A COLUMN GENERATION ALGORITHM FOR ROBUST GATE ASSIGNMENT PROBLEMS

by

SONER BEYHAN

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of
the requirements for the degree of
Master of Science

Sabancı University
Fall 2011

A COLUMN GENERATION ALGORITHM FOR ROBUST GATE ASSIGNMENT PROBLEMS

APPROVED BY

Assist. Prof. Güvenç Şahin (Thesis Co-Supervisor) .....

Assist. Prof. Dilek Tüzün Aksu (Thesis Co-Supervisor) .....

Assoc. Prof. Bülent Çatay .....

Assoc. Prof. Temel Öncan .....

Assoc. Prof. Tonguç Ünlüyurt .....

DATE OF APPROVAL: 03.02.2012

to my beloved ones

Acknowledgments

First, I want to express my deepest gratitude to my thesis advisor Assoc. Prof. Güvenç Şahin for his remarkable patience and invaluable support. I feel extremely fortunate to have him as my advisor. I also wish to express my deepest gratitudes to Assoc. Prof. Dilek Tüzün Aksu for her support.

My special thanks goes to Tuğce Baykent for her patience and for sharing my dreams.

Mehmet Berke Pamay deserves a special mention, who has been a dearest friend, a life saver.

Berkin Keskin, Anıl Coşkun, Hasan Bayraktar and Ergün Çalışgan shared my room, supported me for all my decisions and never let me give up.

I am indebted to all my friends from Sabancı University for their motivation and endless friendship. Many special thanks go to Halil Şen, Çetin A. Suyabatmaz, Taner L. Tunç, Mahir U. Yıldırm, E. Arda Şişbot and all those others who directly and indirectly helped me.

Above all, I would like to thank my family for always being there for me. My parents and my brother supported me and showed great love, patience and support at all times.

A COLUMN GENERATION ALGORITHM FOR ROBUST GATE ASSIGNMENT PROBLEMS

Soner Beyhan

Industrial Engineering, Master of Science Thesis, 2011

Thesis Supervisors:
Assist. Prof. Güvenç Şahin
Assist. Prof. Dilek Tüzün Aksu

Keywords: robust gate assignment, column generation, heterogeneous gate assignment problem, airport operations planning

Abstract

This study adresses the Robust Gate Assignment Problem (RGAP) for the case of both homogeneous and heterogeneous gates. Due to the increased traffic and congestion at airports, scientific approaches to operational problems have gained importance in the airline industry. Operations planning has a vital importance in this environment. One of the most important problem types for the airport management is the well known gate assignment problem (GAP). In this study, a column generation (CG) algorithm is proposed to solve GAP and the algorithm is formulated as a linear programming relaxation of the set covering problem. The pricing subproblem (PSP) for the CG approach is represented with a network structure and solved using the shortest path algorithm. Results show that for both homogeneous and heterogeneous instances, the proposed CG algorithm provides optimal LP solutions according to the idle time of variance robustness measure. Insert capability suggested by Dorndorf [1] is also investigated as a robustness measure and compared to variance of idle time. A computational study performed on data sets from Bolat [2] indicate that the two robustness measures are negatively correlated.

DAYANIKLI KAPI ATAMA PROBLEMİ İÇİN KOLON TÜRETME YÖNTEMİ

Soner Beyhan

Endüstri Mühendisliği, Yüksek Lisans Tezi, 2011

Tez Danışmanı:

Yrd. Doç. Güvenç Şahin

Yrd. Doç. Dilek Tüzün Aksu

Anahtar Kelimeler: dayanıklı kapı atama problemi, kolon türetme, heterojen kapı atama problemi, havaalanı operasyonel planlama

Özet

Bu çalışmada, homojen ve heterojen kapılı durumlar için Dayanıklı Kapı Atama Problemi incelenmiştir. Havaalanlarında artan trafik tıkanıklığı nedeniyle, havayolu sektöründe operasyonel sorunlara bilimsel yaklaşımlar önem kazanmıştır. Bu ortamda operasyonel planlama hayati bir öneme sahiptir. Havaalanı yönetimi için en önemli problem türlerinden biri de kapı atama problemidir. Bu çalışmada sütun türetme yöntemi, küme kapsama probleminin bir doğrusal programlama rahatlatması olarak formüle edilmiştir. Sütun türetme yaklaşımı için fiyatlandırma alt problemi bir ağ yapısı ile temsil edilip en kısa yol yöntemi kullanılarak çözülmüştür. Dayanıklılık ölçütü olarak atıl zaman varyansı kullanıldığında hem homojen hem de heterojen örnekleri için en iyi çözümler bulunmuştur. Dorndorf [1] tarafından önerilen ekleme kabiliyeti dayanıklılık ölçütü incelenmiş ve atıl zaman varyansı ile karşılaştırılmıştır. Bolat [2] veri kümeleri üzerinden yapılan sayısal çalışmalar bu iki dayanıklılık ölçütü arasında negatif bağlantı olduğunu göstermektedir.

Table of Contents

Appendix

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION AND MOTIVATION

As air transport volume increases year by year, people travel much longer distances by large aircrafts and they fly more frequently with cheaper deals. Although the rise in demand for air transportation has also increased the cash flow for airline companies and airports, their costs have also been inflated with growing industry needs. Due to the increased traffic and congestion at airports, scientific approaches to operational problems at airports have gained importance. The right solution techniques for these problems create competitive advantage for the companies. Companies need fast and effective solutions for many different types of air traffic related problems. Consequently, airport management problems have been studied extensively in the operational research literature especially in the recent years. Airport operations planning gained a vital importance in this environment. One of the most important problems in airport operations is the well-known gate assignment problem (GAP).

In GAP, there are a number of flights that arrive at an airport and need to be assigned to a set of available gates, where aircrafts reside until their next departure. Flights and gates might have specifications and restrictions in terms of size, terminals, functions and other resources [1]. Along with these, there are two main hard constraints of this problem [3]:

- flights cannot overlap with each other (conflicting flights constraint), and
- each flight should be assigned to one and only one gate (gate coverage constraint).

Additionally, there can be several other limitations in a GAP in terms of airline

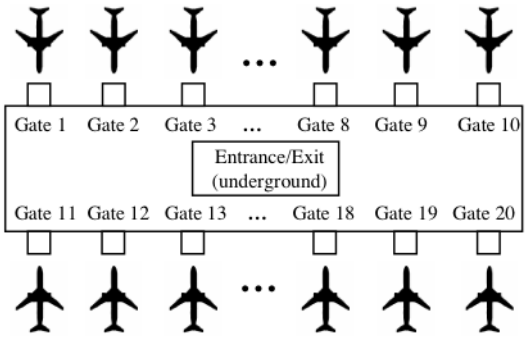

Figure 1.1: Representation of the gate assignment process

rules and airport management regulations. Some of these additional constraints can be associated with aircraft and gate size, consecutive flight assignment rules or some legally enforced restrictions. When these additional constraints are imposed on to the problem, they complicate the problem and may require novel solution methodologies. There are several different objective types [1] used within the gate assignment literature, which also leads to a variety of formulations and new solution approaches.

A very important characteristic of the real life GAP is the frequent changes in the problem environment and resources. Operational delays are common attributes of the airline industry with unpredicted weather conditions and strong security regulations. As expected, robustness gains great importance in gate assignment under these challenging conditions. Airport managements spend large amounts of money for effective planning systems. An approach that creates robust solutions can improve management's ability to react in this unstable environment.

With a large number of flights and huge airports, there are many alternative ways of assigning aircrafts to gates. They need an effective solution approach to choose from the large set of feasible solutions. In this thesis, we study two alternative measures to create robust gate assignments and propose a set covering type formulation for easy representation of these measures. A set covering model for the problem can be formulated easily, but to reach an optimal solution we need to enumarate all gate schedules. Creating all possible schedules is not a time efficient or even a feasible approach for large problems in real life. To surpass these entanglements we present

an easy to implement and effective column generation (CG) approach.

1.1 Contributions

The primary purpose of this thesis is to propose an effective, simple and unified solution approach to GAP with both homogeneous and heterogeneous gates. The following is a list of the contributions of this study:

- While there are efficient algorithms for the homogeneous GAP, the heterogeneous case cannot be solved to optimality within reasonable time using the available algorithms. We propose a column generation algorithm to solve the GAP for both homogeneous and heterogeneous cases.
- We show that the pricing subproblem of the proposed CG algorithm can be represented as a shortest path problem, which can be solved in polynomial time.
- Two robustness measures for the robust gate assignment problem (RGAP) (idle time variance and insert capability) are suggested and compared with each other. Results indicate that these two measures are negatively correlated.
- A new set of problem instances is created for the heterogeneous case by modifying the instances with homogeneous gates from the literature.

1.2 Outline

In Chapter 2, related literature and problem description are presented, and both classical and proposed formulations of GAP are described. In chapter 3, the proposed CG algorithm is presented, and homogeneous and heterogeneous versions of the algorithm are introduced separately. Also, two different robustness measures are introduced within the column generation framework. In Chapter 4, a computational study is presented where problem sets from the literature and newly generated heterogeneous instances are solved using the suggested algorithm. Finally, concluding remarks and ideas for future work are discussed in Chapter 5.

CHAPTER 2

RELATED LITERATURE AND PROBLEM DESCRIPTION

In this chapter, we present the related work for GAP and give the problem description. We refer the reader to Dorndorf et al. [1] for an extensive review of the state of the art developments for GAP.

2.1 Related Work

GAP is studied extensively in the literature from several different perspectives regarding the problem environment. These studies can be classified based on two aspects: objective type and solution method. We should note that the solution methods employed are highly correlated with the selected objective type. Dorndorf et al. [1] list the main objective types as follows:

- the number of un-gated (open) aircraft activities is minimized,
- assignment of certain aircrafts to particular preferred gates is maximized,
- the total walking distance for passengers is minimized,
- the deviation of the current schedule from a reference schedule has to be minimized in order to increase schedule attractiveness and passenger comfort, and
- the amount of expensive aircraft towing procedures (that otherwise decrease the available time for some ground service operations on the ramp as well as in the terminal) is minimized.

Babic et al. [4] formulate GAP as 0-1 integer programming model with an objective of minimizing total walking distance and solve it using a Branch and Bound algorithm. Many studies refer to Mangoubi and Mathaisel [5] for the classical formulation of GAP. In this study, the authors present a linear relaxation of an integer programming problem and use a heuristic algorithm as a solution approach. While Babic et al. [4] only consider the total walking distance of the passengers within the airport, Mangoubi and Mathaisel [5] also takes walking distance for the transfer of passengers into account.

Quadratic assignment formulations appear frequently in the literature. Xu and Bailey [6] propose a quadratic assignment model that minimizes the total passenger connection time from one flight to the next. Ding et al. [7] present a similar formulation which minimizes both the total walking distance and the number of ungated flights (the flights that cannot be assigned to any gate and left at the apron).

Robustness is studied in the GAP literature, where the impact of operational disruptions on the gate assignment is analyzed. Again, Mangoubi and Mathaisel [5] refer to this problem and propose fixed time buffers between consecutive flights, in order to avoid highly utilized gates. If a gate is highly utilized, it is also highly fragile against any delay or congestion, which is in the nature of the problem environment.

Bolat [8] also refers to the robustness issue where he claims that increasing the idle time between consecutive flights in a gate schedule creates a more robust solution. He argues that minimizing the variance of idle times in a gate schedule triggers uniformly distributed flights. As also mentioned in Mangoubi and Mathaisel [5], the buffers created between flights can handle some delays without revising the original gate assignment. We discuss this objective function approach in Section 3.3.

Dorndorf et al. [9] argue that robustness can be integrated into the gate assignment by maximizing a score that represents the schedules' ability to accept other flights without changing its regular assignments. They suggest that this score should be based on two types of moves that can be performed on gate schedules: an insert move that inserts a flight between two consecutive flights at the same gate, and a

swap move that swaps the positions of two flights between two different gates. We discuss the insert capability of schedules in more detail in Section 3.3.

Stochastic optimization approaches for RGAP are also considered in the literature, which explicitly model various sources of uncertainty in the airport operations such as weather conditions, air traffic control delays, gate breakdowns, etc. [3]. Lim and Wang [10] propose a model where they estimate the probability of conflict for each flight pair using an estimation function and try to minimize the expected number of flight conflicts. Seker [3] discusses a two-stage scenario-based optimization approach for the objective functions discussed in Bolat [8].

In this study, we propose a solution method for GAP based on a CG approach. For further information about CG, we refer the reader to Desaulniers et al. [11]. CG algorithm for GAP has not been used in the literature before, but Chapter 2 of this book discusses a CG algorithm for vehicle routing and crew scheduling problems where the pricing subproblem is formulated as a shortest path problem with resources constraints, which is highly related to CG formulation in this thesis. We will provide detailed information about the CG algorithm in Section 3.

2.2 Gate Assignment Problem (GAP)

The classical GAP aims to assign arriving aircraft to available gates at an airport. In this problem, we assume that the arrival and departure times of the aircraft will realize as planned. However, due to many disruptions that may occur in an airline operation, actual aircraft arrival and departure times may deviate significantly from the plan. This results in a large number of gate changes during the day, which impacts the operational cost and service quality adversely. Creating a gate assignment plan that is robust to disruptions is critical for the smooth operation of an airline. In this study, we focus on RGAP which aims to create a robust gate assignment that enables easy recovery from operational disruptions.

We suppose that the problem is to be solved for a finite planning horizon during which the availability of gates is known a priori. The planning horizon starts at a specified starting point in time and continues until an end point, which mostly

coincide with the departure time of the last flight. We assume that a master plan of gate assignments is to be prepared before the beginning of the planning horizon. A schedule of arriving and departing flights is given. A pair of an arriving flight and a departing flight is associated with an aircraft that performs both flights. The aircraft is to reside at a gate from the time it arrives at the airport and until it departs. In this respect, the literature uses the terms flight assignment and aircraft assignment interchangeably as the time period between the arriving flight and departing flight is referred to as the ground time of the aircraft or the flight. Hence, the basic information associated with a flight or an aircraft include arrival time and departure time that jointly imply the ground time.

In our setting, as well as in the literature, GAP contains only one resource type, which is the gate. In a real life airport environment, gates can be classified in many different ways in terms of their size, neighboring gates, terminal type (international vs. domestic) and functional type (cargo - passenger). For the homogeneous case, we assume that all gates are equivalent and identical. For the heterogeneous case, gates might be differentiated with respect to their size, terminal type and function type. In that case, the aircraft to reside at the gate also needs to be specified with additional information regarding the size, terminal and function.

2.3 Mathematical Models

In this section, we first revisit the classical integer programming (IP) formulation for GAP. Then we present an alternative set covering formulation, for both the classical and robust versions of the problem.

2.3.1 Classical IP Formulation

We first present the mathematical model due to Mangoubi and Mathaisel [5] for the classical GAP. In this model, I is the set of flights to be assigned to some gate and K denotes the set of available gates. We assume that there is a certain cost, c_{ik}, of assigning flight $i \in I$ to gate $k \in K$. For each flight $i \in I$, a_i and d_i are the arrival and departure times, respectively. The decision variable x_{ik} is equal to 1 if flight i

is assigned to gate k; and 0, otherwise. Then, the integer programming formulation is:

$$\text{minimize} \quad \sum_{i \in I} \sum_{k \in K} c_{ik} x_{ik} \tag{2.1}$$

$$\text{subject to} \sum_{k \in K} x_{ik} \geq 1 \qquad \forall i \in I \tag{2.2}$$

$$\sum_{h \in L(i)} x_{hk} + x_{ik} \leq 1 \qquad \forall i \in I, \forall k \in K \tag{2.3}$$

$$x_{ik} \in \{0, 1\} \qquad \forall i \in I, \forall k \in K \tag{2.4}$$

where $L(i)$ is the set of flights that are still on ground when flight i arrives, i.e.

$$L(i) = \{h : d_h \geq a_i, a_h \leq d_i, h \in L(i-1) \cup (i-1)\}, \forall i \in I.$$

In the classical GAP formulation (2.1)-(2.4) above, the objective function (2.1) minimizes the total cost of assigning flights to gates. Constraints (2.2) ensures that each flight is assigned to some gate while constraints (2.3) ensures that no two conflicting flights are assigned to the same gate. Constraints (2.4) honor the binary nature of the decision variables.

2.3.2 Set Covering Formulation for GAP

Many studies in the GAP literature are based on Mangoubi and Mathaisel [5] formulation given in the previous section. In this study, we propose an alternative formulation based on the set covering problem. This formulation facilitates the implementation of new robustness measures for GAP which are difficult to formulate using the mathematical model in (2.1)-(2.4).

We first present the set covering formulation which is equivalent to the problem formulation in (2.1)-(2.4). Later on, we discuss the features that can be incorporated into GAP when this set covering formulation is used instead of the classical formulation. In the set covering formulation for GAP, each flight is to be covered by a gate schedule where a gate schedule is a collection of flights that can be assigned

to a certain gate with no conflicts with respect to their arrival and departure times. J denotes the set of all feasible schedules for any gate schedule j with no conflicts and honoring all operational constraints for all flights in the schedule. We define the two problem parameters as follows:

$$b_{ij} = \begin{cases} 1, & \text{if flight } i \text{ is covered by gate schedule } j, \\ 0, \text{ otherwise.} \end{cases}$$

$$c_j = \sum_i b_{ij} c_{ij}$$

where cj is equal to cik when all gates are identical as in the homogeneous case. The decision variable y_j is equal to 1 if gate schedule $j \in J$ is selected and 0, otherwise. Then, the set-covering formulation for GAP is

$$\text{minimize} \quad \sum_{j \in J} c_j y_j \tag{2.5}$$

$$\text{subject to} \quad \sum_{j \in J} b_{ij} y_j \geq 1 \qquad \forall i \in I \tag{2.6}$$

$$y_j \in \{0, 1\} \qquad \forall j \in J \tag{2.7}$$

In this formulation, the objective function (2.5) minimizes the total cost of covering all flights by selected gate schedules while constraints (2.6) ensure that each flight is covered by at least one gate schedule. Constraints (2.7) are the integrality constraints for the decision variables.

In the classical GAP formulation given in Section 2.3.1, the cost of a gate assignment is calculated as the sum of the costs for all flight to gate assignments included in the solution. Using the above set covering formulation, the cost of a gate assignment can be based on each assigned flight itself as well as the other flights at the same gate. This property allows us to model several robustness measures which will be described later. It should also be noted that the classical GAP formulation in Section 2.2 can be revised so that the objective function reflects the costs that

are dependent on the ordering of flights at the gates. However, this requires the introduction of many additional variables and constraints to the problem formulation, which makes it difficult to solve. This issue is one of the major concerns that motivated us to propose an alternate formulation for GAP. On the other hand, the challenge with the set covering formulation is the number of decision variables (i.e. columns of the problem formulation). In a real life problem, the number of feasible gate schedules may be too many. This leads to a very large problem in terms of the number of binary decision variables. In Chapter 3, we first revisit the set covering formulation for GAP and propose a column generation algorithm to solve this problem. Then, we discuss the adaptation of the problem formulation to the heterogeneous case along with the modifications of CG algorithm to solve the heterogeneous problem.

CHAPTER 3

SOLUTION APPROACH

In Chapter 2, we have provided the set covering formulation for the GAP and RGAP. As mentioned earlier, set of all feasible gate schedules is too large, which leads to a large-scale IP problem. Therefore, we need a specialized approach to solve it efficiently. The problem of large number of variables is often handled using a CG algorithm, which consists of two components: the restricted master problem (RMP) and pricing subproblem (PSP). In this study, we solve RMP using linear programming relaxation formulation of the problem. RMP is a smaller version of the original problem that contains a subset of all possible feasible gate schedules (columns). At every iteration RMP is enlarged iteratively with new columns (i.e. decision variables) that are likely to improve the solution. The second component PSP is used to determine the new column (decision variable) to be added to RMP.

We again note that a column in RMP of the CG procedure refers to a gate schedule. It consist of consecutive flights which do not overlap with each other. The advantages of the column generation algorithm against direct approaches such as B&B are the reduced size of RMP and the ability of solving PSP very efficiently. The performance of the CG algorithm is highly dependent on efficiently solving PSP. In this study, a network representation is used to solve the pricing subproblem. This subproblem is reduced to a shortest path problem that can be solved very efficiently in polynomial time.

3.1 Column Generation Algorithm for Homogeneous GAP

Homogeneous problems consist of identical gates which can accommodate all arriving flights. The proposed set covering formulation for homogeneous GAP (2.5) - (2.7) is introduced in Section 2.4.2. Solving (2.5)-(2.7) directly is not practical due to the size of set J, which contains all possible gate schedules J. However, a column generation algorithm may be employed to solve the linear programming relaxation of (2.5)-(2.7). The column generation algorithm starts with a restricted master problem (RMP), which includes only a selected subset of gate schedules (columns)in $j \in J$, and then iteratively adds new columns to RMP to improve its objective function value. At each iteration of the column generation algorithm, RMP is solved and the optimal dual values from RMP are used to solve the PSP, where we search for a new gate schedule that is likely to improve the objective function value of RMP when the new schedule is selected. In CG algorithm (see Figure 3.1), we start with a feasible set of gate schedules (initial feasible solution) to solve our RMP. We gather dual values from the solution of RMP and solve PSP using these values to calculate the reduced cost of the schedules that do not yet exist in RMP. We find the most negative reduced cost column. If the reduced cost of the column is less than zero, we add this column to RMP. Otherwise, the most recent solution to RMP is optimal. The algorithm continues to add columns to RMP until an optimal solution is reached.

Next, we first discuss the structure of the initial solution; then, we outline the solution procedure for PSP.

3.1.1 Initial Solution

We experiment with two approaches for obtaining an initial solution: First, we start with a feasible initial solution acquired by a constructive heuristic; second, we start from an infeasible initial solution with as many gate schedules as the number of flights, each covering only one flight. Eventually the second approach also reaches a feasible schedule after some iterations, provided that there are feasible schedules

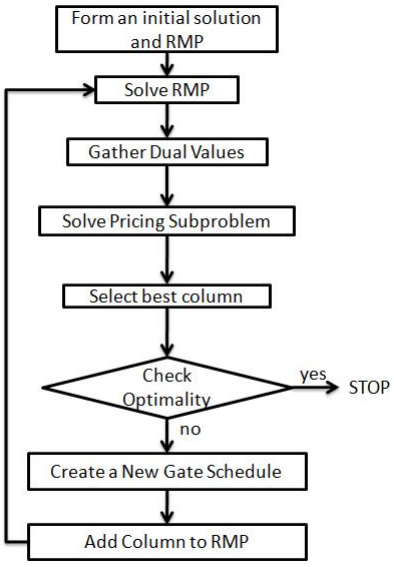

Figure 3.1: Flowchart of the CG algorithms for homogeneous GAP

for the problem instance. Starting from a good initial solution often effects the efficiency of algorithm. In order to test the impact of the initial solution on algorithm performance, we performed a computational study, where the CG algorithm is initialized using both approaches. In the next section the computational results of this study will be mentioned.

3.1.2 The Pricing Subproblem

PSP is used to find a new column that does not yet exist in RMP. As a rule of thumb, the algorithm finds the most negative reduced cost column. In order to do this, we look into the dual of RMP. By using the dual values of RMP, PSP finds the row in the dual (i.e. a column of the original problem) with the largest degree of violation. To further look into this procedure, let us first formulate the dual of RMP.

Let u_i be the dual variable corresponding to constraint i in (2.6). The dual of (2.5)-(2.7) can then be formulated as:

$$\max \quad \sum_{i \in I} u_i \tag{3.1}$$

$$\text{subject to} \sum_{i \in I} b_{ij} u_i \leq c_j \qquad \forall j \in J \tag{3.2}$$

$$u_i \geq 0 \qquad \forall i \in I \tag{3.3}$$

The pricing subproblem is to find a column j that has a negative reduced cost (i.e. a row that violates the corresponding dual constraint (3.2)). Formally, when we search for the column/schedule that is likely to make the best improvement in the objective function value of RMP, the pricing problem is to find the column j^* corresponding to a row with the largest violation with respect to constraint (3.2), i.e.

$$c_{j^*} - \sum_{i \in I} b_{ij^*} u_i = \min_{j \in J} \left\{ c_j - \sum_{i \in I} b_{ij} u_i \right\}. \tag{3.4}$$

If $c_{j^*} - \sum_{i \in I} b_{ij^*} u_i < 0$, then decision variable y_{j^*} is added to RMP. Otherwise, the optimal solution to RMP is reached.

We notice that the reduced cost of a gate schedule $j \in J$ is

$$c_j - \sum_{i \in I} b_{ij} u_i = \sum_{i \in I} b_{ij} (c_{ij} - u_i) \tag{3.5}$$

$$= \sum_{i \in j} (c_{ij} - u_i) \tag{3.6}$$

Therefore, for a gate schedule, the reduced cost is a function of the flights that are included in the schedule. We use a network representation to solve the PSP in (3.4) that finds the schedule with the minimum reduced cost. For this purpose, we create a network $G = (V, A)$. In this network representation, every flight is represented as a node. An arc is created from one flight node to the other if the two flights can be scheduled consecutively at the same gate. In this network, the vertex set V includes a dummy source node, s, and a dummy sink node, t, as well as nodes for each flight

i, i.e. $V = \{i \in j : j \in J\} \cup \{s, t\}$. The arc set is defined as $A = A_f \bigcup A_0$, where A_f is the set of all arcs (i_p, i_s) such that flight $i_p \in I$ can be immediately followed by flight i_s (i.e. p stands for the predecessor and s stands for the successor), and $A_0 = \{(s, i)|i \in V\} \bigcup \{(i, t)|i \in V\}$.

In this network representation,

- the cost of traversing an arc (i_p, i_s) is $c_{i_p j} - u_{i_p}$, $\forall i_p \in V$ and $\exists i_s \in V$ where i_s can be assigned to the gate schedule right after i_p where $c_{i_p j}$ is the cost of assigning flight i_s in gate schedule j while u_{i_p} is the value of the dual variable for flight i_s (obtained from the solution of RMP);
- the cost of traversing an arc (s, i) is zero $\forall i \in V$;
- and the cost of traversing an arc (i, t) is $c_{ij} - u_i$ $\forall i \in V$.

We notice that a path in G from s to t corresponds to a collection of flights that can be simultaneously assigned to a gate schedule (by assigning them one after each other in the same order as they appear on the path). Therefore, all schedules $j \in J$ are represented by an $s - t$ path. Traversing this path by adding up the arc costs, we obtain the reduced cost of the column that represents the gate schedule. From equations (3.4) and (3.5), finding the schedule with the minimum reduced cost is equivalent to finding the shortest path from s to t in G. An example of the network representation is given in Figure 3.2.

There are 10 flights and 3 gates in this example problem instance. The colored arcs show a set of paths corresponding to a set of feasible schedules that make up a feasible solution to the overall problem. Note that each flight-node is included in exactly one of the s-t paths. This corresponds to a feasible solution where each flight is assigned to a gate. As the arcs are created between two flight-nodes, each s-t path corresponds to feasible gate schedule. We also note the large number of feasible arcs in such a small problem instance.

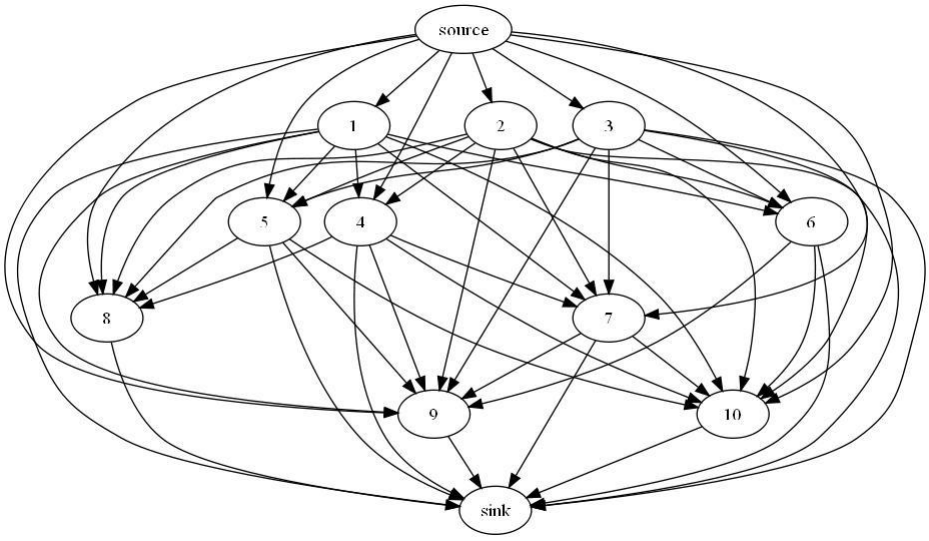

Figure 3.2: A small size flight network with ten flights and three gates

3.2 Column Generation Algorithm for Heterogeneous GAP

In the heterogeneous GAP, gates are not identical and each flight can be assigned to only a subset of available gates that can accommodate it. The proposed CG approach also works for the heterogeneous case; however the algorithm should be slightly modified.

In the set covering formulation, only one additional set of constraints will be inserted for every gate $k \in K$ for gate coverage. The formulation for heterogeneous case is:

$$\text{minimize} \quad \sum_{j \in J} c_j y_j \tag{3.7}$$

$$\text{subject to} \sum_{j \in J} b_{ij} y_j \geq 1 \qquad \forall i \in I \tag{3.8}$$

$$\sum_{j \in J_k} y_j \leq 1 \qquad \forall k \in K \tag{3.9}$$

$$y_j \in \{0, 1\} \qquad \forall j \in J \tag{3.10}$$

In this formulation, the objective function (3.7) minimizes the total cost of covering all flight by selected gate schedules while constraints (3.8) ensure that each flight

is covered by at least in one gate schedule. Constraints (3.9) ensure that only one schedule is chosen for each gate. Constraints (3.10) are the integrality constraints for the decision variables.

To formulate PSP of the heterogeneous case, let u_i and w_k be the dual variables corresponding to, respectively, constraint i in (3.8) and constraint k in (3.9). The dual of (3.7)-(3.10) can then be formulated as:

$$\max \quad \sum_{i \in I} u_i - \sum_{k \in K} w_k \tag{3.11}$$

$$\text{subject to} \sum_{i \in I} b_{ij} u_i + w_k \leq c_j \qquad \forall j \in J_k, \forall k \in K \tag{3.12}$$

$$u_i \geq 0 \qquad \forall i \in I \tag{3.13}$$

$$w_k \geq 0 \qquad \forall k \in K \tag{3.14}$$

Similar to the homogeneous case, PSP finds a column j that has a negative reduced cost (i.e. violates the corresponding dual constraint in (3.12)). Formally, when we search for the column/schedule that makes the best improvement in the objective function value of RMP, the PSP is to find the column

$$j^* = \arg \min_{j \in J_k, k \in K} \left\{ c_j - \sum_{i \in I} b_{ij} u_i + w_k \right\}. \tag{3.15}$$

If $c_{j^*} - \sum_{i \in I} b_{ij^*} u_i + w_k < 0$, then decision variable y_{j^*} is added to RMP. Otherwise, the optimal solution to RMP is reached.

Note that PSP in (3.15) can be solved in two stages:

- The first stage finds the schedule that has the most negative *short* reduced cost for each gate $k \in K$,

$$j_k^* = \min_{j \in J_k} \left\{ c_j - \sum_{i \in I} b_{ij} u_i \right\}, \forall k \in K.$$

- At the second stage, among the schedules with the most negative *short* reduced

cost found at the first stage for each gate $k \in K$, the schedule with the most negative reduced cost is found as

$$j^* = \min_{j \in \{j_1^*, j_2^*, \ldots, j_{|K|}^*\}} \left\{ c_j - \sum_{i \in I} b_{ij} u_i + w_k \right\}.$$

We note that the reduced cost of a gate schedule $j \in J_k$ is

$$c_j - \sum_{i \in I} b_{ij} u_i + w_k = \sum_{i \in I} b_{ij}(c_{ij} - u_i) + w_k \tag{3.16}$$

$$= \sum_{i \in j} (c_{ij} - u_i) + w_k. \tag{3.17}$$

It is easily observed that the reduced cost of a schedule in the heterogeneous case is slightly different than the one in the homogeneous case; it is also dependent on the gate to which the schedule belongs. This is due to the additional constraint (3.9) in RMP which introduces the corresponding dual variable w_k into the reduced cost calculation. Therefore, the difference between homogeneous and heterogeneous GAP is that, in the heterogeneous GAP we solve multiple (as many as the number of gates) PSPs instead of a single PSP and select the most negative reduced cost column among these multiple gate schedules considering also the dual values (w_k).

In terms of the network representation, we solve a shortest path problem for each gate $k \in K$ on the network $G_k = (V_k, A_k)$ which consists only those nodes and arcs that are compatible with gate $k \in K$. A gate schedule compatible with gate k is represented by a path from s_k to t_k in G_k. In Figure 3.3, we give the flowchart of the CG algorithm for the heterogeneous GAP. First, we solve the RMP with the additional constraint in (3.9). Then, we acquire dual values u_i corresponding to conflicting flights constraint in (3.8) and use the dual information by solving the PSPs with only compatible nodes at each gate $k \in K$. Among the gathered best gate schedules for all gates, we select the most negative reduced cost schedule using also the information from the dual values w_k, $k \in K$. We again check the optimality condition; we add the selected column to RMP if the reduced cost of the most negative reduced cost is less than zero. The procedure is verified until the optimality condition is verified.

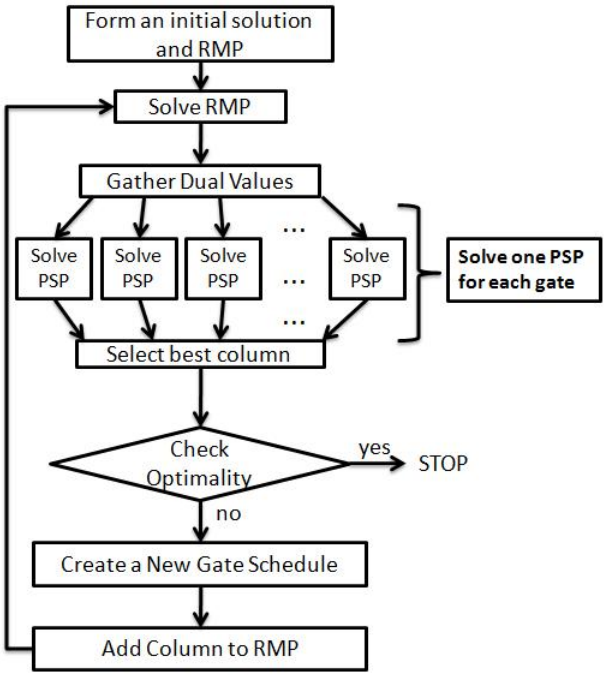

Figure 3.3: Flowchart of the CG algorithm for heterogeneous GAP

3.3 Objective Function and Robustness Measures

Deviations from the planned gate assignment are inevitable due to many disruptions that occur during the planning horizon. The gate assignment plan created by the models described so far do not consider the issue of operational robustness, i.e. how easy it is to recover from disruptions that occur during the operation. In this study, we explicitly incorporate robustness measures into the gate assignment model so that we can plan for a robust gate assignment in advance. We focus on independent flight delays (deviations from the planned arrival and departure times of flight) as the main source of disruption. Massive disruptions that affect the entire operation such as those caused by weather events are outside the scope of this study, since they often require a complete replanning of the gate assignment.

Unfortunately, it is a challenge to define a robustness measure that can predict how well a given gate assignment will recover from a disruption. Bolat [8] argues that minimizing the variance of gate idle times will result in a more robust gate assignment. In [2], he proposes five models that minimize the variance and range of

idle times in an effort to improve robustness.

Alternatively Dorndorf et al. [9] suggest a *robustness index* that measures the availability of recovery options built into the gate assignment plan. In this context, an easy-to-quantify measure is associated with insert opportunities that consider moving the delayed flight to a different gate. Figure 3.4 depicts an insert move opportunity for Flight 2 at Gate 2. Flight 2 is currently assigned to Gate 1, but it can be moved to Gate 2 since Gate 2 is idle from the arrival time of Flight 2 to the departure time of Flight 2. It is worth mentioning that the amount of excess idle time at Gate 2 available before the arrival time of Flight 2 and after the departure time of Flight 2 (marked in dashed lines) is an important factor in the viability of the insert move. Since an *insert* will only be considered if Flight 2 is either early or late, the insert move cannot be carried out unless there is excess idle time at Gate 2. A natural choice would be to use a robustness measure that is an increasing function of the excess idle time available for the move.

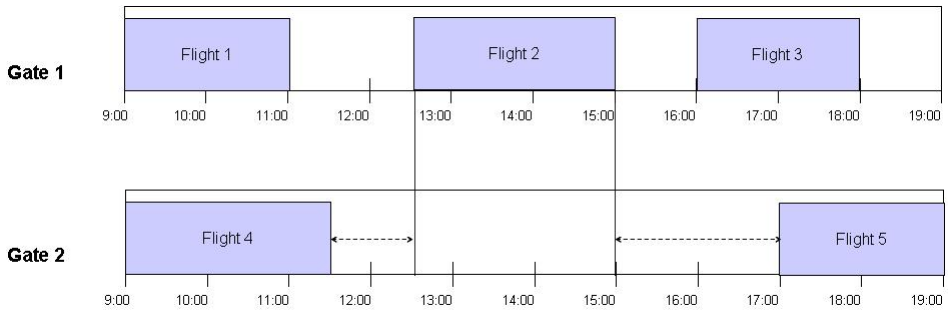

Figure 3.4: An illustration of feasible schedules for two gates

We can employ the gate assignment model given in (2.5)-(2.7) to introduce a robustness measure based on the insert opportunities present in a gate assignment. We propose an objective function that maximizes the sum of all insert move opportunities created by the gate assignment. For this purpose we define the objective function coefficient c_j for gate assignment j as follows:

$$c_j = -\sum_{(i_1,i_2)\in J_j} r(i_1,i_2), j \in J$$

where J_j is the ordered sequence of flights visited in gate schedule j (including the dummy source and sink nodes), and

$$r(i_1, i_2) = \sum_{i \in M_{i_1 i_2}} f(i_1, i, i_2),$$

here $M_{i_1 i_2}$ denotes the set of flights and $f(i_1, i, i_2)$ is an increasing function of the idle time resulting from inserting flight i between flights i_1 and i_2 that can be inserted between i_1 and i_2.

Considering the column generation method for the set-covering problem (2.5)-(2.7), insert moves can be easily incorporated in the evaluation of a new column being considered for addition to RMP. In the pricing problem, the objective function contribution of each arc (i_1, i_2) in the gate schedule j(i.e. r_{i_1,i_2}), can be computed as discussed above.

CHAPTER 4

COMPUTATIONAL STUDY

The goal of our computational study can be summarized as follows:

- to analyze the efficiency and effectiveness of the CG algorithm to solve the set covering formulation of GAP for both homogeneous and heterogeneous cases,
- to test if different types of objective functions, particularly in the case of alternative robustness measures, can be handled within the same generic algorithmic framework, and
- to understand the differences between the two robustness measures.

The computer programming for the experimental study is done in Visual C++. IBM ILOG CPLEX Optimization Studio 12.2 is used as the optimization software to solve LP and IP problems. The experiments are conducted on a PC with a 2.39 GHz Intel Core 2 Quad CPU Q6600 processor with 3.24 GB of RAM.

For homogeneous GAP, we use the data set in Bolat [12]. In this data set, there are 72 instances out of which there are,

- 24 small problems with 5 gates where the number of flights vary from 25 to 29,
- 24 medium problems with 10 gates where the number of flights vary from 50 to 56, and
- 24 large flights with 20 gates, where the number of flights vary from 102 to 112.

Each of these sets consist of 8 underutilized, 8 normally utilized and 8 highly utilized problem instances.

We use the homogeneous data from Bolat [12] to create the heterogeneous instances that are not available in the literature. For small instances, we consider small gates and medium gates: 5 flights are designated as medium-size flights and 2 out of 5 gates are designated as medium gates; the remaining flights are designated as small while the remaining gates are supposed to accommodate small flights only. For medium problem instances, where we have large, medium and small gates 7 flights are designated as medium-size while 3 flights are designated as large-size and 2 out of 10 gates are designated as medium gates while 2 out of 10 gates are designated as large gates; the remaining flights are designated as small while the remaining gates are supposed to accommodate small flights only. For large instances, 12 out of 50 flights are designated as medium-size flights while 4 out of 50 flights are designated as large-size flights and 4 out of 20 gates are designated as medium gates while 4 out of 20 gates are designated as large gates; the remaining flights are designated as small while the remaining gates are supposed to accommodate small flights only. For all instances we assume that all small and medium-size flights can be assigned to medium gates and small medium and large-size flights can be assigned to large-size gates, where small gates can only accommodate small-size flights. A summary of problem settings is given in Table 4.1.

Table 4.1: Characteristics of the homogeneous and heterogeneous problem instances.

			Heterogeneous					
	Homogeneous		Small		Medium		Large	
Problem Size	Flights	Gates	Flights	Gates	Flights	Gates	Flights	Gates
Small	25-29	5	20-24	3	5	2	0	0
Medium	50-56	10	40-46	6	7	2	3	2
Large	102-112	20	86-96	12	12	4	4	4

In order to evaluate different robustness measures, we also alter the original data sets. When we try to conduct insert moves with the original problem instances of Bolat [12], they allow only a limited number of insert moves and we are not able to evaluate this robustness measure with the original instances. Therefore, we reduced the flight durations by half for the robustness comparison runs, and used there

modified instances to compare the two robustness measures. The altered data for the insert capability is experimented also with variance of idle time measure (see Appendix C).

In this section, we provide the results of our computational experiments, we have performed to evaluate the efficiency and effectiveness of CG algorithm under different settings. We test the performance of the CG algorithm to solve the LP relaxation of the problem with respect to the CPU time and number of iterations to reach optimality. To obtain integer-feasible solutions, we use a heuristic idea (IP-ALL): once the LP relaxation solution is found, we solve RMP as an IP problem considering all the columns generated so far in addition to the ones in the initial solution.

CPU time is reported in milliseconds. We impose a limit on the number of iterations for homogeneous problems. The algorithm terminates at 300 iterations. This value is set due to an initial experimentation with the algorithm performance in order to limit CPU time to a reasonable level; results with unlimited CPU time is given in Appendices.

4.1 Results with Homogeneous Problems

The homogeneous problem set is used to test the performance of the CG algorithm with respect to two aspects of the algorithm: Number of columns generated at each iteration and initial solution. The traditional CG algorithm iterates by generating one column in each iteration while the initial RMP is constructed based on a feasible solution. We also test the performance of our IP heuristic, IP-ALL, by comparing it against the performance of another heuristic,which is also based on the column pool created by the CG algorithm. The results where the traditional CG setting is used along with our original IP heuristic, IP-ALL, given in Appendix A.

4.1.1 Column Generation Scheme

Our original CG algorithm uses the commonly used approach where only one column is added to the RMP at each iteration. Alternatively, more than one column can be added to RMP to achieve more improvement per iteration. Although we prefer to use a RMP as small as possible rather than including all feasible columns at once to ensure computational efficiency, we argue that adding a group of columns that complement each other to RMP at each iteration may improve the performance of the CG algorithm. To defend this argument, we modify our algorithm in such a way that we create a number of extra columns in addition to the column with the most negative reduced cost in each iteration, we resort to PSP more than once. After solving PSP once, the flights that belong to the newly generated gate schedule are deleted from the network and PSP is solved again to create additional columns with the flights which have not been selected for the gate schedules in the previous PSP solutions. This is repeated until all flights are covered by one of the new schedules. Therefore, at every iteration, we generate a selected number of new schedules that cover different flights and can be selected simultaneously to construct a new feasible solution. For each problem set, we experiment with different number of extra columns based on the number of gates in the problem. To determine the number of extra columns that yield the best performance, we conduct a parameter analysis where the number of extra columns vary from one to the number of gates in the problem. Table 4.2, Table 4.3 and Table 4.4 show the results of these experiments. These tables consist of six columns. The first column is the utilization level where "U", "N" and "H" correspond to underutilized, normally utilized and highly utilized instances, respectively. The column titled as "Extra columns" shows the number of extra columns generated at each iteration. The third column displays the number of iterations performed by the CG algorithm to reach the LP relaxation optimal solution. The "LP Time" column shows the average CPU time spent by the algorithm to reach the LP optimal solution and "IP Time" shows average CPU time incurred to reach an IP optimal solution within the columns generated so far until the LP optimal solution is obtained. The last column "Frac/Int" column displays

the number of fractional solutions vs number of integer feasible solutions obtained after LP optimal solution is reached by CG algorithm.We note that IP Time is the average for those instances whose LP optimal solution is not integer-feasible.

Table 4.2: Results for small problems with under/normal/high utilized homogeneous gates

Utilization	Extra Columns	Iterations	LP Time	IP Time	Frac/Int
U	0	88	2580	16	1/7
U	1	50	2535	12	0/8
U	2	35	2383	16	1/7
U	3	30	2619	4	0/8
U	4	28	3051	14	0/8
N	0	66	2729	18	0/8
N	1	36	2368	27	1/7
N	2	28	2482	13	0/8
N	3	24	2562	16	0/8
N	4	22	3194	25	1/7
H	0	29	1383	26	1/7
H	1	14	1039	24	1/7
H	2	16	1320	35	2/6
H	3	11	1463	27	1/7
H	4	10	1539	18	0/8

Table 4.2 shows the results for small problems with five gates. The LP time results indicate that creating extra columns does not result in a significant benefit for small problems. It can be argued that in small instances there is no need to increase the size of the column pool in each iteration. We also see that very few fractional solutions obtained by the optimal LP relaxation solution. The results indicate that as the utilization level increases, CPU time required to obtain LP solution tends to decrease. This may be due to the smaller size of the solution space for the highly utilized problems. The limited number of feasible solutions may lead us to the optimal solution faster. Another important observation is that the extra column generation is effective in decreasing the number of iterations to reach the optimal solution. Overall, we may conclude that the CG algorithm is capable to solve small size problems quite fast; small instances can be solved on average in at most three seconds.

The results in Table 4.3 are for medium-size problems. For medium problems, extra column generation improves the algorithm performance significantly. As the number of extra columns increases, the number of CG iterations required decreases sharply. Meanwhile as more columns are generated, the time per iteration increases.

Table 4.3: Results for medium problems with under/normal/high utilized homogeneous gates

Utilization	Extra Columns	Iterations	LP Time	IP Time	Frac/Int
U	0	263	16437	82	4/4
U	1	132	13340	72	3/5
U	2	89	12904	45	2/6
U	3	68	12957	98	4/4
U	4	60	13696	49	2/6
U	5	52	14232	39	1/7
U	6	47	15057	41	1/7
U	7	42	14928	55	2/7
U	8	41	16998	72	3/5
U	9	41	18926	39	1/7
N	0	225	28051	183	6/2
N	1	115	23031	131	4/3
N	2	79	21152	102	3/5
N	3	60	20395	118	4/4
N	4	52	21980	109	2/6
N	5	45	21217	84	2/6
N	6	39	22076	100	3/5
N	7	37	23605	86	2/6
N	8	36	26262	113	3/5
N	9	35	28438	143	5/3
H	0	151	12411	132	5/3
H	1	78	10596	152	4/4
H	2	55	10679	141	5/3
H	3	37	9217	65	1/7
H	4	28	8292	130	5/3
H	5	27	9967	72	3/5
H	6	23	9480	78	3/5
H	7	22	9623	60	2/6
H	8	21	10522	60	1/7
H	9	20	11466	78	2/6

2, 3 and 4 extra columns are the best settings for under utilized, normally utilized and highly utilized cases, respectively. For medium problems, the ratio of fractional solutions to integer solutions is higher when compared to small problems. We observe that the time it takes the IP heuristic to reach an optimal solution within the column pool increases significantly as the problem size increases.

From the results in Table 4.4 for large problems, we observe that with 8 and 12 extra column settings the CG algorithm provides the best performance in terms of the CPU time to obtain LP optimal solutions. We also observe that the ratio fractional LP relaxation solutions are even higher than that of the medium size problems and it takes longer to obtain IP optimal solutions within the given column pool. LP relaxation of large instances can be solved using the best settings in around 50 to 70 seconds depending on the utilization level. However we observe that in some settings (such as the one with four extra columns), the IP heuristic takes extremely

Table 4.4: Results for large problems with under/normal/high utilized homogeneous gates

Utilization	Extra Columns	Iterations	LP Time	IP Time	Frac/Int
U	0	300	29822	4469	8/0
U	4	135	53772	809	8/0
U	8	78	52844	287	6/2
U	12	56	53240	248	6/2
U	16	50	60035	217	6/2
U	19	46	66918	299	8/0
N	0	300	53274	10725	8/0
N	4	115	76500	1193367	8/0
N	8	68	76869	522	8/0
N	12	51	81525	340	7/1
N	16	44	89131	347	7/1
N	19	40	95750	383	8/0
H	0	300	66073	1797	6/2
H	4	88	67651	24240	6/2
H	8	53	69136	474	6/2
H	12	40	75927	352	5/3
H	16	33	80279	401	6/2
H	19	32	96383	370	6/2

long to reach the optimal solution within the given column pool.

For the homogeneous experiments with extra column generation scheme, the proposed CG algorithm performs well with small problems, where the CPU time to obtain a LP optimal solution is around 1 to 3 seconds on average. The algorithm also provides integer feasible solutions for most of the instances, without resorting to an IP heuristic. For medium problems, CPU time to obtain a LP optimal solution increases to 8 to 28 seconds and around half of the instances required to run the IP heuristic to find an integer-feasible solution. The number of iterations needed to reach the LP optimum also increases in comparison to the small problems. Fop large problems, the CPU time increases further as expected, it varies between 52 to 96 seconds. The algorithm needs to execute more iterations to reach the LP optimal solution, and the 300-iteration limit is reached for all large instances if no extra columns are generated(see Appendix A for unlimited results). Moreover in almost every instance, IP heuristic is employed to obtain an integer feasible solution using the generated column pool. In terms of the utilization levels we see that for small and medium problems normal utilization level takes more time. The performance of IP heuristic is discussed in Section 4.1.3.

4.1.2 Initial Solution Strategy

To initiate the CG algorithm, we use a feasible initial solution obtained from a constructive heuristic as discussed in Section 3.1.1. To eliminate the need for a feasible initial solution, we also experiment with infeasible initial solutions. We construct an initial solution that contains as many schedules as the number of flights in the problem, where each schedule contains only one flight. Therefore, the initial solution is infeasible with respect to availability of gates. In order to understand the effect of the initial solution quality, we conduct the same experiments by initiating the CG algorithm with such infeasible initial solution. The experimental results shown in Tables 4.5, 4.6 and 4.7 are obtained using underutilized instances for small, medium and large problems, respectively.

Table 4.5: Results for small problems with underutilized homogeneous gates when CG algorithm is initiated with an infeasible solution

Extra Columns	Iterations	LP Time	IP Time	Frac /Int
0	107	3074	12	0/8
1	58	2760	10	0/8
2	44	3018	12	0/8
3	38	3333	9	0/8
4	50	4047	26	1/7

Table 4.6: Results for medium problems with underutilized homogeneous gates when CG algorithm is initiated with an infeasible solution

Extra Columns	Iterations	LP Time	IP Time	Frac /Int
0	282	16844	163	5/3
1	143	14272	98	3/5
2	118	12617	18	0/8
3	76	12770	19	0/8
4	63	12936	51	2/6
5	54	13061	33	1/7
6	50	13994	43	2/5
7	47	14825	27	1/7
8	45	16100	33	1/7
9	46	18416	51	2/6

As we observe from the results in Tables 4.5, 4.6 and 4.7, starting with an infeasible solution may affect both the CPU time and number of iterations to reach the LP relaxation optimal solution. For small problems, the difference is significant while it is not the case for medium and large problems. We may argue that a good initial solution helps the algorithm to reach the LP optimal more quickly in general,

Table 4.7: Results for large problems with underutilized homogeneous gates when CG algorithm is initiated with an infeasible solution

Extra Columns	Iterations	LP Time	IP Time	Frac /Int
0	300	31715	519436	8/0
4	138	60318	3201	7/1
8	78	58734	2836	8/0
12	58	57346	391	8/0
16	51	66035	336	8/0
19	50	77074	508	7/1

but for large problems a good initial solution is not as effective as it is for small problems. This can be attributed due to the ability of the constructive heuristic to obtain near-optimal results in smaller problems. As the constructive heuristic does not provide good-quality solutions for larger problems, a feasible initial solution might be as poor as an infeasible solution to initiate the CG algorithm. Based on these results, we may conclude that infeasible solutions might be used to initiate the algorithm in the absence of feasible solutions.

4.1.3 IP Heuristics

In this section, we report the computational results for the heuristic methods we have employed to obtain integer feasible solutions. The CG algorithm obtains an optimal solution to the LP relaxation of the problem. It can create fractional solutions as Bolat [2] also discusses. In our first set of experiments, in Tables 4.2 4.3 and 4.4, we utilize the IP-ALL heuristic to obtain an integer feasible solution when the CG algorithm terminates with a fractional solution. All of the previous results shown in Tables 4.2 to 4.7 use this heuristic if necessary. As an alternative heuristic method, we propose IP-NZ, which solves an IP problem that includes only the columns whose corresponding decision variables are nonzero in the optimal LP solution. With this alternative method, we intend to further decrease the CPU time to obtain an integer feasible solution without sacrificing the quality.

To test the integer feasible solution quality of the CG algorithm with IP-ALL and IP-NZ heuristics, we conduct experiments in order to compare the objective function values of the final integer feasible solutions with that of the genetic algorithm in Bolat [12]. As a result of our experiment in Section 4.1 we generate 4 extra columns

for small and medium problems, and 12 for large problems. The results are given in Tables 4.8 4.9 and 4.10. The first column provides information on the problem instance. The second column shows the initial objective function value (OFV). The third column shows the OFV from Bolat [12]. The next three columns show the results of the experiments, with IP-ALL heuristic where OFV of the solution found by the heuristic, IP time shows the CPU time for the integer programming runs. and (Gap%) shows the percentage gap from the OFV in Bolat [12]. The next three columns show the same information for the IP-NZ heuristic.

Table 4.8: Comparison of results for the IP heuristics on small homogeneous problems with under, normal and high utilization

Pr. Set	Init. Sol.	Bolat OFV	IP-ALL			IP-NZ		
			OFV	IP Time	Gap(%)	OFV	IP Time	Gap(%)
SU1	25560	21208	21208	15	0.000	21208	0	0.000
SU2	19887	19561	19561	0	0.000	19561	0	0.000
SU3	32365	20881	20881	0	0.000	20881	0	0.000
SU4	50926	20834	20834	16	0.000	20834	16	0.000
SU5	33196	20700	20700	16	0.000	20700	16	0.000
SU6	26951	21427	21427	15	0.000	21427	16	0.000
SU7	24200	20940	20940	16	0.000	20940	15	0.000
SU8	21351	20253	20253	15	0.000	20253	15	0.000
SN1	4787	4165	4165	16	0.000	4165	15	0.000
SN2	3886	3490	3490	15	0.000	3490	16	0.000
SN3	5671	4105	4105	16	0.000	4105	16	0.000
SN4	4152	3548	3548	16	0.000	3548	16	0.000
SN5	4705	3815	3817	94	0.052	4705	16	23.329
SN6	3722	3374	3374	16	0.000	3374	15	0.000
SN7	5647	4251	4251	15	0.000	4251	0	0.000
SN8	3741	3189	3189	16	0.000	3189	15	0.000
SH1	369	305	305	16	0.000	305	15	0.000
SH2	456	392	392	16	0.000	392	16	0.000
SH3	322	320	320	16	0.000	320	16	0.000
SH4	367	315	315	16	0.000	315	15	0.000
SH5	380	318	318	15	0.000	318	16	0.000
SH6	418	334	334	16	0.000	334	15	0.000
SH7	288	252	252	16	0.000	252	16	0.000
SH8	415	357	357	31	0.000	357	16	0.000

The results in Tables 4.8, 4.9 and 4.10 show us that IP-ALL heuristic is needed mostly for large and medium problems. In small problems, LP optimal solution is also an integer feasible solution. With small problems, both heuristics perform as good as the genetic algorithm in Bolat [12] except for one instance. With medium problems, IP-ALL finds the same OFV as Bolat [12] except for two instances while it is close within less than 0.1% gap. However, IP-NZ fails to perform as good as IP-ALL. For larger problems, IP-ALL finds better solutions in 3 out of 24 instances, same quality solution in 11 out of 24 instances and solutions with 1% gap for the

Table 4.9: Comparison of results for the IP heuristics on medium homogeneous problems with under, normal and high utilization

Pr. Set	Init. Sol.	Bolat OFV	IP-ALL			IP-NZ		
			OFV	IP Time	Gap(%)	OFV	IP Time	Gap(%)
MU1	54839	41233	41233	0	0.000	41233	16	0.000
MU2	89450	41000	41000	93	0.000	89450	32	118.171
MU3	63367	40953	40953	16	0.000	40953	15	0.000
MU4	80497	41203	41203	31	0.000	41203	16	0.000
MU5	76760	39218	39218	15	0.000	39218	16	0.000
MU6	58672	41244	41244	31	0.000	41244	31	0.000
MU7	60857	38975	38975	172	0.000	60857	63	56.144
MU8	69376	41112	41112	31	0.000	41112	31	0.000
MN1	17098	7988	7988	47	0.000	7988	46	0.000
MN2	11080	7538	7538	47	0.000	7538	47	0.000
MN3	8504	7054	7054	47	0.000	7054	47	0.000
MN4	9151	7811	7811	62	0.000	7811	47	0.000
MN5	8997	7043	7045	282	0.028	8997	109	27.744
MN6	12034	7460	7460	47	0.000	7460	47	0.000
MN7	14028	6988	6994	234	0.086	14028	125	100.744
MN8	10334	7710	7710	62	0.000	10334	141	34.034
MH1	482	380	380	188	0.000	482	125	26.842
MH2	627	549	549	250	0.000	627	125	14.208
MH3	828	650	650	250	0.000	828	140	27.385
MH5	696	554	554	94	0.000	696	141	25.632
MH6	802	582	582	16	0.000	582	62	0.000
MH7	913	667	667	94	0.000	913	157	36.882
MH8	667	525	525	16	0.000	525	62	0.000

Table 4.10: Comparison of results for the IP heuristics on large homogeneous problems with under, normal and high utilization

Pr. Set	Init. Sol.	Bolat OFV	IP-ALL			IP-NZ		
			OFV	IP Time	Gap(%)	OFV	IP Time	Gap(%)
LU1	189754	83466	83466	16	0.000	83466	16	0.000
LU2	192322	78274	78274	15	0.000	78274	15	0.000
LU3	200206	84494	84492	172	-0.002	200206	47	136.947
LU4	236776	81332	81342	453	0.012	236776	63	191.123
LU5	188494	83276	83274	312	-0.002	188494	47	126.349
LU6	179560	78760	78766	500	0.008	179560	78	127.984
LU7	151227	81501	81501	219	0.000	151227	79	85.552
LU8	220509	80723	80721	297	-0.002	220509	94	173.167
LN1	34818	14344	14344	313	0.000	34818	94	142.736
LN2	24179	15605	15605	328	0.000	24179	110	54.944
LN3	20840	13736	13738	391	0.015	20840	109	51.718
LN4	19794	14188	14190	312	0.014	19794	234	39.512
LN5	33264	14482	14482	47	0.000	14482	78	0.000
LN6	29965	14271	14277	547	0.042	29965	188	109.971
LN7	33427	14879	14880	344	0.007	33427	156	124.659
LN8	27016	15040	15042	438	0.013	27016	141	79.628
LH1	1601	1111	1111	234	0.000	1601	172	44.104
LH2	2741	1059	1059	94	0.000	1059	93	0.000
LH3	1981	961	961	266	0.000	1981	203	106.139
LH4	1972	1076	1087	656	1.022	1972	171	83.271
LH7	2849	1185	1188	453	0.253	2849	218	140.422
LH8	2362	1113	1115	407	0.180	2363	203	112.309

remaining 10 instances. In general IP-NZ does not perform well when the problems are larger. In most of the instances of medium and large problems, IP-NZ cannot find a better solution than the initial feasible solution. Therefore, we may easily conclude that IP-ALL is a good heuristic that finds close-to-optimal solutions in

reasonable time.

4.2 Experimental Results with Heterogeneous Problems

Modifying the flight arrival and departure data in the problem instances in Bolat [12], we have generated problem instances with heterogeneous gates as discussed in Section 4. We have arranged the new instances in such a way that the initial feasible solutions for the homogeneous problems are also feasible with the new heterogeneous instances. In most cases, we have also realized that the best solution we have found for the homogeneous case is also feasible for the heterogeneous case. Therefore, we guarantee that there exists another integer feasible solution better than the initial one. One should also note that a heterogeneous feasible solution is always feasible for the corresponding homogeneous problem. In case the optimal solution for the homogeneous problem is feasible for the heterogeneous one, it is also optimal for the heterogeneous problem. We conduct the computational experiments under the same settings we employ for the homogeneous problems initially. The results in Appendix B in Tables B.1, B.2, B.3 are obtained by initiating the CG algorithm with the same feasible solution as the one for the homogeneous problem and using the IP-ALL heuristic to solve the IP problem. No extra column generation scheme is used for these experiments.

Table 4.11 compares the CPU time and the number of iterations for the homogeneous and heterogeneous problems. The problem set in the first column shows the problem size (S = small, M = medium, L = large) and the utilization level (U = underutilized, N = normally utilized, H = highly utilized). The second, third and fourth columns show the average CPU time to obtain LP relaxation optimal solutions, CPU time to obtain IP-feasible solutions from the LP optimal solutions and number of CG algorithm iterations for the homogeneous problems. The fifth, sixth and seventh columns show the same information for the heterogeneous problems. The last three columns provide information on the ratio of CPU times and number

of iterations, for homogeneous and heterogeneous problems.

Table 4.11: Comparison of homogeneous and heterogeneous model results in terms of CPU time and number of iterations

	Homogeneous			Heterogeneous			Ratio (HT/HM)		
Pr. Set	LP Time	IP Time	Iterations	LP Time	IP Time	Iterations	LP Time	IP Time	Iterations
SU	1951	14	88	11896	68	119	6.10	4.96	1.35
SN	1625	8	66	9774	61	91	6.01	7.68	1.38
SH	732	14	28	3832	29	38	5.23	2.16	1.34
MU	10674	43	267	117973	1621	361	11.05	37.48	1.35
MN	10633	105	225	105596	1000	305	9.93	9.49	1.35
MH	8114	98	151	66513	306	214	8.20	3.12	1.42
LU	75807	7389	228	1315232	14269889	973	17.35	1931.36	4.27
LN	72576	687666	324	1000373	2602020	767	13.78	3.78	2.37
LH	66945	53438	400	770857	33518	592	11.51	0.63	1.48

As expected, CPU times are in general significantly larger for the heterogeneous problems. The ratios of both CPU time and number of iterations are quite large and they also increase with problem size while the CPU time to obtain LP optimal solutions are 5 to 6 times longer for the small problems. For medium problems the ratio is around 8 to 11 and for large problems, it is 11 to 17. This increase can be explained by investigating the flow charts (see Section 3, Figures 3.1 and 3.3) of the proposed algorithms. We solve as many PSP's as the number of gates for the heterogeneous problems at each iteration of the heterogeneous algorithm. The LP time ratios almost reflect this for small and medium-size problems. For the highly utilized instances, we obtain more reasonable solution times for the heterogeneous problems. While the CPU time to obtain LP optimal solutions depend mostly the problem size, the performance to obtain IP solutions also changes according to the utilization level. For underutilized problems of medium or large size, the column generation algorithm needs more time, because the number of feasible columns are too many. Particularly for large, underutilized problems, the number of iterations to obtain LP optimal solutions are significantly larger than medium underutilized problems. Also the number of generated columns and in parallel to that, the size of the IP problem is larger for the large problems.

4.3 Alternative Robustness Measures: Variance of Idle Times vs. Insert Capability

In Section 3.3, we mention the robustness measures for GAP in the literature: variance of idle time measure introduced by Bolat [8] and insert capability measure introduced by Dorndorf et al. [9]. The computational results presented in the previous sections utilize the variance of idle time as the robustness measure. However using our CG algorithm, we can easily switch to a new robustness measure by only altering the reduced cost calculation while solving the PSP.

To introduce insert capability as a robustness measure, we experiment with different robustness score to evaluate the possible insert moves as discussed in Dorndorf et al. [9]. We first define these scores and then present the computational results obtained with each score.

4.3.1 New Robustness Measures to Represent Insertion Capability

Various robustness scores may be suggested to quantify the insert capability of a schedule. Note that our objective here is to maximize the robustness measures that represent insert capability whereas we seek to minimize the variance of idle time in all of the work discussed so far. We utilize simple scores to quantify the insert capability of a gate schedule. To explain the newly introduced scores, Figures 4.1 and 4.2 display examples of full insert and partial insert moves. In Figure 4.1 flight f is inserted between flights i and j, resulting in two idle times t_1 and t_2, where $t_1 = a_f - d_i$ and $t_2 = a_j - d_f$. On the other hand in Figure 4.2, flight f can only be partially accommodated between flights i and j, which creates only one idle time denoted by t_1, where t_1 is either $t_1 = a_f - d_i$ or $t_1 = a_j - d_f$.

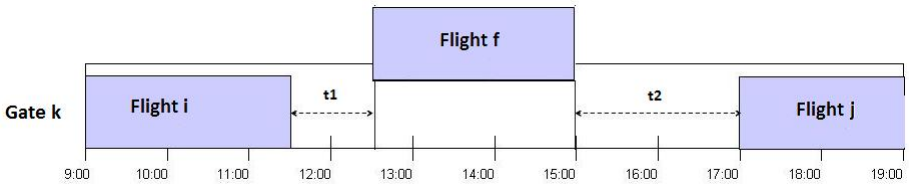

Figure 4.1: Depiction of a full insert move for flight f between flights i and j

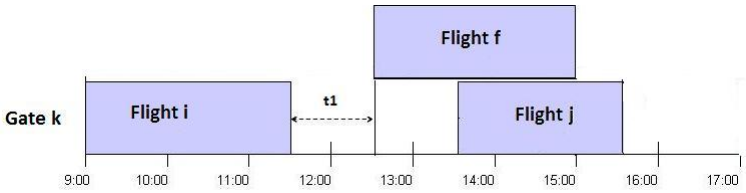

Figure 4.2: Depiction of a partial insert move for flight f between flights i and j

The simplest robustness score, RS^1, sums up over all flights f that can be fully inserted between two flights i and j. In the network representation we employ to solve the PSP, there exists an arc (i, j) for any two flights i and j that can be assigned to the same gate schedule consecutively. To adopt the CG algorithm to maximize insert capability robustness measure, we define the cost of arc (i, j) as the robustness score calculated for the pair of flights i and j.

According to the illustration in Figure 4.1, RS^1 is

$$RS^1_{ij} = \sum_{f:\, a_f \geq d_i,\, d_f \leq a_j} (t1 + t2) \tag{4.1}$$

The second score, RS^2, is a modification of RS^1: we divide RS^1 by the number of flights (denoted by $|F|$) that can be inserted between the two flights, i and j, to obtain an average idle time per flight.

$$RS^2_{ij} = \sum_{f:\, a_f \geq d_i,\, d_f \leq a_j} (t1 + t2)/|F| \tag{4.2}$$

The third score, RS^3, also accounts for the partial insert moves, as illustrated in Figure 4.2 where the ground time of the inserted flight overlaps partially with the time available between the two flights i and j. Then,

$$RS^3_{ij} = \sum_{f:\, a_f \geq d_i \text{ or } d_f \leq a_j} t1 \; + \sum_{f:\, a_f \geq d_i,\, d_f \leq a_j} t1 + t2 \tag{4.3}$$

The fourth score, RS^4_{ij}, is similar to RS^2_{ij} in spirit. We again calculate an average idle time by dividing RS^3_{ij} by the number of flights that can be inserted fully or

partially between the selected flights.

$$RS_{ij}^{4} = \left(\sum_{f:\, a_f \geq d_i \text{ or } d_f \leq a_j} t1 \; + \sum_{f:\, a_f \geq d_i,\, d_f \leq a_j} t1 + t2 \right) / |F| \tag{4.4}$$

All of the previously described scores lead the algorithm to create arcs where an insert move is more likely to be accomplished and this favors schedules with long idle times between two consecutive flights (and also very short idle times between other consecutive flights). Therefore these four robustness measures create schedules with highly variable idle time values, which is in the opposite direction of Bolat [8]'s approach that minimizes idle time variance. To overcome this issue, we propose RS_{ij}^{5}, which only accounts for idle times that are less than the average idle time in the schedule. In RS_{ij}^{5}, we first calculate the length of the planning horizon and multiply it with the number of gates. This value is the total time available for scheduling flights. We also calculate the total ground time of all flights. Then we find the total idle time, by subtracting the total ground time from the total time available for scheduling. Finally, we divide the total idle time by the number of flights in the problem, to obtain the average idle time per flight denoted by IT. In the calculation of RS_{ij}^{5}, the idle times of insert moves are included only if they are less than or equal to the average idle time. RS_{ij}^{5} aims to prevent the algorithm from creating columns, with more than the average idle time between consecutive flights.

$$RS_{ij}^{5} = \left(\sum_{(f:\, a_f \geq d_i \text{ or } d_f \leq a_j),\, (t_1 \leq IT)} t1 \; + \sum_{(f:\, a_f \geq d_i,\, d_f \leq a_j),\, (t_1 + t_2 \leq IT)} t1 + t2 \right) / |F| \tag{4.5}$$

4.3.2 Comparison of Variance Of Idle Time and Insert Capability Robustness Measures

With all of the scores discussed above, we are able to reach LP optimal solutions using our algorithm. But the columns generated by the CG algorithm are not sufficient to form an IP feasible solution. The reason of this outcome can be identified by

looking at the columns generated using the robustness scores as we explain below. Figures 4.3 and 4.4 also give us some insight into this outcome. These two figures display the OFV throughout the algorithm for RS_{ij}^5 and variance of idle time robustness measures, respectively. In Figure 4.3, the algorithm uses RS_{ij}^5 robustness measure to reach a LP optimal solution after some iterations. But in the last step, the algorithm searches for an IP solution in the resulting column pool and generates to the initial feasible solution as the final IP solution. Therefore the CG algorithm cannot improve upon the initial IP solution for this robustness measure. Conversely, the algorithm is capable of generating an IP solution that has a much better OFV for the variance idle time robustness measure as depicted in Figure 4.4.

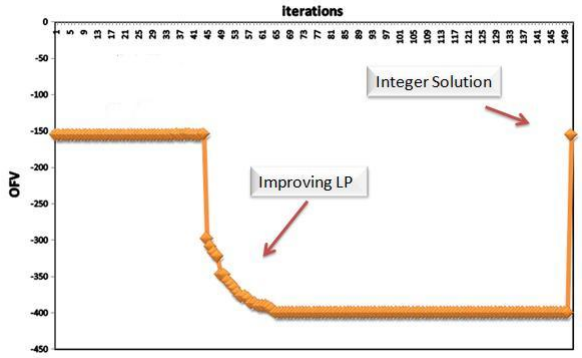

Figure 4.3: A sample run using RS^5 score structure

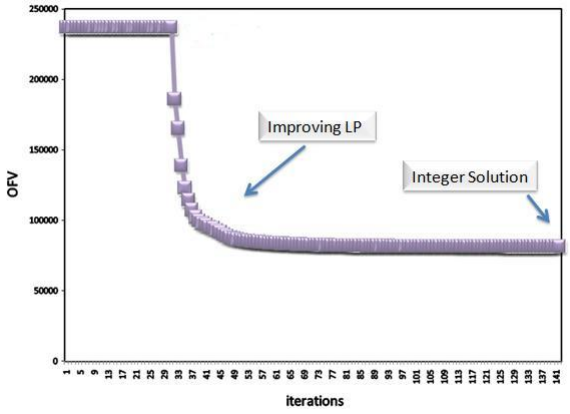

Figure 4.4: A sample run using the variance of idle time cost structure

Using the variance of idle time measure, flights are evenly distributed on the planning horizon for each gate schedule, as expected. Often, the optimal solutions

of the instances provided by Bolat [12] consist of evenly distributed number of flights across the gates. On the other hand the columns created by the insert capability robustness measure have very long or very short idle times between consecutive flights. As the insert capability of a solution increases, the variance of idle time between two consecutive flights also increases for a solution, which causes flights to be distributed unevenly. Our algorithm finds an LP optimal solution, but the resulting columns are not likely to form an integer feasible solution due to this reason. Therefore, the IP solution remains the same as the initial feasible solution even for the small instances.

This observation is also helpful for the comparison of the two suggested methods. While Bolat [2] suggests distributing the flights evenly with similar idle times between flights, Dorndorf et al. [9] proposes to increase the insert capability of the columns by forming some very large idle times in some columns and allowing more insert moves at desired time intervals. So, we can argue that the columns created using these two different robustness measures have fundamentally different structures.

To check whether there is any correlation between these two measures, we compare the robustness measures in terms of solution quality using the following approach: First, we run the algorithm on one test instance for each different problem set (9 instances in total) using the variance of idle time measure and calculate the OFV of the initial and final solution for all of the six robustness measures. Table 4.12 displays the percentage improvement in terms of the OFV from the initial solution to the final solution for all robustness measures. We see that the algorithm improves the OFV of the variance of idle time robustness measure for all test instances, while the OFV's of all other measures deteriorate (except for RS^5 for SN1, and RS^1 for small instances, where the most simple structure is used). So it can be argued that there is a negative correlation between the variance of idle time and insert capability robustness measures.

Table 4.12: Comparison of the variance of idle time and insert capability robustness measures on homogeneous problem instances

Problem Set	Variance of idle time	RS^1	RS^2	RS^3	RS^4	RS^5
DSU1	0.07	0.16	-0.10	-0.21	-0.12	-0.05
DSN1	0.01	0.88	-0.01	-0.01	0.00	0.01
DSH1	0.00	0.66	-0.01	-0.03	-0.17	-0.01
DMU1	0.11	-0.25	-0.15	-0.16	-0.12	-0.11
DMN1	0.15	-0.04	-0.18	-0.22	-0.13	-0.11
DMH1	0.00	0.11	0.00	0.00	0.00	-0.01
DLU1	0.34	-0.56	-0.40	-0.46	-0.37	-0.29
DLN1	0.17	-0.50	-0.22	-0.29	-0.13	-0.20
DLH1	0.01	-0.13	-0.03	-0.03	-0.09	-0.02

CHAPTER 5

CONCLUSION AND FUTURE WORK

In this study, RGAP is formulated as a set covering problem. A column generation algorithm proposed to solve the RGAP and the PSP in this CG algorithm is modeled as a network and solved by a shortest path algorithm. We formulate the objective function of the RGAP using the robustness measures in the literature: variance of idle time and insert capability.

Algorithms proposed for homogeneous and heterogeneous problems provides LP optimal solutions for all instances from Bolat [12], and in many cases they are also IP optimal solutions. Computational results indicate that the proposed CG algorithm performs better for highly utilized instances on both homogeneous and heterogeneous problems. Initializing the algorithm with a better initial solution does not have a large impact on the CPU time and we also observe that infeasible initial solutions can be used for homogeneous problems. We show that CPU time can be improved by using smart column generation approaches such as generating extra columns using multiple PSP runs with a more limited network, as we did in this study. For homogeneous and heterogeneous problems, CPU times are correlated with the size of the problem, especially number of gates plays an important role.

We demonstrate that the algorithm can be easily modified according to other robustness measures, such as the insert capability robustness measure implemented in this study. To modify the algorithm for this measure, only the cost calculation phase in the PSP formulation needs to be rearranged according to the the new objective function structure. While the algorithm is capable of obtaining the LP optimal solutions, for this robustness measure, it cannot improve on the initial IP

solution. A comparative analysis of the two robustness measures indicates that they are negatively correlated and each measure creates different columns in terms of idle time characteristics.

The contributions of this study can be summarized as follows:

- We propose a CG algorithm that can solve both homogeneous and heterogeneous RGAP in reasonable CPU time to obtain near IP optimal solution quality.
- The proposed algorithm generates better quality integer solutions than the genetic algorithm of Bolat [12] for some homogeneous gate instances.
- We suggest heuristics that employ extra column generation to improve the CG algorithm in terms of CPU time and solution quality.
- We show that the two robustness measures proposed in the literature, variance of idle time and insert capability, are negatively correlated.

Several extensions can be suggested as future work.

- PSP performance can be improved using more sophisticated network algorithms and other problem formulations.
- Other information regarding the neighboring flights in a schedule such as the company information or flight type can be also considered in the score calculation rather than only focusing on idle time.
- IP performance can be improved using IP heuristic methods, which can lead to substantial reduction in the CPU times of the large-size problems.
- Other robustness measures can be easily implemented using the CG approach, by only altering the score calculation procedures.

Due to the increased traffic and congestion at airports, methods similar to the proposed algorithm started to be used in real life applications. The performance of the formulated CG algorithm indicates that, CG algorithms, supported with some

heuristic performance improvement methods, can suggest good quality solutions in reasonable times to the operational problems in the airline industry.

Bibliography

[1] U. Dorndorf, A. Drexl, Y. Nikulin, and E. Pesch, "Flight gate scheduling: state-of-the-art and recent developments," *Omega*, vol. 35, pp. 326–334, 2007.

[2] A. Bolat, "Assigning arriving aircraft flights at an airport to available gates," *Journal of the Operational Research Society*, vol. 50, pp. 23–34, 1999.

[3] M. Şeker, "Stochastic airport gate assignment problem," Master's thesis, Sabancı University, 2010.

[4] O. Babic, C. Teodorovic, and V. Tosic, "Aircraft stand assignment to minimize walking," *Journal of Transportation Engineering*, vol. 110, pp. 55–66, 1984.

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

[6] J. Xu and G. Bailey, "The airport gate assignment problem: mathematical model and a tabu search algorithm," *Proceedings of the 34th Hawaii International Conference on System Sciences*, 2001.

[7] H. Ding, A. Lim, B. Rodrigues, and Y. Zhu, "New heuristics for over-constrained flight to gate assignments," *Journal of the Operational Research Society*, vol. 55, pp. 760–768, 2004.

[8] A. Bolat, "Procedures for providing robust gate assignments for arriving aircrafts," *European Journal of Operational Research*, vol. 120, pp. 63–80, 2000.

[9] U. Dorndorf, F. Jaehn, C. Lin, H. Ma, and E. Pesch, "Disruption management in flight gate scheduling," *Statistica Neerlandica*, vol. 61, pp. 92–114, 2007.

[10] A. Lim and F. Wang, "Robust airport gate assignment," *Proceedings of the 17th IEEE International Conference on Tools with Articial Intelligence*, 2005.

[11] D. G., D. J., and M. M. Solomon, *Column Generation*. Springer Science Business Media Inc., 2005.

[12] A. Bolat, "Models and a genetic algorithm for static aircraft-gate assignment problem," *Journal of the Operational Research Society*, vol. 52, pp. 1107–1120, 2001.

Appendix A

Homogeneous Run Results

Table A.1: Results for small problems with under/normal/high utilized homogeneous gates when exta column generation is not considered

Data Set	Iterations	LP Time	IP Time	frac = 1	Total Time
DSU1	77	1750	0	0	1750
DSU2	58	1188	0	0	1188
DSU3	93	1953	16	0	1969
DSU4	114	2484	63	1	2547
DSU5	97	2109	16	0	2125
DSU6	103	2328	15	0	2343
DSU7	79	1844	0	0	1844
DSU8	80	1953	0	0	1953
DSN1	54	1329	0	0	1329
DSN2	54	1250	0	0	1250
DSN3	79	1922	15	0	1937
DSN4	69	1797	0	0	1797
DSN5	68	1656	16	0	1672
DSN6	51	1250	0	0	1250
DSN7	73	1781	16	0	1797
DSN8	82	2015	16	0	2031
DSH1	33	812	0	0	812
DSH2	31	797	16	0	813
DSH3	11	328	16	0	344
DSH4	12	312	0	0	312
DSH5	35	875	0	0	875
DSH6	53	1375	31	1	1406
DSH7	10	266	15	0	281
DSH8	40	1094	31	1	1125

Table A.2: Results for medium problems with under/normal/high utilized homogeneous gates when exta column generation is not considered

Data Set	Iterations	LP Time	IP Time	frac = 1	Total Time
DMU1	229	8187	157	1	8344
DMU2	279	10453	16	0	10469
DMU3	259	10531	16	0	10547
DMU4	322	12469	0	0	12469
DMU5	311	12891	15	0	12906
DMU6	248	10531	16	0	10547
DMU7	249	10187	110	1	10297
DMU8	241	10140	16	0	10156
DMN1	237	10688	109	1	10797
DMN2	216	10484	109	1	10593
DMN3	203	9297	94	1	9391
DMN4	229	10390	16	0	10406
DMN5	205	9469	15	0	9484
DMN6	264	12875	266	1	13141
DMN7	251	12219	109	1	12328
DMN8	194	9641	125	1	9766
DMH1	139	7703	125	1	7828
DMH2	91	4985	78	1	5063
DMH3	160	7907	15	0	7922
DMH5	172	8812	110	1	8922
DMH6	159	8547	218	1	8765
DMH7	172	9954	125	1	10079
DMH8	164	8891	15	0	8906

Table A.3: Results for large problems with under/normal/high utilized homogeneous gates when exta column generation is not considered

Data Set	Iterations	LP Time	IP Time	frac = 1	Total Time
DLU1	679	64782	343	1	65125
DLU2	765	80953	750	1	81703
DLU3	688	75203	344	1	75547
DLU4	758	86375	22359	1	108734
DLU5	644	74969	3031	1	78000
DLU6	642	76094	27281	1	103375
DLU7	619	64625	1031	1	65656
DLU8	716	83453	3969	1	87422
DLN1	597	71922	969	1	72891
DLN2	551	65031	824875	1	889906
DLN3	543	66796	20969	1	87765
DLN4	604	77484	3292281	1	3369765
DLN5	548	73281	4157	1	77438
DLN6	547	76578	3563	1	80141
DLN7	519	72625	1354015	1	1426640
DLN8	554	76891	500	1	77391
DLH1	404	56281	20875	1	77156
DLH2	452	64984	2672	1	67656
DLH3	478	72937	96578	1	169515
DLH4	442	68641	36375	1	105016
DLH7	487	72468	95797	1	168265
DLH8	458	66359	68328	1	134687

Table A.4: Results for small problems with under utilized homogeneous gates when exta column generation is considered

Pr. Set	Extra Columns	Iterations	LP Time	IP Time	Frac=1
DSU1	0	77	1781	0	0
DSU1	1	49	2000	0	0
DSU1	2	31	1813	15	0
DSU1	3	25	1907	15	0
DSU1	4	28	2485	15	0
DSU1	5	28	2609	0	0
DSU2	0	58	1437	0	0
DSU2	1	30	1313	15	0
DSU2	2	26	1453	94	1
DSU2	3	26	1969	0	0
DSU2	4	20	1829	15	0
DSU2	5	20	1875	0	0
DSU3	0	93	2829	15	0
DSU3	1	51	2500	16	0
DSU3	2	33	1953	0	0
DSU3	3	31	2437	0	0
DSU3	4	26	2547	0	0
DSU3	5	26	2562	0	0
DSU4	0	114	3000	78	1
DSU4	1	66	3047	0	0
DSU4	2	44	2953	0	0
DSU4	3	43	3563	0	0
DSU4	4	35	4281	16	0
DSU4	5	35	3875	0	0
DSU5	0	97	2812	0	0
DSU5	1	58	3312	16	0
DSU5	2	36	2719	0	0
DSU5	3	29	2719	0	0
DSU5	4	33	3640	16	0
DSU5	5	33	3937	0	0
DSU6	0	103	3406	16	0
DSU6	1	56	3234	16	0
DSU6	2	38	2828	0	0
DSU6	3	33	3157	0	0
DSU6	4	29	3500	15	0
DSU6	5	29	3594	0	0
DSU7	0	79	2656	0	0
DSU7	1	47	2500	16	0
DSU7	2	43	3297	0	0
DSU7	3	24	2375	16	0
DSU7	4	24	2984	16	0
DSU7	5	24	2812	0	0
DSU8	0	80	2719	15	0
DSU8	1	40	2375	16	0
DSU8	2	26	2046	16	0
DSU8	3	28	2828	0	0
DSU8	4	25	3141	15	0
DSU8	5	25	3032	15	0

Table A.5: Results for small problems with normally utilized homogeneous gates when exta column generation is considered

Pr. Set	Extra Columns	Iterations	LP Time	IP Time	Frac=1
DSN1	0	54	1937	16	0
DSN1	1	39	2296	16	0
DSN1	2	32	2703	0	0
DSN1	3	24	2703	16	0
DSN1	4	25	3500	16	0
DSN1	5	25	3485	0	0
DSN2	0	54	1938	15	0
DSN2	1	30	1813	15	0
DSN2	2	29	2469	16	0
DSN2	3	23	2578	15	0
DSN2	4	20	2829	15	0
DSN2	5	20	2578	16	0
DSN3	0	79	3062	16	0
DSN3	1	42	2703	16	0
DSN3	2	31	2688	15	0
DSN3	3	22	2484	16	0
DSN3	4	27	3860	0	0
DSN3	5	27	3688	0	0
DSN4	0	69	2750	15	0
DSN4	1	33	2219	16	0
DSN4	2	27	2407	15	0
DSN4	3	20	2360	15	0
DSN4	4	19	2719	16	0
DSN4	5	19	2656	16	0
DSN5	0	68	2781	16	0
DSN5	1	36	2485	93	1
DSN5	2	25	2313	16	0
DSN5	3	22	2593	16	0
DSN5	4	22	3250	94	1
DSN5	5	22	3219	94	1
DSN6	0	51	2157	15	0
DSN6	1	38	2672	16	0
DSN6	2	23	2312	16	0
DSN6	3	24	3062	16	0
DSN6	4	21	3109	16	0
DSN6	5	21	3203	15	0
DSN7	0	73	3468	32	0
DSN7	3	34	2390	16	0
DSN7	4	20	3094	15	0
DSN7	5	20	3094	31	0
DSN8	0	82	3735	15	0
DSN8	1	34	2391	16	0
DSN8	4	23	2328	16	0
DSN8	5	20	3218	16	0
DSH1	0	33	1625	31	0

Table A.6: Results for small problems with highly utilized homogeneous gates when exta column generation is considered

Pr. Set	Extra Columns	Iterations	LP Time	IP Time	Frac=1
DSH1	1	17	1266	16	0
DSH1	2	14	1375	15	0
DSH1	3	13	1657	15	0
DSH1	4	14	2141	16	0
DSH1	5	14	2157	15	0
DSH2	0	31	1422	16	0
DSH2	1	21	1562	16	0
DSH2	2	13	1313	15	0
DSH2	3	15	1938	15	0
DSH2	4	10	1594	16	0
DSH2	5	10	1593	16	0
DSH3	2	11	594	15	0
DSH3	3	6	766	15	0
DSH3	4	6	969	16	0
DSH3	5	6	953	15	0
DSH4	0	12	547	16	0
DSH4	1	7	547	15	0
DSH4	2	6	641	31	0
DSH4	3	10	1328	16	0
DSH4	4	8	1281	16	0
DSH4	5	8	1297	15	0
DSH5	0	35	1672	16	0
DSH5	1	18	1390	16	0
DSH5	2	18	1890	16	0
DSH5	3	14	1890	16	0
DSH5	4	13	2125	15	0
DSH5	5	13	2125	16	0
DSH6	0	53	2547	63	1
DSH6	1	11	844	62	1
DSH6	2	15	1594	109	1
DSH6	3	11	1500	16	0
DSH6	4	10	1641	16	0
DSH6	5	10	1672	16	0
DSH7	0	10	485	15	0
DSH7	1	8	625	16	0
DSH7	2	10	1093	16	0
DSH7	3	7	953	31	0
DSH7	4	6	1016	16	0
DSH7	5	6	1000	31	0
DSH8	2	40	2063	62	1
DSH8	3	12	1672	94	1
DSH8	4	9	1547	31	0
DSH8	5	9	1547	31	0

Table A.7: Results for medium problems with under utilized homogeneous gates when exta column generation is considered

Pr. Set	Extra Columns	Iterations	LP Time	IP Time	Frac=1
DMU1	0	229	8016	172	1
DMU1	1	122	7594	94	1
DMU1	2	85	7000	94	1
DMU1	3	61	6859	125	1
DMU1	4	57	7937	0	0
DMU1	5	47	7906	110	1
DMU1	6	44	8625	0	0
DMU1	7	39	8703	110	1
DMU1	8	36	9032	15	0
DMU1	9	38	10672	16	0
DMU1	10	38	11062	16	0
DMU2	0	279	11359	16	0
DMU2	1	143	10203	16	0
DMU2	2	90	9203	16	0
DMU2	3	73	9656	16	0
DMU2	4	65	10610	93	1
DMU2	5	55	9656	16	0
DMU2	6	53	11328	15	0
DMU2	7	41	9547	16	0
DMU2	8	43	12313	16	0
DMU2	9	40	13031	15	0
DMU2	10	41	13922	16	0
DMU3	0	259	12672	16	0
DMU3	1	134	11187	219	1
DMU3	2	92	10969	15	0
DMU3	3	66	10250	219	1
DMU3	4	60	11422	16	0
DMU3	5	49	11078	15	0
DMU3	6	43	11297	16	0
DMU3	7	41	12312	16	0
DMU3	8	36	12203	125	1
DMU3	9	37	13890	141	1
DMU3	10	43	17047	141	1
DMU4	0	300	17031	94	1
DMU4	1	140	13329	15	0
DMU4	2	99	13406	16	0
DMU4	3	76	13187	16	0
DMU4	4	65	13860	31	0
DMU4	5	59	14922	31	0
DMU4	6	50	14797	31	0
DMU4	7	51	17157	31	0
DMU4	8	47	17734	31	0
DMU4	9	48	20219	16	0
DMU4	10	45	19859	31	0
DMU5	0	300	19687	156	1
DMU5	1	149	16266	31	0
DMU5	2	98	15063	15	0
DMU5	3	75	14922	172	1
DMU5	4	67	16250	15	0
DMU5	5	57	16438	31	0
DMU5	6	49	16359	141	1
DMU5	7	46	17625	31	0
DMU5	8	48	20578	16	0
DMU5	9	46	21953	15	0
DMU5	10	45	22313	31	0
DMU6	0	248	18328	16	0
DMU6	1	125	15172	16	0
DMU6	2	88	15125	15	0
DMU6	3	66	16594	31	0
DMU6	4	55	14844	31	0
DMU6	5	55	21578	31	0
DMU6	6	47	17860	31	0
DMU6	7	39	18204	31	0
DMU6	8	42	22704	156	1
DMU6	9	40	23000	32	0
DMU6	10	46	25453	31	0
DMU7	0	249	21125	141	1
DMU7	1	124	16156	157	1
DMU7	2	80	15937	156	1
DMU7	3	65	15110	172	1
DMU7	4	58	16500	172	1
DMU7	5	49	16563	31	0
DMU7	6	42	17578	47	0
DMU7	7	37	16641	156	1
DMU7	8	36	18234	47	0
DMU7	9	41	22641	31	0
DMU7	10	37	22062	219	1
DMU8	0	241	23281	47	0
DMU8	1	119	16813	31	0
DMU8	2	81	16531	31	0
DMU8	3	65	17078	31	0
DMU8	4	56	18141	31	0
DMU8	5	41	15718	47	0
DMU8	6	47	22609	47	0
DMU8	7	39	19234	47	0
DMU8	8	42	23188	172	1
DMU8	9	49	26000	47	0
DMU8	10	35	22875	172	1

Table A.8: Results for medium problems with normally utilized homogeneous gates when exta column generation is considered

Pr. Set	Extra Columns	Iterations	LP Time	IP Time	Frac=1
DMN1	0	237	24203	187	1
DMN1	1	121	19578	157	1
DMN1	2	79	17375	47	0
DMN1	3	64	18047	156	1
DMN1	4	53	18360	47	0
DMN1	5	47	18968	32	0
DMN1	6	37	17172	31	0
DMN1	7	39	21156	47	0
DMN1	8	39	23859	188	1
DMN1	9	35	23172	47	0
DMN1	10	34	24094	188	1
DMN2	0	216	23578	187	1
DMN2	1	111	19407	47	0
DMN2	2	71	17078	32	0
DMN2	3	56	17250	156	1
DMN2	4	45	17172	47	0
DMN2	5	42	18656	172	1
DMN2	6	36	18594	187	1
DMN2	7	36	20765	188	1
DMN2	8	38	23953	47	0
DMN2	9	31	21843	235	1
DMN2	10	32	23781	203	1
DMN3	0	203	24891	187	1
DMN3	1	107	21594	47	0
DMN3	2	71	19219	47	0
DMN3	3	56	19609	47	0
DMN3	4	49	21343	47	0
DMN3	5	41	20985	47	0
DMN3	6	36	20938	47	0
DMN3	7	34	21828	47	0
DMN3	8	32	22172	47	0
DMN3	9	34	30500	172	1
DMN3	10	34	26672	47	0
DMN4	0	229	28204	62	0
DMN4	1	126	24781	63	0
DMN4	2	78	21562	47	0
DMN4	3	59	20344	47	0
DMN4	4	56	24047	62	0
DMN4	5	40	19922	31	0
DMN4	6	41	21031	47	0
DMN4	7	38	24953	62	0
DMN4	8	35	25782	62	0
DMN4	9	35	29672	62	0
DMN4	10	34	29047	62	0
DMN5	0	205	27375	62	0
DMN5	1	103	23734	47	0
DMN5	2	82	22562	188	1
DMN5	3	53	19297	203	1
DMN5	4	51	23609	282	1
DMN5	5	38	20218	63	0
DMN5	6	37	21109	47	0
DMN5	7	35	22890	63	0
DMN5	8	39	29406	47	0
DMN5	9	36	28531	47	0
DMN5	10	35	28485	62	0
DMN6	0	264	35265	360	1
DMN6	1	118	24719	218	1
DMN6	2	86	23547	47	0
DMN6	3	68	23625	172	1
DMN6	4	55	23438	47	0
DMN6	5	49	24219	62	0
DMN6	6	44	25313	172	1
DMN6	7	39	25297	47	0
DMN6	8	39	28219	62	0
DMN6	9	41	32953	203	1
DMN6	10	44	36516	218	1
DMN7	0	251	33703	203	1
DMN7	1	122	25703	203	1
DMN7	2	91	26406	203	1
DMN7	3	67	24593	47	0
DMN7	4	58	25891	234	1
DMN7	5	46	24343	203	1
DMN7	6	45	27063	62	0
DMN7	7	41	28063	47	0
DMN7	8	40	32891	218	1
DMN7	9	36	30719	187	1
DMN7	10	35	30359	63	0
DMN8	0	194	27188	218	1
DMN8	1	112	24734	266	1
DMN8	2	71	21468	203	1
DMN8	5	57	22422	62	0
DMN8	6	36	25391	203	1
DMN8	7	33	23891	188	1
DMN8	8	29	23812	234	1
DMN8	9	33	30110	187	1
DMN8	10	33	33219	62	0

Table A.9: Results for medium problems with highly utilized homogeneous gates when exta column generation is considered

Pr. Set	Extra Columns	Iterations	LP Time	IP Time	Frac=1
DMH1	0	139	21219	235	1
DMH1	1	46	11906	219	1
DMH1	2	62	20219	344	1
DMH1	3	31	13969	78	0
DMH1	4	26	13734	188	1
DMH1	5	24	13953	63	0
DMH1	6	20	13812	63	0
DMH1	7	16	12421	63	0
DMH1	8	18	16031	219	1
DMH1	9	17	17921	250	1
DMH1	10	17	16297	219	1
DMH2	0	91	13875	203	1
DMH2	1	73	18937	219	1
DMH2	2	39	13391	250	1
DMH2	3	26	11078	234	1
DMH2	4	24	13250	250	1
DMH2	5	20	12860	203	1
DMH2	6	24	16562	203	1
DMH2	7	19	14985	62	0
DMH2	8	14	12469	78	0
DMH2	9	14	14172	78	0
DMH2	10	14	14234	63	0
DMH3	0	160	25094	93	0
DMH3	1	77	19438	484	1
DMH3	2	53	19203	203	1
DMH3	3	40	19735	78	0
DMH3	4	22	11625	250	1
DMH3	5	35	22703	63	0
DMH3	6	21	15641	62	0
DMH3	7	20	15766	78	0
DMH3	8	21	20672	94	0
DMH3	9	21	22781	78	0
DMH3	10	21	22282	62	0
DMH5	0	172	5906	78	1
DMH5	1	83	4672	16	0
DMH5	2	63	5078	78	1
DMH5	3	50	5594	16	0
DMH5	4	28	3922	94	1
DMH5	5	25	4047	78	1
DMH5	6	22	4297	94	1
DMH5	7	24	5156	16	0
DMH5	8	23	5140	0	0
DMH5	9	19	4984	94	1
DMH5	10	19	5297	78	1
DMH6	0	159	6047	172	1
DMH6	1	81	5328	109	1
DMH6	2	49	4625	78	1
DMH6	3	47	5266	16	0
DMH6	4	33	4687	16	0
DMH6	5	26	4625	79	1
DMH6	6	24	4672	94	1
DMH6	7	26	5781	109	1
DMH6	8	22	5532	0	0
DMH6	9	18	4907	15	0
DMH6	10	18	5235	15	0
DMH7	0	172	7454	125	1
DMH7	1	92	6671	0	0
DMH7	2	60	5859	16	0
DMH7	3	39	4859	16	0
DMH7	4	31	4891	94	1
DMH7	5	30	5875	0	0
DMH7	6	25	5766	15	0
DMH7	7	29	6875	78	1
DMH7	8	27	7469	16	0
DMH7	9	21	6218	16	0
DMH7	10	21	6703	0	0
DMH8	0	164	7281	16	0
DMH8	1	95	7218	16	0
DMH8	2	60	6375	16	0
DMH8	3	29	4015	16	0
DMH8	4	33	5938	16	0
DMH8	5	26	5703	15	0
DMH8	6	22	5609	16	0
DMH8	7	21	6375	16	0
DMH8	8	22	6344	16	0
DMH8	9	28	9281	15	0
DMH8	10	23	8078	16	0

Table A.10: Results for large problems with under utilized homogeneous gates when exta column generation is considered

Pr. Set	Extra Columns	Iterations	LP Time	IP Time	Frac=1
DLU1	0	300	18859	3750	1
DLU1	4	137	37047	453	1
DLU1	8	77	36281	234	1
DLU1	12	55	36281	16	0
DLU1	16	48	42015	266	1
DLU1	20	43	45656	219	1
DLU2	0	300	22125	1157	1
DLU2	4	143	45656	141	1
DLU2	8	86	48453	16	0
DLU2	12	61	48313	15	0
DLU2	16	52	53203	235	1
DLU2	20	47	56766	250	1
DLU3	0	300	25453	4328	1
DLU3	4	130	46188	406	1
DLU3	8	80	47406	188	1
DLU3	12	56	47437	172	1
DLU3	16	50	55187	16	0
DLU3	20	46	60093	266	1
DLU4	0	300	28125	1203	1
DLU4	4	141	54422	468	1
DLU4	8	83	55156	31	0
DLU4	12	60	59719	453	1
DLU4	16	56	64375	32	0
DLU4	20	48	69703	281	1
DLU5	0	300	30547	7438	1
DLU5	4	128	49078	453	1
DLU5	8	75	51313	469	1
DLU5	12	54	49891	312	1
DLU5	16	47	56375	344	1
DLU5	20	43	63313	281	1
DLU6	0	300	35391	4578	1
DLU6	4	130	61172	3313	1
DLU6	8	75	54890	703	1
DLU6	12	53	55516	500	1
DLU6	16	45	64578	219	1
DLU6	20	49	80469	328	1
DLU7	0	300	35968	7782	1
DLU7	4	124	57953	515	1
DLU7	8	70	60484	344	1
DLU7	12	53	61922	219	1
DLU7	16	49	73890	297	1
DLU7	20	42	78328	312	1
DLU8	0	300	42110	5515	1
DLU8	4	146	78656	719	1
DLU8	8	77	68765	313	1
DLU8	12	59	66844	297	1
DLU8	16	49	70656	328	1
DLU8	20	46	81016	453	1

Table A.11: Results for large problems with normally utilized homogeneous gates when exta column generation is considered

Pr. Set	Extra Columns	Iterations	LP Time	IP Time	Frac=1
DLN1	0	300	43031	8219	1
DLN1	4	125	65609	328	1
DLN1	8	72	63735	375	1
DLN1	12	58	71875	313	1
DLN1	16	43	72250	453	1
DLN1	20	43	83203	297	1
DLN2	0	300	47625	10109	1
DLN2	4	119	69391	38250	1
DLN2	8	71	68703	313	1
DLN2	12	52	69094	328	1
DLN2	16	43	72281	390	1
DLN2	20	36	72344	516	1
DLN3	0	300	45875	11641	1
DLN3	4	110	61734	9750	1
DLN3	8	61	59469	969	1
DLN3	12	49	67656	391	1
DLN3	16	41	73203	437	1
DLN3	20	36	76672	375	1
DLN4	0	300	50610	7328	1
DLN4	4	115	71750	2846890	1
DLN4	8	68	73125	281	1
DLN4	12	48	74016	312	1
DLN4	16	45	100625	312	1
DLN4	20	48	116016	375	1
DLN5	0	300	52360	13640	1
DLN5	4	110	76906	4484	1
DLN5	8	67	86157	422	1
DLN5	12	50	87078	47	0
DLN5	16	43	89031	47	0
DLN5	20	41	101140	297	1
DLN6	0	300	58297	11437	1
DLN6	4	115	88157	484	1
DLN6	8	67	78344	562	1
DLN6	12	52	93531	547	1
DLN6	16	44	97922	328	1
DLN6	20	37	100047	359	1
DLN7	0	300	61516	12109	1
DLN7	4	112	86125	6641015	1
DLN7	8	68	88906	735	1
DLN7	12	48	90203	344	1
DLN7	16	45	105000	390	1
DLN7	20	40	114437	516	1
DLN8	0	300	66875	11313	1
DLN8	4	116	92329	5734	1
DLN8	8	68	96515	516	1
DLN8	12	49	98750	438	1
DLN8	16	47	102734	422	1
DLN8	20	40	102141	328	1

Table A.12: Results for large problems with highly utilized homogeneous gates when exta column generation is considered

Pr. Set	Extra Columns	Iterations	LP Time	IP Time	Frac=1
DLH1	0	300	58453	703	1
DLH1	4	76	52937	4829	1
DLH1	8	47	51672	859	1
DLH1	12	35	55047	234	1
DLH1	16	29	61234	266	1
DLH1	20	27	66688	250	1
DLH2	0	300	57781	5172	1
DLH2	4	85	57031	297	1
DLH2	8	53	60203	328	1
DLH2	12	44	75047	94	0
DLH2	16	36	80109	344	1
DLH2	20	30	81688	328	1
DLH3	0	300	66500	1219	1
DLH3	4	101	73734	282	1
DLH3	8	61	78843	360	1
DLH3	12	43	86063	266	1
DLH3	16	34	79750	281	1
DLH3	20	34	96766	265	1
DLH4	0	300	67984	1281	1
DLH4	4	87	69516	53188	1
DLH4	8	49	71188	359	1
DLH4	12	40	75172	656	1
DLH4	16	34	80719	360	1
DLH4	20	32	101984	297	1
DLH7	0	300	73281	1203	1
DLH7	4	92	77578	76344	1
DLH7	8	54	74422	344	1
DLH7	12	40	79078	453	1
DLH7	16	33	85734	750	1
DLH7	20	35	116922	734	1
DLH8	0	300	72438	1203	1
DLH8	4	87	75110	10500	1
DLH8	8	55	78485	594	1
DLH8	12	40	85156	407	1
DLH8	16	33	94125	406	1
DLH8	20	36	114250	344	1

Appendix B

Heterogeneous Run Results without Extra Columns Generation

Table B.1: Results for small problems with under/normal/high utilized heterogeneous gates when exta column generation is not considered

Data Set	Iteration	LP Time	IP Time	Frac	Total Time
DSH1	48	5250	110	1	5360
DSH2	54	5531	0	0	5531
DSH3	15	1500	15	0	1515
DSH4	26	2563	0	0	2563
DSH5	57	5875	31	1	5906
DSH6	41	4172	31	1	4203
DSH7	25	2297	32	1	2329
DSH8	36	3468	16	1	3484
DSN1	92	8797	109	1	8906
DSN2	93	9172	78	1	9250
DSN3	100	10344	62	1	10406
DSN4	89	9688	47	1	9735
DSN5	100	11343	16	0	11359
DSN6	99	11359	16	0	11375
DSN7	86	10016	78	1	10094
DSN8	72	7469	78	1	7547
DSU1	100	10718	78	1	10796
DSU2	87	9062	0	0	9062
DSU3	123	10407	171	1	10578
DSU4	140	12625	0	0	12625
DSU5	131	12594	172	1	12766
DSU6	155	15640	94	1	15734
DSU7	110	11219	15	0	11234
DSU8	103	12906	16	0	12922

Table B.2: Results for medium problems with under/normal/high utilized heterogeneous gates when exta column generation is not considered

DMU1	329	104078	344	1	104422
DMU2	369	95562	516	1	96078
DMU3	362	108594	125	1	108719
DMU4	396	135172	4312	1	139484
DMU5	420	167266	4641	1	171907
DMU6	352	144469	62	0	144531
DMU7	308	75360	2546	1	77906
DMU8	349	113281	422	1	113703
DMN1	324	116219	156	1	116375
DMN2	261	70953	688	1	71641
DMN3	280	78266	344	1	78610
DMN4	332	105500	109	1	105609
DMN5	311	108109	1860	1	109969
DMN6	333	127656	2063	1	129719
DMN7	311	129578	2750	1	132328
DMN8	284	108485	31	0	108516
DMH1	154	63813	31	0	63844
DMH2	191	52422	422	1	52844
DMH3	214	59625	312	1	59937
DMH5	247	76234	250	1	76484
DMH6	193	61594	312	1	61906
DMH7	264	90640	344	1	90984
DMH8	233	61265	469	1	61734

Table B.3: Results for large problems with under/normal/high utilized heterogeneous gates when exta column generation is not considered

DLU1	1012	1413734	21253937	1	22667671
DLU2	1047	1390797	16468750	1	17859547
DLU3	964	1480609	18676625	1	20157234
DLU4	1027	1365000	18083657	1	19448657
DLU5	914	1149625	3829594	1	4979219
DLU6	904	1214468	6332125	1	7546593
DLU7	910	1175125	13178797	1	14353922
DLU8	1009	1332500	16335625	1	17668125
DLN1	840	1139954	1885312	1	3025266
DLN2	823	1100172	4128438	1	5228610
DLN3	704	894782	1587656	1	2482438
DLN4	799	1111359	2123485	1	3234844
DLN5	702	863797	1606969	1	2470766
DLN6	755	946609	3022719	1	3969328
DLN7	753	999578	2162453	1	3162031
DLN8	763	946734	4299125	1	5245859
DLH1	479	557562	5610	1	563172
DLH2	613	748625	60125	1	808750
DLH3	626	879829	23765	1	903594
DLH4	602	788406	26797	1	815203
DLH7	649	913438	57890	1	971328
DLH8	583	737281	26922	1	764203

Appendix C

Altered Half and 75% Flight Duration Data Results with Variance of Idle Time

Table C.1: Results for small problems half duration flights data with variance of idle time

Data Set	Iteration	LP Time	IP Time	Frac=1	Total Time
HALFDSU1	28	953	109	1	1062
HALFDSU2	32	766	31	1	797
HALFDSU3	41	1031	32	1	1063
HALFDSU4	48	1187	31	1	1218
HALFDSU5	43	1188	15	1	1203
HALFDSU6	53	1438	15	1	1453
HALFDSU7	40	1046	16	1	1062
HALFDSU8	34	890	32	1	922
HALFDSN1	37	1000	16	1	1016
HALFDSN2	28	766	16	1	782
HALFDSN3	42	1125	32	1	1157
HALFDSN4	31	843	32	1	875
HALFDSN5	41	1109	31	1	1140
HALFDSN6	35	969	16	1	985
HALFDSN7	36	953	32	1	985
HALFDSN8	30	781	16	1	797
HALFDSH1	30	797	31	1	828
HALFDSH2	38	1047	32	1	1079
HALFDSH3	36	1031	32	1	1063
HALFDSH4	35	813	31	1	844
HALFDSH5	28	672	16	1	688
HALFDSH6	29	1219	16	1	1235
HALFDSH7	31	766	15	1	781
HALFDSH8	39	969	31	1	1000

Table C.2: Results for medium problems half duration flights data with variance of idle time

Data Set	Iteration	LP Time	IP Time	Frac=1	Total Time
HALFDMU1	106	3734	16	1	3750
HALFDMU2	121	4297	47	1	4344
HALFDMU3	89	3281	32	1	3313
HALFDMU4	107	4031	31	1	4062
HALFDMU5	94	3594	31	1	3625
HALFDMU6	92	3390	32	1	3422
HALFDMU7	91	3359	31	1	3390
HALFDMU8	88	3156	31	1	3187
HALFDMN1	88	3125	31	1	3156
HALFDMN2	87	3172	31	1	3203
HALFDMN3	74	2937	31	1	2968
HALFDMN4	90	3391	31	1	3422
HALFDMN5	82	3031	31	1	3062
HALFDMN6	109	4391	46	1	4437
HALFDMN7	99	4437	31	1	4468
HALFDMN8	77	3266	47	1	3313
HALFDMH1	74	3328	32	1	3360
HALFDMH2	68	2938	31	1	2969
HALFDMH3	82	3593	47	1	3640
HALFDMH5	77	2906	94	1	3000
HALFDMH6	72	2485	31	1	2516
HALFDMH7	87	3031	32	1	3063
HALFDMH8	81	2875	31	1	2906

Table C.3: Results for large problems half duration flights data with variance of idle time

Data Set	Iteration	LP Time	IP Time	Frac=1	Total Time
HALFDLU1	331	24359	375	1	24734
HALFDLU2	367	28906	531	1	29437
HALFDLU3	288	22703	344	1	23047
HALFDLU4	296	24218	125	1	24343
HALFDLU5	295	23047	406	1	23453
HALFDLU6	249	19922	375	1	20297
HALFDLU7	196	15687	47	1	15734
HALFDLU8	263	22266	359	1	22625
HALFDLN1	228	19109	250	1	19359
HALFDLN2	286	24609	328	1	24937
HALFDLN3	226	19125	313	1	19438
HALFDLN4	238	21656	281	1	21937
HALFDLN5	214	18547	250	1	18797
HALFDLN6	269	23516	344	1	23860
HALFDLN7	225	19766	344	1	20110
HALFDLN8	229	20063	375	1	20438
HALFDLH1	181	16297	187	1	16484
HALFDLH2	203	17781	344	1	18125
HALFDLH3	233	21906	375	1	22281
HALFDLH4	215	20438	296	1	20734
HALFDLH5	223	20219	297	1	20516
HALFDLH7	193	15266	453	1	15719
HALFDLH8	207	15453	281	1	15734

Table C.4: Results for small problems 75% duration flights data with variance of idle time

Data Set	Iteration	LP Time	IP Time	Frac=1	Total Time
QUARDSU1	27	875	63	1	938
QUARDSU2	29	718	32	1	750
QUARDSU3	30	750	31	1	781
QUARDSU4	35	906	31	1	937
QUARDSU5	41	1032	15	1	1047
QUARDSU6	38	969	16	1	985
QUARDSU7	30	750	32	1	782
QUARDSU8	36	937	31	1	968
QUARDSN1	29	719	15	1	734
QUARDSN2	29	625	16	1	641
QUARDSN3	34	766	31	1	797
QUARDSN4	29	656	32	1	688
QUARDSN5	28	625	31	1	656
QUARDSN6	29	687	16	1	703
QUARDSN7	31	703	16	1	719
QUARDSN8	28	625	16	1	641
QUARDSH1	26	578	15	1	593
QUARDSH2	33	750	31	1	781
QUARDSH3	32	766	15	1	781
QUARDSH4	28	641	31	1	672
QUARDSH5	30	719	31	1	750
QUARDSH6	30	734	16	1	750
QUARDSH7	31	765	16	1	781
QUARDSH8	35	875	16	1	891

Table C.5: Results for medium problems 75% duration flights data with variance of idle time

Data Set	Iteration	LP Time	IP Time	Frac=1	Total Time
QUARDMU1	90	3047	31	1	3078
QUARDMU2	101	3485	31	1	3516
QUARDMU3	87	3125	31	1	3156
QUARDMU4	82	2969	31	1	3000
QUARDMU5	102	3813	31	1	3844
QUARDMU6	79	2875	31	1	2906
QUARDMU7	75	2750	31	1	2781
QUARDMU8	68	2437	32	1	2469
QUARDMN1	73	2609	32	1	2641
QUARDMN2	68	2625	31	1	2656
QUARDMN3	65	2469	31	1	2500
QUARDMN4	59	2390	32	1	2422
QUARDMN5	58	2156	47	1	2203
QUARDMN6	78	3000	47	1	3047
QUARDMN7	74	2797	31	1	2828
QUARDMN8	61	2218	47	1	2265
QUARDMH1	59	2281	32	1	2313
QUARDMH2	60	2313	31	1	2344
QUARDMH3	56	2719	31	1	2750
QUARDMH5	61	2468	32	1	2500
QUARDMH6	60	2406	31	1	2437
QUARDMH7	60	2469	31	1	2500
QUARDMH8	59	2438	31	1	2469

Table C.6: Results for large problems 75% duration flights data with variance of idle time

Data Set	Iteration	LP Time	IP Time	Frac=1	Total Time
QUARDLU1	187	15953	141	1	16094
QUARDLU2	223	20828	438	1	21266
QUARDLU3	221	19766	359	1	20125
QUARDLU4	239	20640	375	1	21015
QUARDLU5	201	16453	359	1	16812
QUARDLU6	211	18078	313	1	18391
QUARDLU7	204	17110	500	1	17610
QUARDLU8	200	17281	329	1	17610
QUARDLN1	167	14531	297	1	14828
QUARDLN2	160	14266	343	1	14609
QUARDLN3	152	13516	109	1	13625
QUARDLN4	163	15235	125	1	15360
QUARDLN5	151	13172	328	1	13500
QUARDLN6	166	15000	344	1	15344
QUARDLN7	166	14719	156	1	14875
QUARDLN8	154	13875	63	1	13938
QUARDLH1	118	10703	63	1	10766
QUARDLH2	125	11203	62	1	11265
QUARDLH3	134	13062	78	1	13140
QUARDLH4	126	12188	78	1	12266
QUARDLH5	122	11250	94	1	11344
QUARDLH7	132	13063	62	1	13125
QUARDLH8	131	12797	63	1	12860