& max_{\alpha,\beta}(min_{x\in\mathcal{X}}[\sum_{i\in I}\sum_{k\in K_{g(i)}}[(e_i+f_i)ed_k + \sum_{j\in I,j>i}\alpha_{ijk} + \sum_{j\in I,j<i}\beta_{jik}]x_{ik}] \\ &+ min_{w\in\mathcal{W}}[\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\sum_{k\in K_{g(i)}}\sum_{l\in K_{g(j)}}[p_{ij}d_{kl} - \alpha_{ijk} - \beta_{ijl}]w_{ijkl}]) \end{aligned} \tag{5.2}$$

$$\begin{aligned} =& max_{\alpha,\beta}(min_{s=1,...,S_1}[\sum_{i\in I}\sum_{k\in K_{g(i)}}[(e_i+f_i)ed_k + \sum_{j\in I,j>i}\alpha_{ijk} + \sum_{j\in I,j<i}\beta_{jik}]x^s_{ik}] \\ &+ min_{s=1,...,S_2}[\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\sum_{k\in K_{g(i)}}\sum_{l\in K_{g(j)}}[p_{ij}d_{kl} - \alpha_{ijk} - \beta_{ijl}]w^s_{ijkl}]) \end{aligned} \tag{5.3}$$

$$=\text{Max } \eta_1 + \eta_2 \tag{5.4}$$

$$\text{s.t. } \eta_1 \leq \sum_{i \in I} \sum_{k \in K_{g(i)}} [(e_i + f_i)ed_k + \sum_{j \in I, j>i} \alpha_{ijk} + \sum_{j \in I, j<i} \beta_{jik}]x^s_{ik} \quad s = 1, ..., S_1 \tag{5.5}$$

$$\eta_2 \leq \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sum_{k \in K_{g(i)}} \sum_{l \in K_{g(j)}} [p_{ij}d_{kl} - \alpha_{ijk} - \beta_{ijl}]w^s_{ijkl} \quad s = 1, ..., S_2 \tag{5.6}$$

$$\alpha_{ijk} : \ urs \qquad \forall i \in I \setminus \{n\}, j \in I, i < j, k \in K_{g(i)} \tag{5.7}$$

$$\beta_{jik} : \ urs \qquad \forall i \in I, j \in I \setminus \{n\}, j < i, k \in K_{g(i)} \tag{5.8}$$

$$\eta_1 : \ urs \qquad \eta_2 : \ urs \tag{5.9}$$

Large number of constraints are required to be generated as there are a large number of points. Therefore a cutting plane algorithm or a heuristic is required. Subgradient Algorithm is described in Chapter 5.1 and the Bundle Method is described in Chapter 5.2.

5.1 Subgradient Optimization

We provide the subgradient optimization algorithm in Algorithm 2. Table 5.1 shows the notation used throughout the algorithm. Steps of the algorithm are explained as follows.

Table 5.1: Subgradient Algorithm Notation

Parameters	Definitions
z_D	Best lowerbound obtained so far
UB^*	Best upperbound obtained so far
L	Lowerbound obtained at the current iteration
ϵ	Threshold value that when the gap between UB^* and z_D becomes less than ϵ the algorithm stops
s^r	Stepsize at iteration r

We first initialize the values of Lagrange multipliers α and β (Line 2). As stated before, the solution of $SP1(\alpha, \beta)$ can be used to find upper bounds to the AGAP. In order to obtain a good upper bound at the first iteration, initial values of Lagrange multipliers are set to $\alpha^1_{ijk} = (\sum_{l \in K} d_{kl}/|K|)p_{ij}$ and $\beta^1_{jik} = (\sum_{l \in K} d_{kl}/|K|)p_{ji}$. We then initialize the iteration counter (r) as 1, the best lower (z_D) and upper bounds (UB^*) as $-\infty$ and ∞, respectively.

At the beginning of each iteration the stopping conditions are checked and the algorithm stops when at least one of them is satisfied. We use three stopping conditions for the subgradient algorithm: *i)* time limit may be reached *ii)* improvement in a certain number of iterations may be below a threshold value (For instance algorithm stops when the improvement on the lower bound is below a threshold value after a predetermined number of iterations) *iii)* difference between bounds may be below a threshold value (ϵ) (lines 22- 23).

If the stopping conditions are not satisfied, sub-problems using the current Lagrange multipliers are solved, and the corresponding lower-bound value (L) is found (line 8). We then calculate an upper bound using the solution of sub-problem 1, and update the best upper-bound value obtained so far UB^* if a better bound is obtained (lines 7-8).

The Lagrangian bound found in the current iteration (L) is compared to the best Lagrangian bound found so far (z_D) and z_D is updated accordingly. We also keep track of whether there is no improvement at the lower-bound or when the improvement is below a threshold value (when there is not a significant improvement) (lines 9-15). These are related to one of the stopping conditions.

Finally the step size s^r is computed (line 28) using the following formula:

$$s^r = \frac{\lambda(UB^* - L)}{\|\gamma^r\|^2}$$

where UB^* is the best upper bound obtained, L is the lower bound obtained in the rth iteration and $\gamma^r = (\gamma^{\alpha^r}, \gamma^{\beta^r})$. Specifically, $\gamma^{\alpha^r}_{ijk} = x^r_{ik} - \sum_{l \in K_{(g_j)}} w^r_{ijkl}$ and $\gamma^{\beta^r}_{jik} = x^r_{ik} - \sum_{l \in K_{(g_j)}} w^r_{jilk}$ $\forall i, j, k$, and λ is a constant. λ is usually set to 2 initially and halved whenever z_D does not improve for a certain number of consecutive iterations. In our algorithm, if there is not a significant improvement for 25 iterations λ is divided by two (lines 25-27).

κ^{α^r} (κ^{β^r}) is the direction used for updating α (resp. β) in iteration r and is obtained using the following formula:

$$\kappa^{\alpha^r} = \gamma^{\alpha^r} \tag{5.10}$$

$$\kappa^{\beta^r} = \gamma^{\beta^r} \tag{5.11}$$

Finally, the values of Lagrange multipliers to be used in the next iteration $(r+1)$ are calculated as follows (line 25):

$$\alpha^{r+1} = \alpha^r + s^r \kappa^{\alpha^r} \tag{5.12}$$

$$\beta^{r+1} = \beta^r + s^r \kappa^{\beta^r} \tag{5.13}$$

Algorithm 2: Subgradient Optimization

1 Let $SP_1^*(\alpha,\beta), SP_2^*(\alpha,\beta)$ be the optimal values of SP1 and SP2 respectively with given α & β and $argmax\{SP_1(\alpha,\beta)\}$ be the optimal solution of SP1 with the given α & β.
2 Initialize α and β
3 Set input parameters λ, *noImp*, ρ, θ
4 r $\leftarrow$ 1 , $z_D \leftarrow$ -∞, $UB^* \leftarrow \infty$, $\epsilon \leftarrow 1$, $Time \leftarrow 0$, $c \leftarrow 0$
5 **while** *True* **do**
6 **if** *Time* $\geq$ *3600* **then**
7 False
8 $L \leftarrow$SP1$^*(\alpha,\beta)$+ SP2$^*(\alpha,\beta)$
9 $x^* \leftarrow$ argmaxSP1(α,β)
10 **if** $UB^* \geq$ $\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\sum_{k\in K_{g(i)}}\sum_{l\in K_{g(j)}} p_{ij}d_{kl}x_{ik}^*x_{jl}^* + \sum_{i\in I}\sum_{k\in K_{g(i)}}(e_i+f_i)ed_kx_{ik}^*$ **then**
11 $UB^* \leftarrow$ $\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\sum_{k\in K_{g(i)}}\sum_{l\in K_{g(j)}} p_{ij}d_{kl}x_{ik}^*x_{jl}^* + \sum_{i\in I}\sum_{k\in K_{g(i)}}(e_i+f_i)ed_kx_{ik}^*$
12 **if** $L > z_D$ **then**
13 **if** $L > z_D + \rho$ **then**
14 c $\leftarrow$ 0
15 **else**
16 c $\leftarrow$ c + 1
17 $z_D \leftarrow L$
18 $noImp \leftarrow 0$
19 **else**
20 $noImp \leftarrow noImp + 1$
21 c $\leftarrow$ c + 1
22 **if** $UB^* - z_D \leq \epsilon$ *or* $c = \theta$ **then**
23 False
24 **if** *noImp = 25* **then**
25 $\lambda \leftarrow \lambda/2$
26 $noImp \leftarrow 0$
27 Calculate s^r, α & β

Several variants of the subgradient algorithm discussed above are examined in this study. The first variant is taking the Lagrangian multipliers nonnegative. Note that constraints (3.5) and (3.6) can be taken as greater than or equal to constraints, resulting in nonnegative Lagrangian multipliers. We call this variant SGO.

Crainic et al. suggest that using the direction of the previous iteration may result in performance improvements [34]. Therefore, they suggest multiplying the previous direction with a factor δ and adding this term to the subgradient. Their direction computation can be found in equations (5.14) and (5.15). We call this variant SGA.

$$\kappa^{\alpha^r} = \gamma^{\alpha^r} + \delta\kappa^{\alpha^{r-1}} \tag{5.14}$$

$$\kappa^{\beta^r} = \gamma^{\beta^r} + \delta\kappa^{\beta^{r-1}} \tag{5.15}$$

Step size (s^r) decreases as the gap between bounds decreases, which may slow the convergence. To prevent this, Beasley [41] suggests multiplying the upper bound with a constant when computing the step size. We call this variant SGB.

5.2 Bundle Method

The Bundle method attemps to solve the Lagrangian dual through a cutting plane approach. The method is based on solving a cutting plane model to generate new Lagrange multipliers and adding a new cut utilizing them. Therefore, unlike subgradient algorithm Bundle method keeps the past information in a Bundle. Bundle methods can be classified as aggregated and dis-aggregated bundle methods. Aggregated bundle methods use a single cut whereas dis-aggregated Bundle methods add a separate cut for each subproblem. Although dis-aggregated methods result with tighter constraints that may increase the convergence, aggregated methods are more memory efficient and have less computation time [36]. Hence, instead of adding two cuts, we add a single cut at each iteration by taking the sum of constraints (5.5) and (5.6).

Let B be the bundle, which is a subset of the previously found subgradients and the corresponding lowerbounds. The following model can be solved to generate new Lagrange multipliers ($\bar{\alpha}$ & $\bar{\beta}$). Solving $SP1(\bar{\alpha}, \bar{\beta})$ and $SP2(\bar{\alpha}, \bar{\beta})$ creates new cuts for the

Lagrangian Dual model, and the algorithm continues until convergence or until the time limit is reached. Table 5.2 shows the notation used throughout the algorithm.

Table 5.2: Bundle Method Notation

Parameters	Definitions
L	Lowerbound obtained with Lagrange multipliers α & β
L^s	Lowerbound obtained with Lagrange multipliers α^s & β^s
$\bar{L}$	Lowerbound obtained with Lagrange multipliers $\bar{\alpha}$ & $\bar{\beta}$

The problem can be written as follows:

MODEL SC-1

$$\text{Max} \quad \eta \tag{5.16}$$

$$\text{s.t. } \eta \leq L^s + \gamma^{\alpha^s}(\bar{\alpha} - \alpha^s) + \gamma^{\beta^s}(\bar{\beta} - \beta^s) \quad \forall s \in B \tag{5.17}$$

$$\bar{\alpha}_{ijk}: \ urs \qquad \forall i \in I \setminus \{n\}, j \in I, i < j, k \in K_{g(i)} \tag{5.18}$$

$$\bar{\beta}_{jik}: \ urs \qquad \forall i \in I, j \in I \setminus \{n\}, j < i, k \in K_{g(i)} \tag{5.19}$$

$$\eta: \ urs \tag{5.20}$$

Let α and β be the Lagrangian multipliers at the current iteration. Using $\bar{\alpha} = \alpha + \tau \times \kappa^{\alpha}$ and $\bar{\beta} = \beta + \tau \times \kappa^{\beta}$, Model SC-1 can be written equivalently as follows:

MODEL SC-2

$$\text{Max} \quad \eta \tag{5.21}$$

$$\text{s.t. } \eta \leq L^s + \gamma^{\alpha^s}(\alpha + \tau \times \kappa^{\alpha} - \alpha^s) + \gamma^{\beta^s}(\beta + \tau \times \kappa^{\beta} - \beta^s) \quad \forall s \in B \tag{5.22}$$

$$\kappa^{\alpha}_{ijk}: \ urs \qquad \forall i \in I \setminus \{n\}, j \in I, i < j, k \in K_{g(i)} \tag{5.23}$$

$$\kappa^{\beta}_{jik}: \ urs \qquad \forall i \in I, j \in I \setminus \{n\}, j < i, k \in K_{g(i)} \tag{5.24}$$

(5.20)

Note that the first degree linear approximation of L using α^s & β^s is as follows:

$$L^s + \gamma^{\alpha^s}(\alpha - \alpha^s) + \gamma^{\beta^s}(\beta - \beta^s)$$

Hence the difference between L and its approximation is:

$$\omega^s = L^s + \gamma^{\alpha^s}(\alpha - \alpha^s) + \gamma^{\beta^s}(\beta - \beta^s) - L \quad \forall s \in B$$

,which is referred to as *linearization error*. η is an upperbound on $\bar{L}$. Therefore, $\upsilon = \eta - L$ is referred as the *predicted increase*. Using ω^s and υ, we can re-write Model SC-2 as follows:

$$\text{Max} \quad \upsilon + L \tag{5.25}$$

$$\text{s.t.} \quad \upsilon \leq \omega^s + \tau\gamma^{\alpha^s}\kappa^{\alpha} + \tau\gamma^{\beta^s}\kappa^{\beta} \quad \forall s \in B \tag{5.26}$$

$$(5.20), (5.23) - (5.24)$$

As L is a constant, it can be ignored, which leads to the following LP:

SC-P

$$\text{Max} \quad \upsilon \tag{5.27}$$

$$\text{s.t.} \quad (5.20), (5.23) - (5.24), (5.26)$$

If we take the dual of this LP :

SC-D

$$\text{Min} \quad \sum_{s \in B} \theta^s \omega_s \tag{5.28}$$

$$\text{s.t.} \quad \sum_{s \in B} \theta^s = 1 \tag{5.29}$$

$$\sum_{s \in B} \theta^s \gamma_{ijk}^{\alpha^s} = 0 \quad \forall i \in I \setminus \{n\}, j \in I, i < j, k \in K_{g(i)} \tag{5.30}$$

$$\sum_{s \in B} \theta^s \gamma_{jik}^{\beta^s} = 0 \quad \forall i \in I, j \in I \setminus \{n\}, j < i, k \in K_{g(i)} \tag{5.31}$$

$$\theta^s \geq 0 \quad \forall s \in B \tag{5.32}$$

As mentioned in Chapter 2 the primal problem can be unbounded, which results in dual infeasibility. The Bundle Algorithm deals with this issue by adding a "stabilization

term" in the primal problem. From the dual viewpoint, this can be considered as relaxing constraints (5.30) & (5.31) in a Lagrangian manner.

In order to make the primal problem bounded the "stabilization term" $\frac{-1}{2\tau}(\|\tau\kappa^{\alpha}\|^2 + \|\tau\kappa^{\beta}\|^2)$ is added to the objective function. τ is referred as the trust region parameter in the literature as it effects the punishment weight of the stepsize in the objective function. In later steps of the algorithm, as a result of having more information, we have more "trust" in our model. Therefore, in general τ is increased as the algorithm proceeds. SC-P with the stabilization term in the objective function is the following QP:

SC-QP

$$\text{Max} \quad \upsilon - \frac{\tau}{2}(\|\kappa^{\alpha}\|^2 + \|\kappa^{\beta}\|^2) \tag{5.33}$$

$$\text{s.t.} \ (5.20), (5.23) - (5.24), (5.26) \tag{5.34}$$

The dual of this QP is as follows:

SC-DQP

$$\text{Min} \quad \sum_{s\in B}\theta^s\omega^s + \frac{\tau}{2}\|\sum_{s\in B}\theta^s\gamma^{\alpha^s}\|^2 + \frac{\tau}{2}\|\sum_{s\in B}\theta^s\gamma^{\beta^s}\|^2 \tag{5.35}$$

$$\text{s.t.} \quad \sum_{s\in B}\theta^s = 1 \tag{5.36}$$

$$\theta^s \geq 0 \quad \forall s \in B \tag{5.37}$$

When we divide the objective function by τ, we obtain the Bundle Model used in the literature [32, 30, 35] is obtained, which is as follows:

Bundle Model

$$\text{Min} \quad \frac{1}{\tau}\sum_{s\in B}\theta^s\omega^s + \frac{1}{2}\|\sum_{s\in B}\theta^s\gamma^{\alpha^s}\|^2 + \frac{1}{2}\|\sum_{s\in B}\theta^s\gamma^{\beta^s}\|^2 \tag{5.38}$$

$$\text{s.t.} \ (5.36) - (5.37) \tag{5.39}$$

At each iteration of the bundle method, the bundle model is solved to obtain new cuts. However, as the problem size, i.e. the bundle size, increases; the time required to solve this

model increases. One way to address this issue is using a *B-Strategy* [34, 42]. B-Strategy is based on removing bundle members that have been inactive for a predetermined number of successive iterations (*Inactive Count*) [34]. We measure inactivity by comparing the dual prices of the cuts with our tolerance. When the dual price of a cut is less than the tolerance we consider it as 0 and take the corresponding cut as inactive.

From the derivation of the Quadratic Dual:

$$\kappa^{\alpha} = \sum_{s \in B} \theta^s \gamma^{\alpha^s} \quad \& \quad \kappa^{\beta} = \sum_{s \in B} \theta^s \gamma^{\beta^s}$$

Then, $\omega = \overline{L} - \gamma^{\bar{\alpha}}(\kappa^{\alpha}\tau) - \gamma^{\bar{\beta}}(\kappa^{\beta}\tau) - L$ is the linearization error obtained when estimating $\overline{L}$ using α and β. When the errors are aggregated over all bundle members, the aggregated linearization error is found: $\sigma = \sum_{s \in B} \omega^s \theta^s$.

The pseudo-code of the Bundle Algorithm is provided in Algorithm 3. We first initialize the parameters (lines 1-2). Then we create the initial bundle members (line 5). To create the initial bundle, we apply the subgradient algorithm with initial multipliers set as provided in Chapter 5.1 for 4 iterations and we set all Lagrange multipliers as 0 in the fifth iteration. Then we use the B-Strategy, to remove bundle members that remained inactive for a number of successive iterations. If any change is made in the bundle, we update ω_B, the set of linearization errors of the bundle members (lines 7-8). Then, we solve the Bundle model (line 9).

Using the solution of the Bundle model, new directions κ^{α}, κ^{β} and predicted increase υ are calculated (lines 10-11). Then, tentative Lagrange multipliers and aggregated linearization error are calculated (lines 12-14). The lower and upper bounds obtained by these multipliers are computed. If the upperbound is higher than the best upperbound known (UB^*), UB^* is updated. Subgradients are computed and new information is added in the Bundle B (line 21).

Linearization error corresponding to the lowerbound obtained by new multipliers (ω) is computed. If the obtained increase in the lower bound ($\overline{L} - L$) is significantly higher than predicted increase (υ), a *Serious Step* ,a move to the new point, is made (lines 23-25). Otherwise, a *null step* is made, information is added to the bundle but a move is not made. The trust region parameter is updated according to the τ strategy suggested in [30]. z_D is updated if the new lowerbound is higher.

Algorithm 3: Bundle Algorithm Pseudo-code

Let $SP_1^*(\alpha,\beta), SP_2^*(\alpha,\beta)$ be the optimal values of SP1 and SP2 respectively with given α & β and $argmax\{SP_1(\alpha,\beta)\}$ be the optimal solution of SP1 with the given α & β. Initialize α and β

$r \leftarrow 1$, $z_D \leftarrow$ -∞, $\lambda \leftarrow 2$, $UB^* \leftarrow \infty$, $Time \leftarrow 0$, $t_0 \leftarrow 0.5$

while $Time \leq 3600$ **do**

  if $r \leq 5$ **then**

    Create Initial Bundle

  else

    B-Strategy

    Update ω_B

    Solve the Bundle Model

    Calculate κ^α and κ^β

    Calculate υ

    $\bar{\alpha} \leftarrow \alpha + \tau \times \kappa^\alpha$

    $\bar{\beta} \leftarrow \beta + \tau \times \kappa^\beta$

    Calculate $\sigma = \omega_B \theta$

    solve SP1($\bar{\alpha}, \bar{\beta}$) and solve SP2($\bar{\alpha}, \bar{\beta}$)

    $\overline{L} \leftarrow SP_1^*(\bar{\alpha}, \bar{\beta}) + SP_2^*(\bar{\alpha}, \bar{\beta})$

    $x^* \leftarrow$ argmaxSP1($\bar{\alpha}$,$\bar{\beta}$)

    if $UB^* \geq \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\sum_{k\in K_{g(i)}}\sum_{l\in K_{g(j)}} p_{ij}d_{kl}x_{ik}^*x_{jl}^* + \sum_{i\in I}\sum_{k\in K_{g(i)}}(e_i+f_i)ed_kx_{ik}^*$ **then**

      $UB^* \leftarrow \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\sum_{k\in K_{g(i)}}\sum_{l\in K_{g(j)}} p_{ij}d_{kl}x_{ik}^*x_{jl}^* + \sum_{i\in I}\sum_{k\in K_{g(i)}}(e_i+f_i)ed_kx_{ik}^*$

    Calculate $\gamma^{\bar{\alpha}}$ and $\gamma^{\bar{\beta}}$.

    Add $\overline{L}, \gamma^{\bar{\alpha}}$ and $\gamma^{\bar{\beta}}$ to B.

    Calculate ω.

    if $\overline{L} - L \geq m_1 \times \upsilon$ **then**

      $\alpha \leftarrow \bar{\alpha}, \quad \beta \leftarrow \bar{\beta} \quad L \leftarrow \overline{L} \quad \gamma^\alpha \leftarrow \gamma^{\bar{\alpha}}, \quad \gamma^\beta \leftarrow \gamma^{\bar{\beta}}$

      $\tau \leftarrow max(min(t_M, M \times \tau, 2 \times \tau \times \upsilon \times (\upsilon - \overline{L} + L)), \tau)$

    else

      if $\omega > m_2 \times \sigma$ **then**

        $\tau \leftarrow min(max(t_m, m \times \tau, (\omega + \overline{L} - L)/(2 \times \omega)), \tau)$

  $z_D \leftarrow max(z_D, \overline{L})$

  if $UB^* - z_D \leq \epsilon$ **then**

    False

  $r \leftarrow r + 1$

Chapter 6

Computational Results

In this chapter we provide the results of our computational experiments. We first provide information on the benchmark instances used and then provide the results of the preliminary experiments, which are used for parameter tuning in the subgradient and bundle methods. We then compare the algorithms with state-of-the-art MIP and MIQP solvers.

Experiments were performed on a laptop with 64-bit Operating System, x64-based Intel(R) Core(TM) i7-9750H processor, CPU @ 2.60GHz to 2592 Mhz, and with installed memory (RAM) equal to 8.00 GB. SP1, MILP model and LP relaxation are solved by CPLEX (version 20.1.0), QP model is solved by Gurobi (version 9.5.1).

6.1 Data

In this chapter, we describe the data we used for our computational experiments. We use the distance data of three different airports: Esenboğa (ESB), Atatürk (ATA), and Incheon (ICN). These three airports correspond to problem instances of various size: Esenboğa airport has 18 gates (9 domestic, 9 international), Atatürk airport has 38 gates (12 domestic, 26 international), Incheon airport only provides international flights and has 74 gates. Distances for the Esenboğa Airport are as used in [14], generated based on the airport's layout, which is provided in Appendix 8.1. Distances of the Atatürk Airport are based on Ersan [43] and finally, data for Incheon Airport are taken from Cheng et al. [16]. Layout of Atatürk Airport can be found in Appendix 8.2, gates 101-112 and 201-226 are considered in our experiments as in [14].

In Esenboğa and Atatürk instances, we use the flight data used in Karsu et al. [14]. They use two sets of data to represent different levels of apron usage. Data regarding the flights are generated by the discrete uniform distribution (DU) and distribution parameters can be found in Table 6.1. In this table a_i and d_i denote the arrival and departure times of aircraft i, respectively. Set1 instances have fewer apron requirements as Set1 flights have less duration on the ground, whereas Set2 flights have more ground time, resulting in more apron assignments.

For each airport (hence the number of gates m), they generate instance data for $n = 50, 100$ & 200. For each (m, n) combination, 10 instances are generated. Thus, we have 60 ESB and 60 ATA instances in total.

Table 6.1: Flight-related parameters for Esenboğa and Atatürk Instances

Set	Parameters	Distribution
Set1 & Set2	p_{ij}	DU(0,200/n)
Set1	a_i	DU(0,300)
	d_i	30 + DU(0,30) + a_i
Set2	a_i	DU(0,150)
	d_i	60 + DU(0,60) + a_i

Incheon Airport instances are constructed for Terminal 1 which has 74 gates and 14 exits. The daily flight data of these instances is obtained from FlightStats by Li et al. [22]. In these instances, the number of daily flights change between 279 and 304 [15]. Three levels are used for creating the percentage of transfer passengers (π), which are 0.1, 0.3, and 0.5, respectively. However, we do not have the data for Thursday 0.1 instances and we work with 20 ICN instances as in [22, 15].

6.2 Preliminary experiments

In this section we present our parameter tuning experiments, starting with the Subgradient Algorithm.

6.2.1 Subgradient Algorithm Parameters

Recall that the algorithm terminates when one of the three stopping conditions is reached. Different parameter combinations are tested in order to find the most suitable parameters (ρ and θ) for the stopping condition. We also checked the performance of three strategies for setting the λ parameter: λ always decreases (1), λ becomes 1 at every 100 iterations (2), λ becomes 2 at every 100 iterations (3).

We used the smallest ESB instances for the preliminary experiments, which are with 50 aircraft and 18 gates. Table 6.2 shows the results of the experiments. In the table we report the average and maximum CPU Times and GAP(%) results for each parameter combination over the 10 instances of each set. The GAP(%) is calculated as follows:

$$GAP(\%) = \frac{(SG_{UB} - SG_{LB})}{SG_{UB}} \times 100$$

, where SG_{UB} and SG_{LB} are the best upper and lower bound obtained by the subgradient algorithm, respectively. In all of these experiments termination was due to the slow improvement in the lower-bound.

It can be seen that λ strategy 1 has the smallest gaps when other parameters (ρ, θ) are fixed. Figure 6.1 depicts the performance of alternative λ strategies on an example instance (ESB, Set 1, instance 1). It is seen that λ strategy 1 has a smoother curve due to the monotone decrease in the step size. λ strategy 2 and 3 are based on increasing the stepsize every 100 iterations. As the subgradient algorithm is based on local improvement, increasing the stepsize can be considered as changing the neighbourhood. However, such strategies did not prove to be effective in our preliminary experiments. Results for the computational experiment can be found in Appendix 8.3.

When λ strategy 1 is used, ($\rho = 5$, $\theta = 1000$), ($\rho = 5$, $\theta = 2000$),($\rho = 10$, $\theta = 1000$),($\rho = 10$, $\theta = 2000$),($\rho = 25$, $\theta = 2000$) have the lowest gap results for both sets. Among them ($\rho = 5$, $\theta = 1000$) and ($\rho = 10$, $\theta = 1000$) use the least CPU time. Based on these, we decided to set the parameters as: (λ $strategy = 1$, $\rho = 5$, $\theta = 1000$).

Tables 6.4 and 6.5 summarize the results of alternative subgradient algorithm approaches discussed in Chapter 5.1, which are also shown in Table 6.3.

Table 6.2: Preliminary Experiments for parameters of the Subgradient Algorithm

			SET 1				SET 2			
			CPU Time		GAP (%)		CPU Time		GAP (%)	
λ strategy	ρ	θ	Max	Avg	Max	Avg	Max	Avg	Max	Avg
1	5	500	56	32.1	3.05	1.47	61	35.7	4.22	1.81
		1000	**102**	**54.2**	**3.04**	**1.46**	**83**	**54.3**	**4.22**	**1.81**
		2000	286	107.3	3.04	1.46	124	81.3	4.22	1.81
	10	500	49	31	3.05	1.48	57	33.1	4.22	1.81
		1000	98	54.4	3.04	1.46	79	48.6	4.22	1.81
		2000	200	92.8	3.04	1.46	121	78.1	4.22	1.81
	25	500	38	25.3	3.05	1.48	56	31.4	4.22	1.82
		1000	83	46.6	3.04	1.47	78	46	4.22	1.81
		2000	185	87.9	3.04	1.46	120	75.2	4.22	1.81
2	5	500	50	30.6	3.12	1.51	54	33.1	4.36	1.91
		1000	69	44.1	3.07	1.49	106	62	4.35	1.85
		2000	99	66.3	3.05	1.46	212	114.4	4.33	1.82
	10	500	40	25.2	3.12	1.52	53	30.2	4.36	1.91
		1000	58	37.4	3.07	1.49	74	50.5	4.36	1.86
		2000	89	60	3.05	1.47	188	89.5	4.35	1.82
	25	500	35	21.1	3.12	1.54	52	29.5	4.36	1.91
		1000	53	33.1	3.07	1.50	73	44.8	4.36	1.87
		2000	85	58.5	3.05	1.47	115	74	4.35	1.83
3	5	500	37	28	3.42	1.82	91	49.5	4.99	2.27
		1000	59	41.9	3.42	1.75	106	78	4.84	2.20
		2000	135	78	3.39	1.69	209	148.8	4.79	2.14
	10	500	33	21.1	3.55	1.85	66	41.2	4.99	2.27
		1000	59	37.9	3.42	1.77	87	62.8	4.84	2.22
		2000	91	60.6	3.39	1.73	155	107.3	4.83	2.17
	25	500	29	16.5	3.55	1.87	59	29.6	4.99	2.31
		1000	51	28.1	3.42	1.82	78	45.3	4.84	2.26
		2000	74	48.9	3.42	1.75	118	74	4.84	2.21

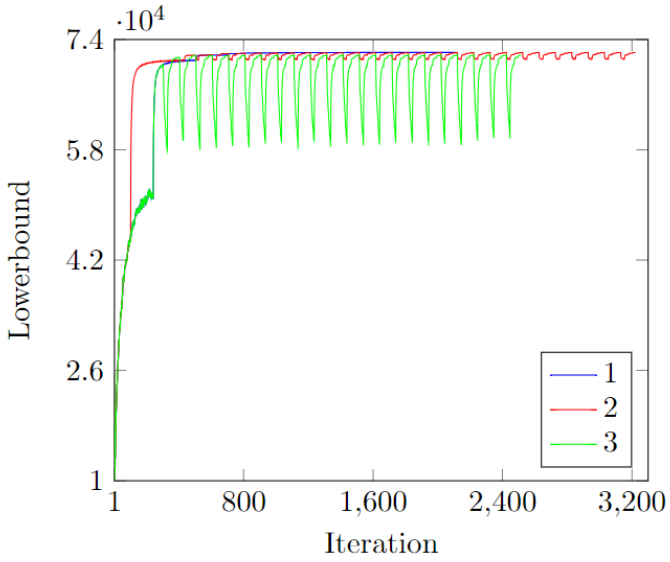

Figure 6.1: Performance of alternative λ setting strategies ($\rho = 5, \theta = 1000$)

Table 6.3: SG Variants

Variant	Definitions
SGA	Uses the previous direction when computing the current direction
SGB	Multiplies UB^* with a constant when computing s^r
SGO	Forces Lagrange multipliers to be greater than or equal to 0

In these experiments, we used the smallest-sized instances of the ESB and ATA datasets. We report the average and maximum gap values over the 10 instances. We used $\delta = 0.01$ for SGA experiments and we multiplied UB^* with 1.05 in SGB experiments. It is seen that all approaches have similar performance. SGO slightly outperforms the others in terms of the average gap in ESB Set1 instances whereas SGA provides the smallest maximum gap. For ESB Set2 instances, however, SGB provided the the least average gap, whereas the least maximum gap is obtained in both SGO and SGB. In ATA instances SGB outperforms other subgradient approaches in both sets in terms of average and maximum gaps. As the gaps were very close to each other, we conclude that these approaches are comparable. Therefore, in our main experiments we implement the main variant SG. Details for the results can be found in Appendix 8.4.

Table 6.4: Gap performance of subgradient algorithm variants (ESB, $n = 50$)

	SG (Gap (%))		SGB (Gap (%))		SGA (Gap (%))		SGO (Gap (%))	
	Avg	Max	Avg	Max	Avg	Max	Avg	Max
Set1	1.46	3.04	1.47	3.06	1.46	2.94	1.43	3.06
Set2	1.81	4.22	1.78	4.19	1.81	4.20	1.81	4.19

Table 6.5: Gap Performance of subgradient algorithm variants (ATA, $n = 50$)

	SG (Gap (%))		SGB (Gap (%))		SGA (Gap (%))		SGO (Gap (%))	
	Avg	Max	Avg	Max	Avg	Max	Avg	Max
Set1	10.58	13.00	10.56	12.96	10.57	13.03	10.58	13.03
Set2	15.34	17.41	15.33	17.31	15.35	17.42	15.34	17.40

6.2.2 Bundle Method Parameters

We also performed experiments to set the parameter values used in the Bundle method. As this method requires more computational effort, we set the time limit to 10 minutes in these experiments.

Recall that, the bundle method relies on solving quadratic programming problems to determine the new multipliers at each iteration, which takes more time as the the bundle size increases. To alleviate the increasing computational burden, we use the so called B-Strategy and remove the bundle members that have been inactive for a number of successive periods, which is referred to as the inactive count (being inactive means the corresponding cut has a dual price that is below a tolerance). In our preliminary experiments for B-Strategy, we try three different levels for the tolerance and the inactive count. The rest of the parameters (m_2 to t_M) are fixed as given in Table 6.6. We used two benchmark instances with $n = 50$ from sets 1 and 2 of the ESB dataset. The results can be seen in Table 6.7. Setting the tolerance as 10^{-3} and removing members that have been inactive for 2 iterations gave the best results for both benchmark istances. Therefore, we fixed the tolerance = 10^{-3} and InactiveCount = 2 in further experiments.

Table 6.6: Parameters of Bundle Method

m_2	t_m	m	t_0	m_1	M	t_M
3	0.1	0.1	0.1	0.05	1.05	7

Table 6.7: Bundle B-Strategy Trials (10 minutes)

Tolerance	Inactive Count	set1 ins1 LB	set2 ins1 LB
10^{-2}	1	23569.6	71926.6
	2	23602.4	71997.1
	3	23630.2	72016.7
	5	23633	72026.7
10^{-3}	1	23645.7	72027.9
	2	**23649.4**	**72037.4**
	3	23641.4	72028.5
	5	23640.3	72016.7
10^{-5}	1	23641.9	72024.3
	2	23640.7	72013.6
	3	23641.6	72018.3
	5	23635.6	71996.3

We then focused on the trust region (τ-strategy) parameters, which are m_2, t_m, m, t_0, m_1, M & t_M. Note that m_1 has to be in the interval [0,1) [34]. Frangioni & Gallo [30] suggest taking $m_1 = 0.1$ while Crainic et al. [34] state that although m_1 is taken as 0.1 in general; $m_1 = 0$ worked better in some of their instances. Based on our initial observations, we used $m_1 = 0.1$ and $m_1 = 0.05$ as m_1 configurations in our preliminary experiments. Crainic et al. [34] set $t_0 = 1$ but state that $t_0 = 0.1$ worked better in one of their sets. We fixed t_0 to 0.1 in our experiments. We also set $m_2 = 3$ in our experiments, based on the suggestions of Frangioni & Gallo and Crainic et al. [30, 34]. Table 6.8 shows the Bundle parameter tuning results for one instance from Set 2. The best lower bound was obtained when $M = 1.05$, $t_M = 7$ & $m_1 = 0.05$. Hence, we used these parameter configurations in our main experiments.

Table 6.8: Bundle Parameter Tuning (10 minute)

		$m_1 = 0.05$	$m_1 = 0.1$
M	t_M	LB	LB
1.03	6	72019.5	71982.4
	7	72021.2	71980.2
	8	72016.4	71976.5
1.05	6	72029	71991.9
	7	**72037.4**	71999
	8	72017.4	72003.3
1.06	6	72033.4	71985.9
	7	72020.9	72001
	8	72031.6	72004.4

6.3 Main experiments

We started out main experiments by comparing the performances of Subgradient and Bundle Methods, which can be seen in Tables 6.9 and 6.10. In these tables we report the average and the maximum gap values for ESB and ATA instances, respectively.

As the Bundle Method is computationally more demanding, we first implemented the Subgradient Algorithm and set a time limit of 10 minutes for the bundle method to have a fair comparison in terms of the solution time. We expected the Bundle method to outperform Subgradient Algorithm since Bundle Method uses the previously obtained information while computing the new direction. However, our computational experiments showed the opposite. The gaps obtained in the Bundle Method are higher than those of the Subgradient Algorithm. This could be caused by the time limit, precluding convergence. As convergence would take significantly long in the Bundle Method, in our further experiments we decided to use the Subgradient Algorithm. Details for the experiments can be found in Appendix 8.5.

Table 6.9: SG vs Bundle Gap Results (ESB)

			SG Gap(%)		Bundle Gap(%)	
	m	n	Avg	Max	Avg	Max
Set1	18	50	1.46	3.04	1.70	3.33
		100	1.77	2.41	3.81	5.07
		200	1.87	2.49	3.81	4.56
Set2	18	50	1.81	4.22	1.90	4.41
		100	1.23	1.84	1.80	2.73
		200	0.55	0.74	1.66	2.04

Table 6.10: SG vs Bundle Gap Results (ATA)

			SG Gap(%)		Bundle Gap(%)	
	m	n	Avg	Max	Avg	Max
Set1	38	50	10.58	13.00	11.04	13.55
		100	13.89	14.43	18.02	18.52
		200	21.59	33.68	36.63	40.85
Set2	38	50	15.34	17.41	18.13	21.51
		100	6.18	8.97	8.43	12.13
		200	2.62	3.03	6.07	7.06

Note that one could also obtain bounds by utilizing the state-of-the-art solvers for the mixed integer programming formulation (MILP) and its LP relaxation. To see the qualities of these bounds and the performance of the IP solver on various problem sizes, we solved the MILP model using CPLEX 20.1.0 in C++. Moreover, as the quadratic formulations are reported to perform better in some other assignment problem variants (see e.g. [44]), we also solved the MIQP formulation, in which the multiplication of the x variables are kept as it is, using Gurobi 9.5.1 in Python for ESB instances where n=50. We used Gurobi for solving the MIQP model as Gurobi was reported to have much better performance than CPLEX in similar quadratic assignment programming settings [44]. 10 instances are solved in each set with a one hour time limit and the results are reported in Table 6.11. It is seen that MIQP model has both larger gaps and higher computation times. Therefore, in our main experiments we compared the performance of the subgradient algorithm with the results of the MILP model.

Table 6.11: CPLEX IP Solver & Gurobi QP Solver Results (ESB, $n = 50$)

	CPLEX				Gurobi			
	CPU Time		Gap(%)		CPU Time		Gap(%)	
	Avg	Max	Avg	Max	Avg	Max	Avg	Max
Set1	324.4	812	0	0	3600	3600	28.10	33.3
Set2	3381.7	3600	1.07	3.44	3600	3600	23.64	30.7

We also compared the suggested MILP model's performance with the three index formulation used in [15]. We analyzed the lowerbounds computed by CPLEX for four-indexed MILP, three-indexed MILP, four-indexed LP Relaxation, and 3-indexed LP Relaxation for ESB (n=50) instances. Results of the experiments for 10 instances from set 1 and set 2 can be found in Tables 6.12 and 6.13, respectively. Computational results indicate that four-indexed formulation provides tighter bounds in both MILP and LP Relaxation models for both sets, as expected.

Table 6.12: Lowerbound Performance of 3 index and 4 index formulations (ESB Set1)

	MILP		LP	
Instance	4-indexed	3-indexed	4-indexed	3-indexed
1	23934	23347.38	23702.4	22571.49
2	26782	26040.45	26509.8	24745.21
3	23333	22572.81	22745	21747.09
4	26453	25657.38	26160	24973.25
5	24122	23563.58	23879	22683.54
6	23354	22869.91	22863.8	21903.93
7	25129	24570.38	24959.5	23494.24
8	24868	24143.82	24516.8	23559.60
9	24615	24031.93	24322.6	23272.80
10	27348	26905.83	27277.7	25726.00

Table 6.13: Lowerbound Performance of 3 index and 4 index formulations (ESB Set2)

	MILP		LP	
Instance	4-indexed	3-indexed	4-indexed	3-indexed
1	72560.4	71035.93	72190.2	68085.59
2	68473	66561.03	67621.1	64123.11
3	77550.8	75837.90	77149.4	72850.70
4	80492.4	79430.30	80145.9	76328.04
5	79465.5	78280.57	79133.7	75031.02
6	75015.4	73903.52	74570.2	70605.19
7	85460.3	83893.98	84734.1	79689.10
8	80836	80041.09	80507	76864.95
9	78221.2	77420.34	77962.2	74123.12
10	68409.5	67382.59	68033.5	64395.83

Finally, we compare the subgradient algorithm (SG) to the MILP and its LP relaxation in terms of the solution times and the tightness of obtained lower and upper bounds. To measure tightness, we compared the upper bound obtained in the subgradient algorithm (SG_{UB}) with upper bound provided by CPLEX for the MILP model (C_{UB}) and our lower bounds (SG_{LB}) with lower bounds provided by the MILP model (C_{LB}) and its linear relaxation (LP) bound. Tightness computations using the MILP solutions are provided below:

$$LB(\%) = \frac{(C_{LB} - SG_{LB})}{SG_{LB}} \times 100$$
$$UB(\%) = \frac{(SG_{UB} - C_{UB})}{SG_{UB}} \times 100$$

That is we report the percentage improvement that the MILP and LP bounds provide over the subgradient algorithm bounds. Results for the main experiments can be found in Tables 6.14 and 6.15 for the ESB and ATA data sets, respectively. The results show that in the majority of instances, the MILP model cannot be solved to optimality within the 1 hour time limit: only 11 instances out of 80 were solved optimally. Those instances are the smallest instances ($n = 50, m = 18$) and SG upper bounds are less than 2.66% away from the optimal values. Also on average, SG CPU Times are 83.31% lower than MILP CPU Times in these instances.

CPLEX was not able to provide any lower bounds for 29 ESB (9 from set1 and 20 from set2) and 16 ATA (10 from set1 and 6 from set2) instances. Hence, in those instances we were only able to compare SG and MILP upper bounds. In these ESB instances, SG upper bounds are 15.38%, 8.62% and 5.77% lower for n=100 set1, n=100 and 200 set2 instances, respectively. The quality of SG upper bounds are much better in ATA instances compared to the MILP upper bounds, which indicates the high potential of the SG in larger problem instances. Specifically, on average, SG upper bounds are 33.49% lower for $n = 50$, 21.09% lower for $n = 100$ instances. Note that when a bound could not be obtained within the time limit, we indicate it as -.

MILP model was not able to provide any bound within the time limit for one ESB ($m = 18, n = 100$) and four ATA ($m = 38, n = 100$) instances. When MILP model cannot provide any bound for all instances in a set-size configuration, number of instances that were solved is reported in parenthesis. For a problem size (m, n combination), if this occurs in at least one of the 10 instances, we do not solve the larger instances which is denoted as ** in the Tables. Therefore, larger problem sizes for those sets are not attempted to be solved by MILP model. Moreover, we could not construct ATA set1 $n = 200$ instances due to memory overflow, which is denoted with the -* symbol in Table 6.15.

For relatively smaller instances, we observe that LP relaxation bounds are tighter than SG lower bounds. However, as the problem size increases SG bounds become significantly better. We expected to have tighter lower bounds in all instances as our problem does not have the integrality property. The results imply that SG does not converge to the Lagrangian Dual. Details can be found in Appendices 8.6-8.9.

Table 6.14: ESB Main Experiments

		SG				MILP						LP			
		CPU Time		Gap(%)		CPU Time		LB(%)		UB(%)		CPU Time		LB(%)	
Set	n	avg	max	avg	max	avg	max	avg	max	avg	max	avg	max	avg	max
1	50	54.2	102	1.46	3.04	324.4	812	1.30	2.66	0.19	0.46	19.8	37	0.05	0.08
	100	381.4	663	1.77	2.41	3600 (9)	3600 (9)	-	-	-15.38 (9)	0.00 (9)	203	330	0.13	0.18
	200	1801.3	3600	1.87	2.49	**	**	**	**	**	**	3411.8	3600	-37.68	0.15
2	50	54.3	83	1.81	4.22	3381.7	3600	0.68	1.38	0.07	0.76	31	44	0.09	0.12
	100	168.6	198	1.23	1.84	3600	3600	-	-	-8.62	-7.41	105.9	144	0.09	0.12
	200	794.2	1029	0.55	0.74	3600	3600	-	-	-5.77	-4.61	3078.9	3600	-2.18	0.13

Table 6.15: ATA Main Experiments

		SG				MILP						LP			
		CPU Time		Gap		CPU Time		LB(%)		UB(%)		CPU Time		LB(%)	
set	n	avg	max	avg	max	avg	max	avg	max	avg	max	avg	max	avg	max
1	50	68.7	76	10.58	13.00	3600	3600	0.02	0.03	-6.98	-3.59	1198.2	1401	0.09	0.21
	100	441	574	13.89	14.43	3600	3600	-	-	-21.09	-19.83	3600	3600	-89.55	-78.45
	200	3600	3600	21.59	33.68	-*	-*	-*	-*	-*	-*	3600	3600	-80.08	-84.21
2	50	114.4	137	15.34	17.41	3600 (6)	3600 (6)	-	-	-33.49 (6)	-18.37 (6)	1744.2	3157	0.30	0.50
	100	570.6	666	6.18	8.97	**	**	**	**	**	**	3600	3600	-97.19	-94.83
	200	2839.4	3087	2.62	3.03	**	**	**	**	**	**	2839.4	3087	-52.32	-91.85

Subgradient Algorithm results for the ICN instances can be seen in Table 6.16. There are different number of aircrafts in each day and each day has different transfer passenger ratio (π) configurations. IUB denotes the initial upperbound found by SG with all initial Lagrange multipliers set to 0. UB and LB denote the tightest upper and lower bounds obtained by SG. Since these problems are significantly larger than the ESB and ATA instances, we use a time limit of one hour for SG. Moreover, comparison with the LP relaxation and the MILP models was not possible as MILP model cannot even construct those models due to the considerable increase in problem size. Thus, currently the only possible way of getting a lower bound is the proposed Subgradient Algorithm. We observe that for instances with relatively low transfer passenger ratios ($\pi = 0.1$), the gaps obtained by SG can be considered relatively good considering the size of these instances. However, the gaps tend to increase as π increases. This could be due to the poor performance of the upper bound, lower bound or both.

Our upper bound mechanism is based on finding feasible solutions utilizing SP1 solutions. SP1 only considers non-transfer passengers in the objective function, which may be the cause of higher gaps at higher transfer passenger ratio.

Table 6.16: ICN SG Results

Day	π	IUB	UB	Time	# of Iterations	LB	Gap (%)
Monday (n= 297)	0.1	25907380	25907380	3613	305	23928326	7.64
	0.3	27042985	27033755	3606	210	22009265	18.58
	0.5	27887245	27887245	3601	135	19507766	30.05
Tuesday (n= 290)	0.1	24012255	24012255	3648	217	22518088	6.22
	0.3	24918480	24918480	3617	235	21098172	15.33
	0.5	25940875	25940875	3614	223	19857466	23.45
Wednesday (n= 279)	0.1	23424200	23415730	3602	351	21685956	7.39
	0.3	24859210	24859210	3616	275	20594025	17.16
	0.5	25999230	25999230	3659	247	18968451	27.04
Thursday (n= 289)	0.3	26082180	26082180	3693	157	21023036	19.40
	0.5	27374815	27374815	3664	194	19622133	28.32
Friday (n= 294)	0.1	24989575	24989575	3640	104	22728861	9.05
	0.3	26258000	26258000	3645	73	20886265	20.46
	0.5	27275220	27275220	3620	90	18443555	32.38
Saturday (n= 290)	0.1	25002270	25002270	3606	291	23427295	6.30
	0.3	26192655	26192655	3611	162	21134752	19.31
	0.5	27179305	27179305	3621	218	19725826	27.42
Sunday (n= 304)	0.1	27984650	27984650	3611	180	25787390	7.86
	0.3	28838605	28838605	3646	128	23700085	17.82
	0.5	30017515	30017515	3653	115	21059980	29.84

In order to observe the performance of SG lowerbounds more accurately, we calculated new gaps using upperbounds suggested by Karsu and Solyalı [15], which are tighter than SG upperbounds except for one instance (Friday, $\pi = 0.1$). Results can be found in Table 6.17. RFH is the upperbound developed by Karsu and Solyalı [15] through a Restricted formulation-based heuristic. SG(LB) denotes the lowerbound found by SG and GAPRFH (%) denotes how far SG lowerbound is from the upperbound suggested by Karsu & Solyalı [15].

$$GAPRFH(\%) = \frac{(RFH - SG(LB))}{RFH} \times 100$$

As seen in Table 6.17 the gaps are slightly lower.

Table 6.17: ICN Comparisons

Day	π	RFH	SG (LB)	GAPRFH (%)
Monday (n= 297)	0.1	25829915	23928326	7.36
	0.3	26841520	22009265	18.00
	0.5	27750140	19507766	29.70
Tuesday (n= 290)	0.1	23981610	22518088	6.10
	0.3	24838515	21098172	15.06
	0.5	25899600	19857466	23.33
Wednesday (n= 279)	0.1	23382520	21685956	7.26
	0.3	24780635	20594025	16.89
	0.5	25729615	18968451	26.28
Thursday (n= 289)	0.3	26040010	21023036	19.27
	0.5	27186485	19622133	27.82
Friday (n= 294)	0.1	25006045	22728861	9.11
	0.3	26226690	20886265	20.36
	0.5	27100040	18443555	31.94
Saturday (n= 290)	0.1	24975980	23427295	6.20
	0.3	26060650	21134752	18.90
	0.5	26987935	19725826	26.91
Sunday (n= 304)	0.1	27925355	25787390	7.66
	0.3	28665125	23700085	17.32
	0.5	29897910	21059980	29.56

Chapter 7

Conclusion

We have considered Lagrangian relaxation based algorithms for the airport gate assignment problem (AGAP). AGAP is a widely studied problem in the literature as it helps making one of the operational decisions in airports. AGAP focuses on the assignment of aircraft to gates or to the apron while avoiding any flight conflicts. Our objective is to minimize the total distance travelled by passengers while keeping the assignments to the apron at minimum. Due to the intractability caused by the large problem size, most of the previous studies on AGAP are based on heuristic or metaheuristic approaches, which do not have a mechanism to compare the quality of the suggested upper bounds. In this work, we use algorithms that are able to find both upper and lower bounds; and hence can report on the quality of the obtained upper bound. To the best of our knowledge, this is the first Lagrangian relaxation application to AGAP in the literature.

We propose a MILP formulation for the problem using 4-index decision variables. We then solve the Lagrangian dual problem utilizing Subgradient and Bundle Methods. Both algorithms have been widely adapted in the literature for different problem types. We conduct extensive computational experiments to observe the performances of the proposed algorithms using real-life based instances from three airports: Esenboğa, Atatürk and Incheon. The results of our experiments show that for the AGAP, subgradient algorithm outperforms the bundle approach owing to its time advantage over the bundle method. Comparing the obtained lower bounds to the LP relaxation bound on smaller problems, we conclude that convergence to the Lagrangian dual is not achieved in both methods. However, the subgradient method still has the advantage of providing stronger bounds for large instances compared to the ones returned by CPLEX, which is used to solve the MILP model.

There are various further research directions that can be considered. In terms of tighter bounds, upper bound improvement methods such as neighbourhood search can be studied. For obtaining stronger lower bounds, branch and price and Benders decomposition algorithms can be devised. We are currently working on adapting Benders Decomposition on AGAP. Moreover, a stochastic component could be added to the model. Due to the uncertainties in real life, aircrafts may not arrive/depart on the scheduled time. Therefore, uncertain arrival/departure times may be considered and a stochastic or robust optimization model can be formulated to address the uncertainties.

Bibliography

[1] TAV Airports, "ESB Gelis." https://webcmsesb.tav.aero/files/1551878584_ESB%20Gelis_2019.pdf, 2019. Accessed: July 10, 2023.

[2] "ATA Layout." http://www.primeclass.com.tr/tr/Documents/istanbul Accessed: July 10, 2023.

[3] G. S. Daş, F. Gzara, and T. Stützle, "A review on airport gate assignment problems: Single versus multi objective approaches," *Omega*, vol. 92, p. 102146, 2020.

[4] R. S. Mangoubi and D. F. X. Mathaisel, "Optimizing gate assignments at airport terminals," *Transportation Science*, vol. 19, no. 2, pp. 173–188, 1985.

[5] S. Yan and C.-M. Huo, "Optimization of multiple objective gate assignments," *Transportation Research Part A: Policy and Practice*, vol. 35, no. 5, pp. 413–432, 2001.

[6] K. Srihari and R. Muthukrishnan, "An expert system methodology for aircraft-gate assignment," *Computers & Industrial Engineering*, vol. 21, no. 1-4, pp. 101–105, 1991.

[7] U. Dorndorf, F. Jaehn, and E. Pesch, "Modelling robust flight-gate scheduling as a clique partitioning problem," *Transportation Science*, vol. 42, no. 3, pp. 292–301, 2008.

[8] U. Dorndorf, A. Drexl, Y. Nikulin, and E. Pesch, "Flight gate scheduling: State-of-the-art and recent developments," *Omega*, vol. 35, no. 3, pp. 326–334, 2007.

[9] U. Dorndorf, F. Jaehn, C. Lin, H. Ma, and E. Pesch, "Disruption management in flight gate scheduling," *Statistica Neerlandica*, vol. 61, no. 1, pp. 92–114, 2007.

[10] G. S. Daş, "New multi objective models for the gate assignment problem," *Computers Industrial Engineering*, vol. 109, pp. 347–356, 2017.

[11] C. Yu, D. Zhang, and H. Y. Lau, "Mip-based heuristics for solving robust gate assignment problems," *Computers & Industrial Engineering*, vol. 93, pp. 171–191, 2016.

[12] A. Lim, B. Rodrigues, and Y. Zhu, "Airport gate scheduling with time windows," *Artificial Intelligence Review*, vol. 24, pp. 5–31, 2005.

[13] A. Drexl and Y. Nikulin, "Multicriteria airport gate assignment and pareto simulated annealing," *Iie Transactions*, vol. 40, no. 4, pp. 385–397, 2008.

[14] Özlem Karsu, M. Azizoğlu, and K. Alanlı, "Exact and heuristic solution approaches for the airport gate assignment problem," *Omega*, vol. 103, p. 102422, 2021.

[15] Ö. Karsu and O. Solyalı, "A new formulation and an effective matheuristic for the airport gate assignment problem," *Computers & Operations Research*, vol. 151, p. 106073, 2023.

[16] C.-H. Cheng, S. C. Ho, and C.-L. Kwan, "The use of meta-heuristics for airport gate assignment," *Expert systems with applications*, vol. 39, no. 16, pp. 12430–12437, 2012.

[17] R. P. Brazile and K. M. Swigger, "Gates: An airline gate assignment and tracking expert system," *IEEE Intelligent Systems*, vol. 3, no. 02, pp. 33–39, 1988.

[18] G. D. Gosling, "Design of an expert system for aircraft gate assignment," *Transportation Research Part A: General*, vol. 24, no. 1, pp. 59–69, 1990.

[19] A. Haghani and M.-C. Chen, "Optimizing gate assignments at airport terminals," *Transportation Research Part A: Policy and Practice*, vol. 32, no. 6, pp. 437–454, 1998.

[20] A. Bolat, "Assigning arriving flights at an airport to the available gates," *Journal of the Operational Research Society*, vol. 50, no. 1, pp. 23–34, 1999.

[21] A. Aktel, B. Yagmahan, T. Özcan, M. M. Yenisey, and E. Sansarcı, "The comparison of the metaheuristic algorithms performances on airport gate assignment problem," *Transportation research procedia*, vol. 22, pp. 469–478, 2017.

[22] M. Li, J.-K. Hao, and Q. Wu, "Learning-driven feasible and infeasible tabu search for airport gate assignment," *European Journal of Operational Research*, vol. 302, no. 1, pp. 172–186, 2022.

[23] M. Held and R. M. Karp, "The traveling-salesman problem and minimum spanning trees," *Operations Research*, vol. 18, no. 6, pp. 1138–1162, 1970.

[24] M. Held and R. M. Karp, "The traveling-salesman problem and minimum spanning trees: Part ii," *Mathematical programming*, vol. 1, no. 1, pp. 6–25, 1971.

[25] M. Fisher, "The lagrangian relaxation method for solving integer programming problems," *Manag. Sci.*, vol. 50, pp. 1861–1871, 2004.

[26] A. Frangioni, "About lagrangian methods in integer optimization," *Annals of Operations Research*, vol. 139, pp. 163–193, 2005.

[27] S. Elhedhli, J.-L. Goffin, and J.-P. Vial, *Nondifferentiable optimization*, pp. 2584–2590. Boston, MA: Springer US, 2009.

[28] A. Frangioni, "Generalized bundle methods," *SIAM Journal on Optimization*, vol. 13, no. 1, pp. 117–156, 2002.

[29] I. Contreras, J. Díaz, and E. Fernández, "Lagrangean relaxation for the capacitated hub location problem with single assignment," *OR Spectrum*, vol. 31, p. 483–505, 2009.

[30] A. Frangioni and G. Gallo, "A bundle type dual-ascent approach to linear multicommodity min-cost flow problems," *INFORMS Journal on Computing*, vol. 11, pp. 370–393, 1999.

[31] A. Frangioni and E. Gorgone, "Bundle methods for sum-functions with "easy" components: applications to multicommodity network design," *Mathematical Programming*, vol. 145, pp. 133–161, 2014.

[32] A. Kadri, O. Koné, and B. Gendron, "A lagrangian heuristic for the multicommodity capacitated location problem with balancing requirements," *Computers & Operations Research*, vol. 142, p. 105720, 2022.

[33] B. Gendron, J.-Y. Potvin, and P. Soriano, "A tabu search with slope scaling for the multicommodity capacitated location problem with balancing requirements," *Annals of Operations Research*, vol. 122, pp. 193–217, 2003.

[34] T. G. Crainic, A. Frangioni, and B. Gendron, "Bundle-based relaxation methods for multicommodity capacitated fixed charge network design," *Discrete Applied Mathematics*, vol. 112, no. 1, pp. 73–99, 2001. Combinatorial Optimization Symposium, Selected Papers.

[35] R. S. Shibasaki, M. Baiou, F. Barahona, P. Mahey, and M. C. de Souza, "Lagrangian bounds for large-scale multicommodity network design: a comparison between volume and bundle methods," *International Transactions in Operational Research*, vol. 28, no. 1, pp. 296–326, 2021.

[36] S. D. Jena, J.-F. Cordeau, and B. Gendron, "Lagrangian heuristics for large-scale dynamic facility location with generalized modular capacities," *INFORMS Journal on Computing*, vol. 29, no. 3, pp. 388–404, 2017.

[37] M. Krishnamoorthy, A. T. Ernst, and D. Baatar, "Algorithms for large scale shift minimisation personnel task scheduling problems," *European Journal of Operational Research*, vol. 219, no. 1, pp. 34–48, 2012.

[38] A. Haghani and M.-C. Chen, "Optimizing gate assignments at airport terminals," *Transportation Research Part A: Policy and Practice*, vol. 32, no. 6, pp. 437–454, 1998.

[39] B. Maharjan and T. I. Matis, "Multi-commodity flow network model of the flight gate assignment problem," *Computers & Industrial Engineering*, vol. 63, no. 4, pp. 1135–1144, 2012.

[40] J. Xu and G. Bailey, "The airport gate assignment problem: mathematical model and a tabu search algorithm," in *Proceedings of the 34th annual Hawaii international conference on system sciences*, pp. 10–pp, IEEE, 2001.

[41] J. E. Beasley, *Lagrangian Relaxation*, p. 243–303. USA: John Wiley Sons, Inc., 1993.

[42] A. Frangioni, *Dual-ascent methods and multicommodity flow problems.* Università degli studi di Pisa. Dipartimento di informatica, 1997.

[43] C. Ersan, *Implementation of Airport Gate Assignment Problem.* PhD thesis, Marmara Universitesi (Turkey), 2001.

[44] P. Schulze and R. Walter, "A note on the exact solution of the minimum squared load assignment problem," *Computers & Operations Research*, p. 106309, 2023.

Chapter 8

Appendices

8.1 Esenboğa Airport's Layout

Figure 8.1: Esenboğa Airport's Layout [1]

8.2 Atatürk Airport's Layout

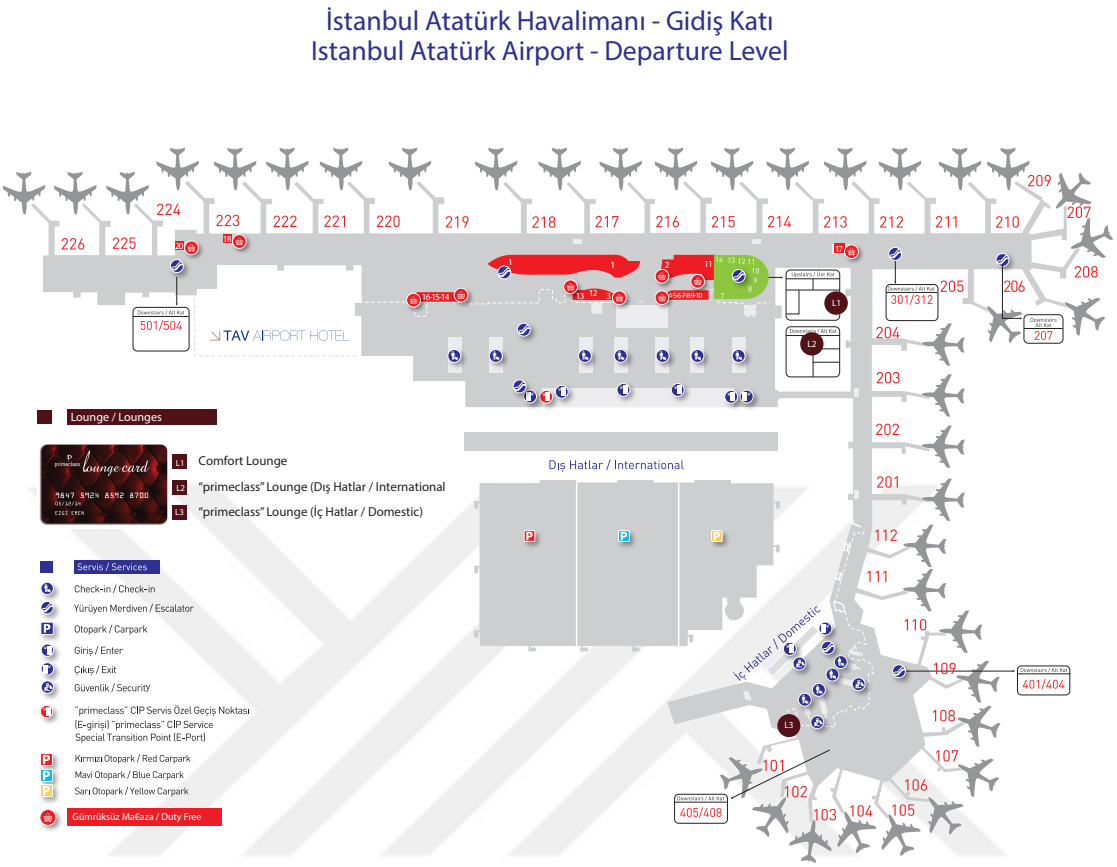

Figure 8.2: Atatürk Airport's Layout [2]

8.3 SG λ Trials for a ESB Instance

Table 8.1: λ Trials for a ESB set1 instance

Trial	IUB	UB	Time	# of Iterations	LB
1	73203	73203	53	2109	72118.3
2	73203	73203	77	3222	72120.2
3	73203	73203	58	2515	71789.5

8.4 SG Alternatives Comparison

Table 8.2: Alternatives Comparison (ESB Set1 n=50)

Upperbounds				Lowerbounds				Individual Gaps (%)			
SG	SGB	SGA	SGO	SG	SGB	SGA	SGO	SG	SGB	SGA	SGO
23956	23956	23956	23956	23695.7	23695.6	23698	23695	1.09	1.09	1.08	1.09
26796	26815	26825	26815	26493.1	26492	26489.4	26488.3	1.13	1.20	1.25	1.22
23440	23443	23413	23443	22728.5	22725.2	22724.6	22725.3	3.04	3.06	2.94	3.06
26459	26459	26459	26459	26140.8	26142.7	26137.1	26126.3	1.20	1.20	1.22	1.26
24202	24202	24202	24138	23861	23867.2	23869	23870.3	1.41	1.38	1.38	1.11
23381	23399	23399	23399	22857.9	22860.4	22859.8	22855	2.24	2.30	2.30	2.32
25184	25153	25184	25184	24951.6	24951.2	24949	24946.8	0.92	0.80	0.93	0.94
24929	24929	24929	24927	24502.6	24494	24496.9	24503.8	1.71	1.74	1.73	1.70
24707	24707	24659	24631	24310.9	24311.4	24300.8	24310.3	1.60	1.60	1.45	1.30
27348	27350	27352	27352	27262.9	27259.3	27262.8	27262.7	0.31	0.33	0.33	0.33

Table 8.3: Alternatives Comparison (ESB Set2 n=50)

Upperbounds				Lowerbounds				Individual Gaps (%)			
SG	SGB	SGA	SGO	SG	SGB	SGA	SGO	SG	SGB	SGA	SGO
73203	73203	73203	73203	72118.3	72120.3	72116.2	72107.5	1.48	1.48	1.48	1.50
69459	69459	69459	69459	67538.7	67563.7	67547.8	67554.4	2.76	2.73	2.75	2.74
79175	79175	79175	79175	77066.9	77109	77062.8	77090.5	2.66	2.61	2.67	2.63
81156	81156	81156	81156	80079	80096.6	80082.3	80070	1.33	1.31	1.32	1.34
79689	79689	79689	79689	79080.8	79101.7	79066.2	79072.4	0.76	0.74	0.78	0.77
75636	75636	75636	75636	74500.3	74536.9	74478.1	74454.5	1.50	1.45	1.53	1.56
88401	88401	88401	88401	84671.7	84693.1	84688.7	84698	4.22	4.19	4.20	4.19
80930	80934	80908	80910	80452	80474.3	80450.8	80441.7	0.59	0.57	0.57	0.58
78799	78799	78799	78799	77912.9	77936.3	77908.7	77898.4	1.12	1.09	1.13	1.14
69113	69113	69113	69113	67967	67985.3	67968.7	67978	1.66	1.63	1.66	1.64

8.5 Bundle and SG Comparisons

Table 8.4: Bundle vs SG (ATA Set1 n= 50)

	Bundle				SG			
n	UB	Time	LB	GAP	UB	Time	LB	Gap(%)
50	35284	600	31162.9	11.68	35284	66	31290.5	11.32
	34339	601	29687.6	13.55	34339	76	29875.4	13.00
	35081	602	31631.3	9.83	35081	64	31799.6	9.35
	36740	603	32748.1	10.87	36740	72	32916	10.41
	35622	600	31430.8	11.77	35622	73	31572.2	11.37
	36163	603	31823.4	12.00	36163	66	31979.7	11.57
	35789	604	31693.1	11.44	35789	71	31828	11.07
	34619	602	31138.3	10.05	34619	59	31292.7	9.61
	34922	602	31627.4	9.43	34922	69	31834.7	8.84
	38237	603	34488.4	9.80	38237	71	34703.9	9.24
100	83275	605	68742	17.45	83275	331	72015.3	13.52
	82458	601	67186.7	18.52	82458	483	70676.7	14.29
	87017	603	71581.5	17.74	87017	387	75050.4	13.75
	79644	600	65038.5	18.34	79644	435	68230.6	14.33
	84554	608	69482	17.83	84554	399	73751	12.78
	84394	610	69900.2	17.17	84394	507	73188.8	13.28
	78457	605	63953	18.49	78457	386	67132.3	14.43
	82973	606	67912.9	18.15	82973	393	71239.9	14.14
	85826	603	70256.4	18.14	85826	574	73870.8	13.93
	83306	607	68001.3	18.37	83306	515	71293	14.42
200	211449	605	132126	37.51	211449	3601	173021	18.17
	229091	613	141494	38.24	229091	3601	180649	21.15
	221670	615	147737	33.35	223519	3600	185656	16.94
	263020	603	171731	34.71	263020	3610	221059	15.95
	217455	607	141204	35.07	217455	3601	179951	17.25
	221043	603	143170	35.23	221043	3603	181818	17.75
	243669	611	144122	40.85	243669	3611	166172	31.80
	222288	607	136125	38.76	222288	3606	147421	33.68
	220938	600	145774	34.02	220938	3615	185513	16.03
	239492	600	147267	38.51	239492	3600	174326	27.21

Table 8.5: Bundle vs SG (ATA Set2 n= 50)

	Bundle				SG			
n	UB	Time	LB	GAP	UB	Time	LB	Gap(%)
50	48120	600	39896.4	17.09	48120	114	40706.9	15.41
	48889	600	38375.3	21.51	48889	114	40377.6	17.41
	55704	602	45936	17.54	55704	102	47847.6	14.10
	62451	601	51720.4	17.18	62451	110	53313.7	14.63
	51659	601	41770.8	19.14	51659	116	43810.4	15.19
	47784	602	39952.9	16.39	47784	117	40587.1	15.06
	65437	600	53459.2	18.30	65437	137	55142	15.73
	57738	601	46804.6	18.94	57738	107	48729	15.60
	58139	600	46796.1	19.51	58139	123	48867.7	15.95
	49742	600	41938.1	15.69	49742	104	42600.1	14.36
100	237318	604	213212	10.16	237318	619	219343	7.57
	265274	606	247753	6.60	265274	592	252859	4.68
	262912	605	235368	10.48	262912	661	243384	7.43
	281458	602	261899	6.95	281458	418	267579	4.93
	271838	603	255450	6.03	271838	576	259629	4.49
	274301	606	256781	6.39	274301	656	262142	4.43
	233112	602	210987	9.49	233112	620	216521	7.12
	265462	604	243196	8.39	265462	463	248373	6.44
	299192	601	276155	7.70	299192	435	281889	5.78
	236533	601	207843	12.13	236533	666	215305	8.97
200	749545	600	704503	6.01	749610	2624	733377	2.17
	791720	608	746611	5.70	792805	2742	774518	2.31
	830542	606	781265	5.93	830728	2890	809602	2.54
	806860	609	756315	6.26	808073	2695	784027	2.98
	780831	604	729910	6.52	781373	2936	757664	3.03
	776935	603	722089	7.06	777056	2893	753672	3.01
	820012	600	773910	5.62	820012	2731	800810	2.34
	782897	602	737891	5.75	783612	2986	765786	2.27
	798698	606	748182	6.32	798974	3087	775274	2.97
	798675	602	754408	5.54	799990	2810	779725	2.53

Table 8.6: Bundle vs SG (ESB Set1 n= 50)

	Bundle				SG			
n	UB	Time	LB	GAP	UB	Time	LB	Gap(%)
50	23956	601	23649.4	1.28	23956	65	23695.7	1.09
	26825	602	26383.4	1.65	26796	102	26493.1	1.13
	23443	601	22661.4	3.33	23440	50	22728.5	3.04
	26459	600	26089.1	1.40	26459	60	26140.8	1.20
	24202	601	23821.7	1.57	24202	36	23861	1.41
	23399	600	22821.7	2.47	23381	54	22857.9	2.24
	25184	600	24913.5	1.07	25184	40	24951.6	0.92
	24929	600	24452.5	1.91	24929	35	24502.6	1.71
	24707	602	24272.7	1.76	24707	61	24310.9	1.60
	27369	600	27225.5	0.52	27348	39	27262.9	0.31
100	86856	601	83529.3	3.83	86856	270	85382	1.70
	76744	600	73251.1	4.55	76744	446	75440.1	1.70
	81555	602	78343.7	3.94	81555	281	80194	1.67
	79453	600	75948.8	4.41	79453	450	77883.6	1.98
	95020	601	92278.7	2.88	95020	337	93677.7	1.41
	77061	600	75158.1	2.47	77061	301	76461.5	0.78
	89702	604	85935.1	4.20	89702	333	87542.1	2.41
	90897	602	87138.8	4.13	90897	225	88753.1	2.36
	92074	602	89631	2.65	92074	508	90787.8	1.40
	83886	600	79636	5.07	83886	663	81949.3	2.31
200	338313	607	323713	4.32	338313	1052	330923	2.18
	338542	601	323095	4.56	338544	1365	330113	2.49
	320629	610	306781	4.32	320629	1058	313650	2.18
	347932	610	336355	3.33	347932	1380	342816	1.47
	344976	612	332865	3.51	345168	3397	338943	1.80
	341038	607	330482	3.10	341038	1972	336360	1.37
	344099	604	329653	4.20	344166	1456	336813	2.14
	320235	610	308996	3.51	320235	3602	315282	1.55
	337698	615	323854	4.10	337698	1346	331021	1.98
	340297	604	329437	3.19	340743	1385	335342	1.59

Table 8.7: Bundle vs SG (ESB Set2 n= 50)

	Bundle				SG			
n	UB	Time	LB	GAP	UB	Time	LB	Gap(%)
50	73203	605	72032.5	1.60	73203	49	72118.3	1.48
	69459	604	67422.5	2.93	69459	41	67538.7	2.76
	78653	602	76892.4	2.24	79175	50	77066.9	2.66
	81156	602	79994.5	1.43	81156	75	80079	1.33
	79689	602	78979.9	0.89	79689	59	79080.8	0.76
	75619	601	74304.6	1.74	75636	83	74500.3	1.50
	88401	601	84506.1	4.41	88401	37	84671.7	4.22
	80904	601	80395	0.63	80930	48	80452	0.59
	78799	601	77836.7	1.22	78799	52	77912.9	1.12
	69113	600	67786.1	1.92	69113	49	67967	1.66
100	207333	603	204536	1.35	207568	135	205355	1.07
	212446	603	209154	1.55	212446	182	210439	0.94
	205315	606	202045	1.59	205315	136	203442	0.91
	203626	600	200030	1.77	203801	196	201369	1.19
	209626	600	206551	1.47	209881	178	207599	1.09
	222075	603	218238	1.73	222432	180	219575	1.28
	212082	600	206346	2.70	212269	156	208435	1.81
	211336	601	205566	2.73	211336	140	207457	1.84
	201812	604	198881	1.45	201940	198	199813	1.05
	211856	602	208387	1.64	212045	185	209752	1.08
200	477341	602	471086	1.31	477635	701	476441	0.25
	511712	604	502746	1.75	512347	651	509063	0.64
	468288	605	460743	1.61	468804	752	466749	0.44
	503200	602	495688	1.49	504505	723	501456	0.60
	503395	603	493251	2.02	503453	607	500070	0.67
	503102	606	496208	1.37	504221	879	501309	0.58
	502625	606	492685	1.98	503161	915	499444	0.74
	477439	608	470592	1.43	477794	1029	476400	0.29
	499819	602	489599	2.04	500327	761	496653	0.73
	503274	604	495026	1.64	504255	924	501364	0.57

8.6 MILP and SG bounds Compared for ESB Instances

Table 8.8: MILP and SG bounds Compared for ESB Instances

	Set 1		Set 2	
n	LB(%)	UB(%)	LB(%)	UB(%)
50	1.01	0.09	0.61	-0.14
	1.09	0.05	1.38	0.36
	2.66	0.46	0.63	0.76
	1.19	0.02	0.52	-0.10
	1.09	0.33	0.49	0.05
	2.17	0.12	0.69	0.07
	0.71	0.22	0.93	-0.12
	1.49	0.24	0.48	0.12
	1.25	0.37	0.40	0.15
	0.31	0.00	0.65	-0.41
100	***	-21.80	***	-8.38
	***	-17.70	***	-8.72
	***	-14.65	***	-8.50
	***	***	***	-8.84
	***	-17.39	***	-7.89
	***	-14.27	***	-8.20
	***	-14.04	***	-7.41
	***	-21.79	***	-10.65
	***	-18.54	***	-9.69
	***	-13.61	***	-7.93
200	*-*-	*-*-	***	-6.93
	--	*-*-	***	-4.76
	--	*-*-	***	-6.82
	--	*-*-	***	-5.77
	--	*-*-	***	-5.22
	--	*-*-	***	-6.91
	--	*-*-	***	-4.61
	--	*-*-	***	-5.71
	--	*-*-	***	-5.31
	--	*-*-	***	-5.64

Note: *-*- denotes that problem is not solved as neither upper nor lower bound was reported for one of smallest size instances, and *** denotes that the bound was not reported by cplex.

8.7 LP Relaxation and SG bounds Compared for ESB Instances

Table 8.9: LP Relaxation and SG bounds Compared for ESB Instances

	Set 1			Set 2		
	Time			Time		
n	LP	SG	LB(%)	LP	SG	LB(%)
50	18	65	0.03	27	49	0.10
	25	102	0.06	33	41	0.12
	14	50	0.07	27	50	0.11
	12	60	0.07	30	75	0.08
	9	36	0.08	30	59	0.07
	37	54	0.03	30	83	0.09
	31	40	0.03	36	37	0.07
	28	35	0.06	44	48	0.07
	12	61	0.05	24	52	0.06
	12	39	0.05	29	49	0.10
100	294	270	0.18	77	135	0.07
	330	446	0.11	93	182	0.09
	95	281	0.12	138	136	0.11
	130	450	0.17	122	196	0.11
	161	337	0.09	82	178	0.07
	201	301	0.07	94	180	0.09
	246	333	0.16	99	156	0.07
	85	225	0.16	144	140	0.11
	247	508	0.10	107	198	0.08
	241	663	0.16	103	185	0.12
200	3611	1052	-42.23	2670	701	0.13
	3605	1365	-46.97	3607	651	-4.32
	3606	1058	-46.37	3607	752	-4.62
	3606	1380	-36.71	3079	723	0.10
	2935	3397	0.15	3605	607	-3.85
	3611	1972	-38.75	2645	879	0.11
	2325	1456	0.15	2017	915	0.10
	3606	3602	-36.66	3606	1029	-4.94
	3607	1346	-34.25	3606	761	-4.58
	3606	1385	-95.19	2347	924	0.10

8.8 MILP and SG bounds Compared for ATA Instances

Table 8.10: MILP and SG bounds Compared for ATA Instances

	Set 1		Set 2	
n	LB(%)	UB(%)	LB(%)	UB(%)
50	0.02	-4.72	***	-33.39
	0.03	-4.85	***	-18.37
	0.03	-7.98	***	-37.06
	0.02	-6.49	***	***
	0.03	-9.83	***	-28.71
	0.02	-7.02	***	-38.33
	0.03	-9.44	***	***
	0.02	-10.05	***	***
	0.02	-5.86	***	0.00
	0.02	-3.59	***	-45.05

8.9 LP Relaxation and SG bounds Compared for ATA Instances

Table 8.11: LP Relaxation and SG bounds Compared for ATA Instances

	Set 1			Set 2		
n	LP Time	SG Time	LB(%)	LP Time	SG Time	LB(%)
50	1032	66	0.05	1355	114	0.35
	1392	76	0.10	1223	114	0.50
	1050	64	0.21	1726	102	0.24
	1401	72	0.06	1649	110	0.26
	1331	73	0.11	1432	116	0.20
	1084	66	0.05	1343	117	0.27
	1167	71	0.14	2210	137	0.18
	1212	59	0.10	1803	107	0.28
	1001	69	0.05	1544	123	0.32
	1312	71	0.07	3157	104	0.45
100	3610	331	-92.04	3610	619	-97.20
	3605	483	-79.07	3624	592	-97.61
	3828	387	-92.47	3613	661	-97.40
	3659	435	-91.72	3890	418	-94.83
	3915	399	-92.51	3682	576	-97.67
	3734	507	-92.63	3611	656	-97.57
	3604	386	-91.75	3826	620	-97.11
	3659	393	-78.45	3616	463	-97.52
	4119	574	-92.75	3657	435	-97.90
	3883	515	-92.08	3611	666	-97.09
200	3618	3601	-83.89	3625	2624	-96.20
	3619	3601	-87.20	3621	2742	-96.78
	3620	3600	-82.50	3620	2890	-95.99
	3620	3610	-89.50	3621	2695	-96.04
	3620	3601	-83.71	3620	2936	-96.13
	3621	3603	-83.32	3619	2893	-95.98
	3623	3611	-81.34	3620	2731	-96.13
	3621	3606	-80.08	3621	2986	-96.16
	3622	3615	-83.78	3621	3087	-96.73
	3621	3600	-86.76	3618	2810	-52.32