Improving Lower Bound for Aircraft Routing Optimization

An Application of Lagrange Relaxation

Master's thesis in Complex Adaptive Systems

Karin Hult

DEPARTMENT OF MATHEMATICAL SCIENCE

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2023
www.chalmers.se

Master's thesis 2023

Improving Lower Bound for Aircraft Routing Optimization

An Application of Lagrange Relaxation

Karin Hult

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematical Sciences
Chalmers University of Technology
Gothenburg, Sweden 2023

Improving Lower Bound for Aircraft Routing Optimization
An Application of Lagrange Relaxation
Karin Hult

Supervisor: Andreas Westerlund, Jeppesen, a Boeing company
Examiner: Ann-Brith Strömberg, Mathematical Sciences

Master's Thesis 2023
Department of Mathematical Sciences
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Photo by Karin Hult.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2023

Improving Lower Bound for Aircraft Routing Optimization
An Application of Lagrange Relaxation
Karin Hult
Department of Mathematical Sciences
Chalmers University of Technology

Abstract

Aircraft routing is a complicated optimization problem which can be modelled as a form of resource-constrained integer multi-commodity network flow problem. When optimizing this problem, a good lower bound is useful to assess the quality of feasible solutions and for determining convergence of the solution algorithm. The technique currently used to calculate lower bounds is very fast, but does several relaxations on top of each other, which might yield a quite weak lower bound. This fact raises the question if it could be possible to develop an efficient method to strengthen the lower bound.

This master thesis examines an alternative method of calculating a lower bound by performing Lagrange relaxation with respect to only one constraint group, which theoretically would lead to a stronger lower bound. Subgradient optimization with Polyak step size is used to solve the Lagrangian dual problem. Tests are performed, both with realistic implementations and with idealistic parameters, to investigate the potential of Lagrange relaxation, and solving the Lagrange dual problem with subgradient optimization. Lastly the results are compared with an linear programming-based relaxation of the original problem.

The Lagrange relaxation shows promise as the idealized tests resulted in a significant improvement of the lower bound for most of the studied test cases. However, the realistic implementations of subgradient optimization did not perform consistently well for all the test cases; this indicates that improvements of the implementations are required for this method of solving the Lagrange dual problem to be usable in practice.

Keywords: Aircraft routing, Lagrange relaxation, non-smooth optimization, subgradient optimization, Polyak step size, network flow, lower bounds.

Acknowledgements

I would like to give my thanks to everyone who has helped me at Jeppesen, from contributing to a pleasant surrounding to helping with technical difficulties. Most of all I extend the utmost of gratitude to the Fleet optimization team: Mattias Grönkvist, Johan Fredlund, Chithong Luong, and especially my supervisor Andreas Westerlund, for their continuous support and willingness to help.

Karin Hult, Gothenburg, June 2023

Contents

List of Terms and Acronyms

Below is the list of terms and acronyms that have been used throughout this thesis listed in alphabetical order:

Glossary

Commodity	Something which traverses through a network, for example vehicles
Graph	A set of nodes connected by arcs representing a network

Acronyms

LP	Linear Program
LPBR	LP-based relaxation

Nomenclature

Below is the nomenclature of indices, sets, parameters, and variables that are used throughout this thesis.

Indices

i, j	Indices for flights or nodes corresponding to flights
k	Index for aircraft

Sets

$\mathcal{C}$	Set of connections between two flights
$\mathcal{F}$	Set of flights
$\mathcal{K}$	Set of aircraft
$\mathcal{P}_k \subseteq \mathcal{F}$	Set of preassigned flights for aircraft k
$\mathcal{R}_k \subseteq \mathcal{F}$	Set of prohibited flights for aircraft k

Variables

$x_{ij}^k \in \{0, 1\}$	Equals 1 is aircraft k connects from flight i to flight j
μ_i	Lagrangian multiplier for flight i
$\upsilon_i \in \{0, 1\}$	Equals 1 if flight i is left unassigned

Parameters

$\bar{q}$	Estimated objective value used for Polyak step-size
β	Parameter for Polyak step-size

List of Figures

List of Tables

1

Introduction

1.1 Background

This thesis was made as a project at the Jeppesen, which is a subsidiary of Boeing and develops optimization-based software products for airlines. These products help with a variety of tasks for different stages; scheduling crew training, flight planning, and calibration afterward to name a few. Each product is specialized in its own task but also interacts with other products, for example by matching aircraft or crew to flights.

One of the products from the fleet optimization team is *Jeppesen Aircraft Routing* (JAR). Given a schedule of flights to be covered, JAR determines which aircraft should operate which flight. For the aircraft routing problem, each flight should be operated by a single aircraft. The assignments of aircraft are also subjected to constraints such as maintenance, airport curfews, and aircraft restrictions. Each individual aircraft can be considered to have individual expenses such as fuel consumption. Thus the optimization problem is to minimize these costs while also fulfilling the given constraints.

Jeppesen's primary optimization algorithm for this problem uses column generation combined with an integer fixing heuristic. During the optimization run, a feasible solution is evaluated by comparing its objective value to a lower bound on the optimal value. The lower bound is also used to help determine convergence. Due to these interactions, a high lower bound could potentially increase the computational efficiency.

The current method for computing a lower bound is a so-called network flow relaxation which uses LP relaxation as well as relaxes the resource constraints and the multi-commodity aspects of the problem.

Since this is a few relaxations on top of each other, alternative methods for calculating lower bounds with fewer relaxations could potentially give a stronger lower bound.

1.2 Aim

The aim of the thesis is to investigate alternate methods for calculating lower bounds. Such a new method should generate a higher lower bound than the current one does while still being an actual lower bound. In order for any alternate method to be commercially useful, it also should not take too long to compute.

1.3 Limitations

This thesis corresponds to 20 weeks of work and was limited to investigating methods for calculating lower bounds. Thus it has not investigated ways to improve the current optimization algorithm aside from the impact on the algorithm from a higher lower bound.

A certain modelling feature, named global constraints, was also considered to be out of scope for this master thesis. However, it should be rather straight-forward to incorporate this feature in the developed framework in the future.

2

Theory

A lower bound is a value which is less or equal to all elements in a set. Common methods of computing lower bounds to minimization problems is to relax some of the constraints.

This chapter introduces topics within optimization which are required to understand the thesis. Readers who are already familiar with these topics may want skip this chapter and continue from Chapter 3.

2.1 Integer Programming

An integer linear program with binary variables is defined as

$$\begin{aligned} z^* := \min \ & \boldsymbol{c}^T \boldsymbol{x} \\ \text{s.t. } & A\boldsymbol{x} \geq \boldsymbol{a}, \\ & \boldsymbol{x} \in \{0,1\}^n, \end{aligned} \tag{2.1}$$

where $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m\times n}$, and $a \in \mathbb{R}^m$ This problem for a general matrix $A \in \mathbb{R}^{m\times n}$ is one of the NP-complete problems listed by R.M. Karp [1]; this means that even though this problem looks simple at first glance, it is generally very computationally burdensome to solve to proven optimality.

2.2 LP relaxation

LP relaxation is used to transform a (mixed) integer linear program into a continuous linear program, normally called LP. This is done by changing all integer variables into continuous variables. Specifically, an LP relaxation of a binary variable x would be:

$$x \in \{0,1\} \to 0 \leq x \leq 1. \tag{2.2}$$

Tardos [2] has shown that any linear program can be solved to proven optimality in polynomial time, for example an algorithm presented by Karmarkar [3]. However, these polynomials can be rather cumbersome as well and therefore other types of methods are often used in practice, such as simplex method [4]. These other methods are often faster in practice even though they have exponential time computational complexity.

2.3 Lagrange Relaxation

Consider the integer linear optimization problem (2.1), and let

$$A = \begin{bmatrix} B \\ D \end{bmatrix}, \quad \boldsymbol{a} = \begin{bmatrix} \boldsymbol{b} \\ \boldsymbol{d} \end{bmatrix}, \tag{2.3}$$

where B and $\boldsymbol{b}$ corresponds to all constraints which will be Lagrangian relaxed. The problem can then be written as

$$\begin{aligned} z^* := \min \ & \boldsymbol{c}^T\boldsymbol{x} \\ \text{s.t.} \ & B\boldsymbol{x} \geq \boldsymbol{b}, \\ & \boldsymbol{x} \in X, \end{aligned} \tag{2.4}$$

for $\boldsymbol{b} \in \mathbb{R}^m$, and $X := \{\boldsymbol{x} \in \{0,1\}^n | D\boldsymbol{x} \geq \boldsymbol{d}\}$, where $D\boldsymbol{x} \geq \boldsymbol{d}$ corresponds to all the constraints which will not be relaxed. For an arbitrary vector $\boldsymbol{\mu}$ the Lagrange function is

$$L(\boldsymbol{x}, \boldsymbol{\mu}) := \boldsymbol{c}^T\boldsymbol{x} + \sum_{i=1}^{m} \mu_i(b_i - \boldsymbol{B_i}\boldsymbol{x}) = \boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{\mu}^T(\boldsymbol{b} - B\boldsymbol{x}). \tag{2.5}$$

The Lagrangian dual function $q : \mathbb{R}^m \mapsto \mathbb{R}$ is defined as

$$\begin{aligned} q(\boldsymbol{\mu}) := \min L(\boldsymbol{x}, \boldsymbol{\mu}), \\ \text{s.t.} \quad \boldsymbol{x} \in X. \end{aligned} \tag{2.6}$$

If set X is separable as

$$X = X_1 \times X_2 \times ... \times X_\gamma, \tag{2.7}$$

it is possible to divide the computation of the Lagrangian dual function for a fixed value of $\boldsymbol{\mu}$ into γ independent problems.

In this thesis only equality constraints will be Lagrangian relaxed. For that reason the rest of theory will be specifically for equality constraints. The Lagrange dual problem is then

$$q^* := \max_{\boldsymbol{\mu} \in \mathbb{R}^m} \ q(\boldsymbol{\mu}), \tag{2.8}$$

which is a convex problem, since the objective function is concave (see [5], Ch. 6). Here μ_i is non-restricted in sign since only equality constraints are relaxed.

The weak duality theorem [5, p. 160-161] becomes:

Theorem 2.3.1. *Let* $\boldsymbol{x}$ *and* $\boldsymbol{\mu}$ *be feasible in the problems* (2.4) *and* (2.8) *respectively. Then,*

$$q(\boldsymbol{\mu}) \leq z^*. \tag{2.9}$$

Proof. Assume $\boldsymbol{\mu} \in \mathbb{R}^m$. Then, it holds that

$$\begin{aligned} q(\boldsymbol{\mu}) &= \min_{\boldsymbol{x}\in X} L(\boldsymbol{x}, \boldsymbol{\mu}) \\ &= \min_{\boldsymbol{x}\in X} \Big(\boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{\mu}^T(\boldsymbol{b} - B\boldsymbol{x})\Big) \\ &\leq \min_{\boldsymbol{x}\in X, B\boldsymbol{x}=\boldsymbol{b}} \Big(\boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{\mu}^T(\boldsymbol{b} - B\boldsymbol{x})\Big) \\ &= \min_{\boldsymbol{x}\in X, B\boldsymbol{x}=\boldsymbol{b}} \boldsymbol{c}^T\boldsymbol{x} \\ &= z^*. \end{aligned}$$

□

This implies that $q^* = \max q(\boldsymbol{\mu})$ is a lower bound on z^*, as defined in problem (2.4).

The lower bound obtained from Lagrangian relaxation can be shown to not be lower than the lower bound z^*_{LP} from LP relaxation [6], i.e.,

$$z^*_{LP} \leq q^*. \tag{2.10}$$

Combining (2.10) and (2.9) gives

$$z^*_{LP} \leq q^* \leq z^*, \forall \mathbb{R}^m. \tag{2.11}$$

2.4 Subgradient Optimization

One method for optimizing a Lagrangian dual problem is to use subgradient optimization, which is usable for convex optimization problems even when the objective function is not differentiable. A vector $\boldsymbol{g} \in \mathbb{R}^n$ is called a subgradient of a function f at $\boldsymbol{x} \in \mathbb{R}^n$ if

$$f(\boldsymbol{x}) \geq f(\boldsymbol{x}) + \boldsymbol{g}^T(\boldsymbol{y} - \boldsymbol{x}), \quad \forall \boldsymbol{y} \in \mathbb{R}^n. \tag{2.12}$$

Given a Lagrangian dual function q defined as

$$q(\boldsymbol{\mu}) = \min_{x\in X} \Big(f(\boldsymbol{x}) + \boldsymbol{\mu}^T\boldsymbol{g}(\boldsymbol{x})\Big), \tag{2.13}$$

where $\boldsymbol{g}(\boldsymbol{x}(\boldsymbol{\mu})) := \boldsymbol{b} - B\boldsymbol{x}(\boldsymbol{\mu})$ ($\boldsymbol{x}(\boldsymbol{\mu})$ being optimal in the solution to the subproblem in (2.6) for the dual point $\boldsymbol{\mu}$) is a subgradient for q with regards to $\boldsymbol{\mu}$. Based on work done by Shor [7], Algorithm 1 can be used to solve the Lagrangian dual problem with relaxed equality constraints.

Algorithm 1 Subgradient Optimization

$\boldsymbol{\mu}_0 \leftarrow \mathbf{0}$
$q_{\text{best}} \leftarrow -\infty$
$\tau \leftarrow 0$ ▷ τ is an iteration number
repeat
  Find solution $\boldsymbol{x}(\boldsymbol{\mu}_\tau)$ to the subproblem for computing $q(\boldsymbol{\mu}_\tau)$ in (2.6).
  $\boldsymbol{\mu}_{\tau+1} \leftarrow \boldsymbol{\mu}_\tau + s_\tau(\boldsymbol{b} - B\boldsymbol{x}(\boldsymbol{\mu}_\tau)).$ ▷ $s_\tau > 0$ is step size at iteration τ
  if $q(\boldsymbol{\mu}_{\tau+1}) > q_{\text{best}}$ **then**
    $q_{\text{best}} \leftarrow q(\boldsymbol{\mu}_{\tau+1}).$
  end if
  $\tau \leftarrow \tau + 1.$
until some termination criterion is met

Under certain assumptions this algorithm will converge in the sense that each dual iterate $\boldsymbol{\mu}_\tau$ will be closer to the set of optimal dual points than the previous iterate $\boldsymbol{\mu}_{\tau-1}$, meaning in practice that $\{q(\boldsymbol{\mu}_\tau) \to q^*\}$. One of these assumptions is that an infinite number of iterations is performed, which is not possible in practice.

To ensure the loop does not go on for infinity, termination criteria are generally used. For example a termination criterion can be to limit the number of iterations. Another termination criterion could be $|\boldsymbol{g}(\boldsymbol{x}_\tau)| \leq \epsilon$.

This algorithm can be expanded upon and modified to enable faster convergence (see [6], Ch. 5). For example there are multiple ways to choose the step size s_τ which may lead to faster convergence. With a constant step size it might not be possible to reach convergence. Decreasing the step size for each iteration can handle fine tuning of q, but might diminish before getting close enough to an optimal solution to the dual problem.

2.4.1 Polyak step size

Polyak [8] introduced a step size formula which allows for both taking larger steps as well as fine tuning. This step size formula is

$$s_\tau = \beta_\tau \frac{q^* - q(\boldsymbol{\mu}_\tau)}{||\boldsymbol{g}(\boldsymbol{x}_\tau)||^2}, \quad 0 < \epsilon_1 \leq \beta_\tau \leq 2 - \epsilon_2 < 2, \tag{2.14}$$

where q^* is defined in (2.8) and β_τ is a parameter. For cases where q^* is not known, which is usually the case, an approximate value can be used. In this report, such an approximate value is sometimes referred to as target. Using such an approximate value will likely not be as efficient as using q^*.

2.5 Multi-commodity Network Flow

Chapter 1.2 in *Network Flows* [9] introduces the *Minimum Cost Flow Problem*, which is the problem of minimizing the cost of sending a commodity from supply nodes, through a network, to satisfy demands at other nodes. This problem is generally modelled by a directed graph, where the arc from node i to node j has a cost c_{ij} associated with it. Each arc may also have a minimum and a maximum flow capacity.

Chapter 17 in *Network Flows* [9] describes the case when different commodities shall be sent through a shared network. This extension adds additional complexity compared to simple network flow problems as the different commodities may interact with each other in a shared network.

Given a graph with nodes V, arcs E, commodities K, and $x_{ij}^k \geq 0$ denoting the amount of flow of commodity k through arc (i, j), the general continuous multi-commodity network flow problem is:

$$\min \quad \sum_{(i,j)\in E} \sum_{k\in K} c_{ij}^k x_{ij}^k, \tag{2.15a}$$

$$\text{s.t.} \quad \sum_{k\in K} x_{ij}^k \leq u_{ij}, \quad \forall (i,\, j) \in E, \tag{2.15b}$$

$$\sum_{\{j:(i,j)\in E\}} x_{ji}^k - \sum_{\{j:(i,j)\in E\}} x_{ij}^k = b_i^k, \quad \forall i \in V, \forall k \in K, \tag{2.15c}$$

$$l_{ij}^k \leq x_{ij}^k \leq u_{ij}^k, \quad \forall (i,\, j) \in E, \forall k \in K. \tag{2.15d}$$

Here u_{ij} limits the total flow through arc (i, j). Furthermore b_i^k determines the flow balance for node i and commodity k. Generally $b_i^k > 0$ for a supply node, and $b_i^k < 0$ for a demand node. For the case when $b_i^k = 0$, any amount of commodity k which flows into node i also has to flow out of it. Finally u_{ij}^k and l_{ij}^k limits how much of each commodity k can flow through an arc.

3

Earlier Work

The aircraft routing problem is similar to a problem called the tail assignment problem. Since these two problems can be modelled very similarly, this master thesis relies on previous work done by Grönkvist in his PhD-thesis *The Tail Assignment Problem* [10], which combines mathematical modelling and column generation (see [10], Part II) with constraint programming (see [10], Part III).

3.1 Tail assignment problem

In [10] Chapter 4.1 Grönkvist introduces a mathematical model for the tail assignment problem, which is the same for the aircraft routing problem. For this problem, the objective is to construct routes for a set of aircraft which cover as many flights as possible while considering maintenance, regulations, and some aircraft already being assigned to some flights. For the convenience for this report, a slightly modified model is stated here.

In this model, k represents a aircraft and the nodes i, j represents flights. We denote the set of aircraft as $\mathcal{K}$, the set of flights which should be covered as $\mathcal{F}$, the set of connections between two flights i and j as $\mathcal{C}$, the set of preassigned flights for aircraft k as $\mathcal{P}_k$, and the set of restricted flights for aircraft k, based on regulations etc., as $\mathcal{R}_k$. Notably if $\mathcal{P}_k \cap \mathcal{R}_k \neq \emptyset$ the problem is infeasible, as preassigned flights should not also be restricted. We have variables x_{ij}^k which are 1 if aircraft k connects from flight i to flight j, with related cost c_{ij}^k, and this modified model explicitly allows for leaving flights unassigned by introducing a variable υ_i, with related cost h_i.

Additionally, the original model has a resource constraint for each maintenance type m and flight i using variables r_{im}. However since each aircraft has a maintenance type, here it is stated for each aircraft k instead of maintenance type. The modified constraint (3.1h) also has a non-linear function $r_k(\boldsymbol{x})$, instead of constraining the variables r_{im}, to explicitly state the dependence on $\boldsymbol{x}$.

$$
\begin{aligned}
\min \quad & \sum_{k\in\mathcal{K}} \sum_{(i,j)\in\mathcal{C}} c_{ij}^k x_{ij}^k + \sum_{i\in\mathcal{F}} h_i \upsilon_i, && (3.1a)\\
\text{s.t.} \quad & \upsilon_i + \sum_{k\in\mathcal{K}} \sum_{\{j:(i,j)\in\mathcal{C}\}} x_{ij}^k = 1, \quad \forall i \in \mathcal{F}, && (3.1b)\\
& \sum_{\{j:(i,j)\in\mathcal{C}\}} x_{ij}^k = 1, \quad \forall i \in \mathcal{P}_k, \forall k \in \mathcal{K}, && (3.1c)\\
& \sum_{\{j:(i,j)\in\mathcal{C}\}} x_{ij}^k = 0, \quad \forall i \in \mathcal{R}_k, \forall k \in \mathcal{K}, && (3.1d)\\
& \sum_{\{j:(i,j)\in\mathcal{C}\}} x_{ji}^k - \sum_{\{j:(i,j)\in\mathcal{C}\}} x_{ij}^k = 0, \quad \forall i \in \mathcal{F}, \forall k \in \mathcal{K}, && (3.1e)\\
& x_{ij}^k \in \{0,1\}, \quad \forall (i,j) \in \mathcal{C}, \forall k \in \mathcal{K}, && (3.1f)\\
& \upsilon_i \in [0,1], \quad \forall i \in \mathcal{F}, && (3.1g)\\
& r_k(\boldsymbol{x}) \leq l_k, \quad \forall k \in \mathcal{K}. && (3.1h)
\end{aligned}
$$

Here x_{ij}^k is a binary variable that determines whether aircraft k travels directly between node i and j. The cost for aircraft k to travel between node i and j is denoted as c_{ij}^k. Variable υ_i obtains the value 1 if flight i is left unassigned and h_i is the penalty cost for leaving flight i unassigned. Constraints (3.1b) ensure that only one aircraft is assigned to flight i. Constraints (3.1c) and (3.1d) limit which aircraft are assigned to flight i. The constraints (3.1e) are flow balance constraints which ensure continuous routes for each aircraft k.

The last constraints (3.1h) are resource constraints to ensure that each aircraft gets enough maintenance. This constraint limits how long an aircraft k can fly without being maintained. l_k is the maximum distance interval between two maintenance checks. Notably this is the only constraint which is not necessarily linear; for example $r_k(\boldsymbol{x})$ can be reset at certain airports. Kjerrström [11] describes in his master thesis a way of linearizing each resource constraint by dividing it into four different linear constraints and introducing additional binary variables.

3.1.1 Comparison to General Multi-Commodity Network Flow

Comparing this model without the maintenance constraints (3.1h) to the general multi-commodity model (2.15a)–(2.15d), several similarities can be observed. The objective function (3.1a) is analogous to (2.15a), it just sums over two sets of nodes instead of arcs. Similarly constraint (3.1e) corresponds to constraint (2.15c) with $b_i^k = 0$. However a notable difference is the integer requirements on the variables $\boldsymbol{x}$ and $\boldsymbol{\upsilon}$ which add additional complexity.

3.1.2 Current Solution Technique

As mentioned in Section 1.1, Jeppesen's current optimizer uses column generation combined with integer fixing heuristics. During column generation, the pricing problem is solved via a labeling algorithm described in Chapter 5 of [10]. More on how general labeling algorithms work can be read in e.g. Chapters 4 and 5 in [9].

3.1.3 Current Technique for Calculating a Lower Bound

When solving minimization problems, having calculated a good lower bound can be very helpful when assessing the quality of a feasible solution. The larger the gap between the upper and lower bounds, the less certain we can be of the quality of the solution. If the upper bound equals the lower bound the problem has been solved to optimality; however this is not always possible due to the properties of relaxations.

Jeppesen is currently using a fast technique for calculating lower bound called network flow relaxation; this lower bound is referred to as flow lower bound. This technique is expressed as a relaxed model in Chapter 4.6 in [10]. In this model several constraints are relaxed in different ways:

1. An LP relaxation is made on the variables $\boldsymbol{x}$ and $\boldsymbol{v}$.
2. The resource constraints (3.1h) are removed.
3. Multi-commodity aspects are mostly removed. Instead of having constraints for each aircraft k, the relaxed model only has one commodity, with notable changes:
   - Variables x_{ij} denoting the total amount of flow through arc (i, j), instead of a variable x_{ij}^k for each commodity k.
   - Arc-costs $c_{ij} = \min_{k \in \mathcal{K}} c_{ij}^k \quad \forall (i, j) \in \mathcal{C}$, which yields a possibly lower objective value.
   - The amount going out of the network equals the number of aircraft, i.e.

     $$\sum_{i \in \mathcal{F}} x_{is} = |\mathcal{K}|,$$

     where s is a so-called sink node. This is the only multi-commodity aspect of this model

This relaxed model is solved using FICO's Xpress LP-solve.

Even though this is a few relaxations on top of each other, due to how fast this method is it most likely will not be replaced but rather used in tandem with any new technique.

4 Methods

The main focus of this master thesis is to compute a lower bound via Lagrangian relaxation of the constraints (3.1b). Subgradient optimization is used to optimize the Lagrange dual problem. As it can be somewhat challenging to perform subgradient optimization to near-optimality, an LP-based relaxation was performed separately to obtain a comparative LP-based lower bound, which can be used for some consistency checks.

4.1 LP-based Relaxation

To obtain this comparative lower bound, an LP relaxation was made on the variables $\boldsymbol{x}, \boldsymbol{v}$ as seen in (2.2). Additionally the resource constraint (3.1h) was removed since it is rather time consuming to implement as an integer linear optimization model[1]. Due to the additional relaxation, this method will be referred to as LP-based relaxation (LPBR). These two relaxations are already present in the network flow relaxation, which means LPBR is less relaxed as the multi-commodity aspects are intact.

Even though the purpose of this method is to compare to a Lagrangian lower bound according to (2.11), it is possible that the removal of the constraints (3.1h) might decrease the lower bound too much for a proper comparison. However, this comparison still should work well for instances without resource constraints.

The relaxed model corresponds to model (3.1a)–(3.1e) with variable restrictions

$$0 \leq x_{ij}^k \leq 1, \quad \forall (i,j) \in \mathcal{C}, \forall k \in \mathcal{K}. \tag{4.1}$$

4.2 Lagrange Relaxation

In our case we have linear equality constraints (3.1b) which we will relax. As discussed above, this means that all the dual variables in $\boldsymbol{\mu}$ are unrestricted in sign.

The Lagrangian dual function of model (3.1a)–(3.1h) when performing Lagrange

[1]Especially considering that this thesis is focused on Lagrange relaxation.

relaxation with respect to constraint (3.1b) is:

$$q(\boldsymbol{\mu}) := \min_{\boldsymbol{x}\in X, \boldsymbol{\upsilon}\in\{0,1\}^{|F|}} \left(\sum_{k\in\mathcal{K}} \sum_{i\in\mathcal{F}} \sum_{j\in\mathcal{F}} c_{ij}^k x_{ij}^k + \sum_{i\in\mathcal{F}} c_i \upsilon_i + \sum_{i\in\mathcal{F}} \mu_i \Big(1 - \upsilon_i - \sum_{k\in\mathcal{K}} \sum_{j\in\mathcal{F}} x_{ij}^k \Big) \right) \tag{4.2}$$

$$= \sum_{i\in\mathcal{F}} \mu_i + \min_{\boldsymbol{x}\in X} \sum_{k\in\mathcal{K}} \left(\sum_{i\in\mathcal{F}} \sum_{j\in\mathcal{F}} (c_{ij}^k - \mu_i) x_{ij}^k \right) + \min_{\boldsymbol{\upsilon}\in\{0,1\}^{|F|}} \sum_{i\in\mathcal{F}} (c_i - \mu_i)\upsilon_i \tag{4.3}$$

$$= \sum_{i\in\mathcal{F}} \mu_i + \sum_{k\in\mathcal{K}} \min_{\boldsymbol{x}\in X_k} \left(\sum_{i\in\mathcal{F}} \sum_{j\in\mathcal{F}} (c_{ij}^k - \mu_i) x_{ij}^k \right) + \sum_{i\in\mathcal{F}} \min(c_i - \mu_i, 0) \tag{4.4}$$

$$:= \sum_{i\in\mathcal{F}} \mu_i + \sum_{k\in\mathcal{K}} \Psi_k(\boldsymbol{\mu}) + \sum_{i\in\mathcal{F}} \min(c_i - \mu_i, 0). \tag{4.5}$$

4.2.1 Solving the Lagrangian Subproblem

The third part is solved by assessing each term $\min(c_i - \mu_i, 0)$ separately for a given $\boldsymbol{\mu}$.

The second part of (4.5) separates into $|\mathcal{K}|$ independent resource-constrained shortest path problems, one for each aircraft k. Using the same sets and variables as in Section 3.1 these problems are:

$$\Psi_k(\boldsymbol{\mu}) = \min \sum_{(i,j)\in\mathcal{C}} (c_{ij}^k - \mu_i) x_{ij}^k, \tag{4.6a}$$

$$\text{s.t.} \quad \sum_{\{j:(i,j)\in\mathcal{C}\}} x_{ij}^k = 1, \quad \forall i \in \mathcal{P}_k, \tag{4.6b}$$

$$\sum_{\{j:(i,j)\in\mathcal{C}\}} x_{ij}^k = 0, \quad \forall i \in \mathcal{R}_k, \tag{4.6c}$$

$$\sum_{\{j:(i,j)\in\mathcal{C}\}} x_{ji}^k - \sum_{\{j:(i,j)\in\mathcal{C}\}} x_{ij}^k = 0, \quad \forall i \in \mathcal{F}, \tag{4.6d}$$

$$x_{ij}^k \in \{0,1\}, \quad \forall (i,j) \in \mathcal{C}, \tag{4.6e}$$

$$r_k(\boldsymbol{x}) \le l_k, \quad \forall k \in \mathcal{K}, \tag{4.6f}$$

where constraints (4.6f) corresponds to (3.1h), constraints (4.6d) corresponds to (3.1e), and constraints (4.6b) and (4.6c) corresponds to (3.1c) and (3.1d) respectively. These optimization problems are solved via existing code for the labeling algorithm mentioned in Section 3.1.2.

4.2.2 The Lagrange Dual Problem

The Lagrange dual problem

$$q^* := \max_{\boldsymbol{\mu}\in R^m} q(\boldsymbol{\mu}), \tag{4.7}$$

is a convex optimization problem with a non-smooth objective function. As such it is possible to use subgradient optimization to solve this problem.

4.2.3 Subgradient Optimization

Given a solution $(\boldsymbol{x}(\boldsymbol{\mu}), \boldsymbol{\upsilon}(\boldsymbol{\mu}))$ to $q(\boldsymbol{\mu})$, the component of the subgradient corresponding to node i is

$$g_i(\boldsymbol{x}_i(\boldsymbol{\mu}), \upsilon_i(\boldsymbol{\mu})) = 1 - \upsilon_i(\boldsymbol{\mu}) - \sum_{k \in \mathcal{K}} \sum_{\{j:(i,j) \in \mathcal{C}\}} x_{ij}^k(\boldsymbol{\mu}), \quad \forall i \in \mathcal{F} \tag{4.8}$$

which is then used in Algorithm 2 to perform subgradient optimization. There exists cases where q^* is negative due to negative arc costs. To allow for negative values of q to be considered, q_{best} is initialized as essentially $-\infty$.

Algorithm 2 Subgradient Optimization of Dual Problem

```
μ_0 ← 0
q_best ← −∞
τ ← 0
Set q̄ to some start target value.
repeat
    for all commodities k do
        Find solution x^k(μ_τ) to subproblem (4.6a)–(4.6f) which yields Ψ_k(μ_τ).
    end for
    for all nodes i do
        Find solution to min(c_iτ − μ_iτ, 0).
    end for
    Calculate q(μ_τ) = Σ_i μ_iτ + Σ_k Ψ_k(μ_τ) + Σ_i min(c_iτ − μ_iτ).
    if q̄ < q(μ_τ) then
        Increase q̄.
    end if
    Calculate step size s_τ, according to formula (2.14), with q̄ in place of q*.
    if s_τ < 0 then
        s_τ ← −s_τ                                        ▷ To ensure s_τ ≥ 0
    end if
    if q(μ_τ) > q_best then
        Set q_best ← q(μ_τ).
    end if
    μ_{τ+1} ← μ_τ + s_τ g(x_τ).
    Set τ ← τ + 1.
until termination criterion is met
```

The Polyak step size formula expressed in (2.14) needs the optimal value q^* and a parameter $\beta \in (0, 2)$. Related to the Polyak step size formula, Sjögren [12] investigated a method called *Variable Target Value Method* (VTVM) to more efficiently solve Lagrange dual problems. This method is mainly focused on the idea of raising the target $\bar{q}$ if $q(\boldsymbol{\mu}) > \bar{q}$, and decreasing $\bar{q}$ if q_{best} has not increased for a number of iterations. Even though he found VTVM worked poorly for solving linear programs, it appeared to be effective for producing a rough lower bound for these linear programs.

As q^* is unknown, an approximate value $\bar{q}$ needs to be used in the Polyak formula. Two different methods for setting $\bar{q}$ were used:

- Set $\bar{q}$ to the current flow lower bound obtained by network flow relaxation described in Section 3.1.3. Increase $\bar{q}$ if $q_{\text{best}} + \alpha > \bar{q}$ for some $\alpha > 0$.
- Set $\bar{q}$ to an upper bound from an input solution to problem (3.1), which is an input into Jeppesen's optimizer from the airlines to create the test cases[2].

Three different methods were used to choose β:

- constant $\beta \in (0, 2)$,
- linearly decreasing β from 2 to 0.0001, i.e. $\beta_\tau = \frac{1.9999(\tau_{\max} - \tau)}{\tau_{\max}} + 0.0001$,
- varying β based on how q_{best} is updated.

The last method of varying β is created for other parts of Jeppesen's code where they perform subgradient optimization, which means that it might have additional required assumptions and is not necessarily suitable for this thesis. However since this method of choosing β is part of well established code and is similarly with how $\bar{q}$ can be decreased in VTVM, there could be potential usage when solving our Lagrangian dual problem as well.

To investigate how well the method based on Lagrange relaxation performs given ideal values, tests were also performed with $\beta = 0.5$, $\bar{q} = z_{\text{best}}$, i.e. $\bar{q}$ set to the best known objective value to the problem (3.1). These tests was performed with 10^5 iterations. This test setup will be referred to as "idealistic" or "idealistic Lagrange relaxation" as it uses values which should not be known beforehand.

[2]The quality of the input solutions vary greatly; some are decently close to the optimal solution, and some leave all flights unassigned resulting in a very high upper bound.

5

Tests

All tests were performed within Jeppesen's development environment. The tests with subgradient optimization of the Lagrange dual problem were implemented in C++.

To solve the LPBR, a model was implemented in FICO Xpress Mosel. The model is given data of all costs and arcs in the problem in the form of a data file generated in C++. The optimization problem was then solved with FICO Xpress Optimizer. Even though this optimizer uses state-of-the-art techniques, it might be rather slow for larger test cases.

5.1 Test Cases

There are several hundred test cases in the Jeppesen test suite. Out of these, 20 representative test cases with different properties, have been selected by Jeppesen for evaluating the developed techniques for computing lower bounds. Some properties of these test cases are presented in Table 5.1, where the cases are numbered by their order in the internal test suite. For ten of the 20 test cases the set of resource constraints in (3.1h) is empty.

For the purpose of this thesis "Arc density" is defined as the number of arcs divided by the number of nodes in the graph.

All bounds in this thesis will be compared to the best known objective value z_{best} (i.e. an upper bound on the optimal value), with a relative gap defined as

$$\text{Relative gap} = \left| \frac{z_{\text{best}} - \underline{z}}{z_{\text{best}}} \right|, \tag{5.1}$$

where $\underline{z}$ is the lower bound which is compared. This relative gap is a bit optimistic since we would ideally use compare against z^*. Considering $z_{\text{best}} \geq z^* \geq \underline{z}$, the relative gap (5.1) is smaller than it should be. To compare the possible relative gaps we have:

Table 5.1: Sizes and properties of the 20 selected test cases.

Test case number	#Aircraft, $\lvert\mathcal{K}\rvert$	#Flights, $\lvert\mathcal{F}\rvert$	Arc density, $\lvert\mathcal{C}\rvert/\lvert\mathcal{F}\rvert$	Resource constraints
1	24	2,887	87.91	No
2	30	4,457	11.82	No
3	30	2,451	176.59	Yes
4	84	189,468	30.67	No
5	22	621	24.67	No
6	284	23,191	119.27	Yes
7	223	8,655	199.23	Yes
8	30	8,188	11.26	Yes
9	13	2,906	7.72	No
10	4	12	1.18	Yes
11	28	8,624	14.31	Yes
12	43	6,534	4.30	Yes
13	84	33,883	38.02	Yes
14	151	3,419	17.71	Yes
15	27	14,989	42.87	Yes
16	15	560	6.21	No
17	251	40,802	61.51	No
18	251	4,193	43.27	No
19	258	3,075	30.65	No
20	258	7,644	78.60	No

$$\left|\frac{z_{\text{best}} - \underline{z}}{z_{\text{best}}}\right| \le \left|\frac{z^* - \underline{z}}{z^*}\right| \le \left|\frac{z^* - \underline{z}}{\underline{z}}\right| \le \left|\frac{z_{\text{best}} - \underline{z}}{\underline{z}}\right|,$$

where the last two relative gaps were deemed inappropriate for this thesis since some tests ended up with a few lower bounds $\underline{z} = 0$. Considering we have no way of knowing the actual value of z^*, the relative gap (5.1) will be used for the purpose of comparing the different results to each other.

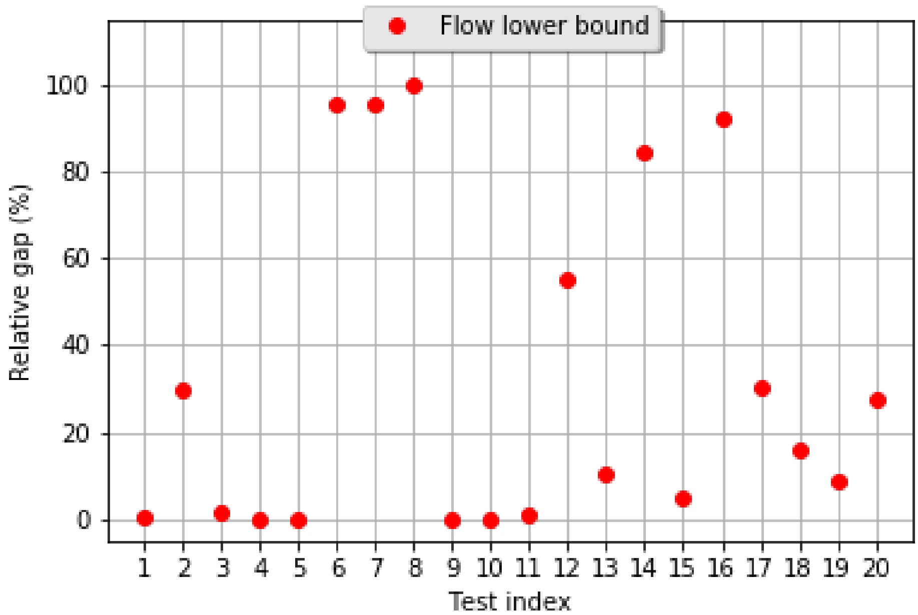


Figure 5.1: Relative gap between the upper and the flow lower bound

The relative gap for the flow lower bound is illustrated in Figure 5.1. This relative gap will be used when comparing results.

As we will discuss in Section 6.1.2, two of the computational experiments use upper bound from an input solution, given by the airline, as $\bar{q}$. The relative gap of these upper bounds can be observed in Figure 5.2, which were calculated using formula (5.1) with the upper bound compared to z_{best} instead of lower bound. Since this upper bound to z^* is a lot larger than z_{best} for some of the cases, these relative gap becomes very large.

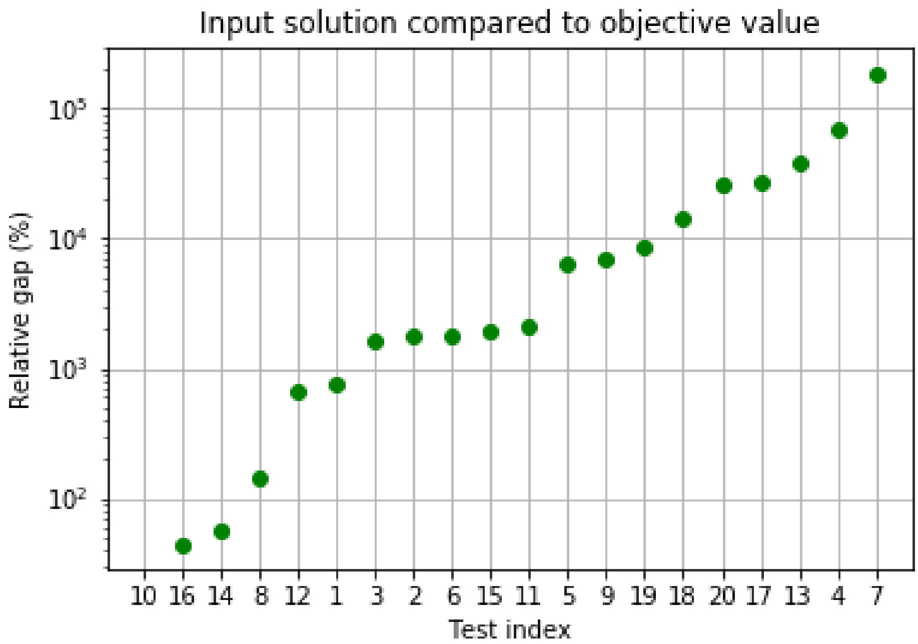


Figure 5.2: Relative gap for upper bound, obtained from an input solution, compared to z_{best}, in log-scale. Test case 10 has a relative gap of 0, and is therefore not visible in the log-scale. The test cases are sorted based on the relative gaps (5.1) with the upper bound instead of lower bound.

6

Results and Discussion

We next present results obtained using Lagrange relaxation. Then follows results from the LP-based relaxation compared to the flow lower bound and the idealistic Lagrange relaxation.

6.1 Lagrange Relaxation

First an idealistic subgradient optimization of the Lagrange dual problem is presented to investigate its potential for strengthening the lower bound. Afterward results from more realistic implementations of subgradient optimization are presented, and these results are compared.

6.1.1 Idealistic Implementation

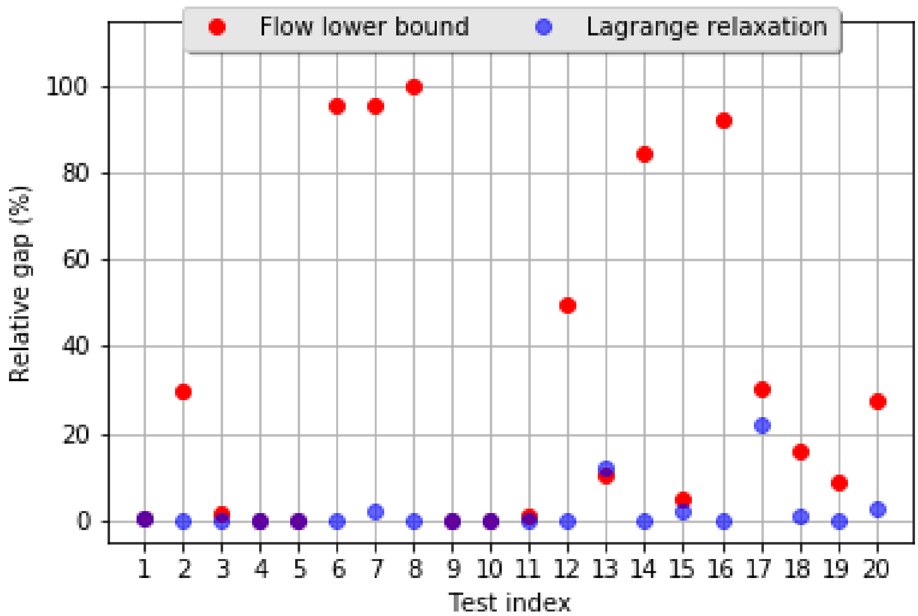


Figure 6.1: Relative gap (5.1) for the idealistic Lagrange relaxation and the flow lower bound, respectively. Blue dots covering red dots appear purple.

For this section the focus is to obtain as good of a lower bound as possible, with no regard to execution time. Also this implementation uses z_{best} which is normally not known beforehand. This experiment is characterized by:

- $\bar{q} = z_{\text{best}}$,
- $\beta = 0.5$,
- 10^5 iterations.

Figure 6.1 shows the relative gap for lower bounds obtained by Lagrange relaxation when using the best known objective value z_{best}, of a feasible solution to the original optimization problem, as target $\bar{q}$ in the Polyak step size formula. Generally these relative gaps are very close to 0, which indicates that the Lagrange relaxation could theoretically be very useful with a good enough implementation of subgradient optimization, especially considering the mean relative gap over all test cases being 2.2%.

Some test cases results in a bigger gap than the others, which either indicates that the relaxation performs poorly on these case or simply that they have not converged to near optimality in 10^5 iterations.

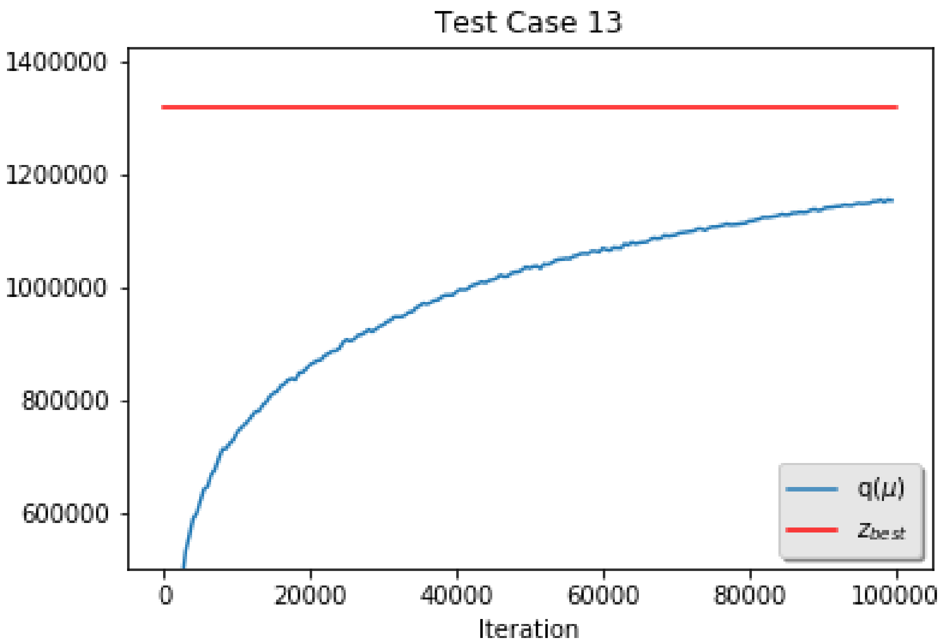


Figure 6.2: Values of the Lagrange dual function for every 500th iteration of the subgradient optimization, with the y-axis starting at $500,000$. This example is of test case 13.

To examine if the this gap is due to lack of convergence, we investigated test case 13 as an example. Figure 6.2 shows how the values of $q(\boldsymbol{\mu})$ changes with each iteration for test case 13. From Figure 6.2 it is apparent that $q(\boldsymbol{\mu})$ did not converge for test case 13, and there is no obvious reason to believe this case is the only test case which did not converge.

6.1.2 Realistic Implementations

As z_{best} is normally not known, an estimated value $\bar{q}$ has to be used instead. For this section the test cases have been ordered based on the relative gap of the upper bound, as seen in Figure 5.2, to make them easier to compare.

Figure 6.3 shows results from the computational experiments described by:

- $\bar{q}$ starting at flow lower bound, and $\bar{q}$ is increased when $q_{\text{best}} + \alpha > \bar{q}$ for some $\alpha > 0$,
- β started at 0.5, and was increased if q_{best} was updated two iterations in a row, and decreased if q_{best} was not updated for several iterations,
- 10^4 iterations.

From the results in Figure 6.3 we can observe that for most cases the Lagrange relaxation has a smaller relative gap than the flow lower bound. A lot of the cases still have a relatively big gap. These results indicate that this choice of $\bar{q}$ probably is not the best choice for raising the lower bound.

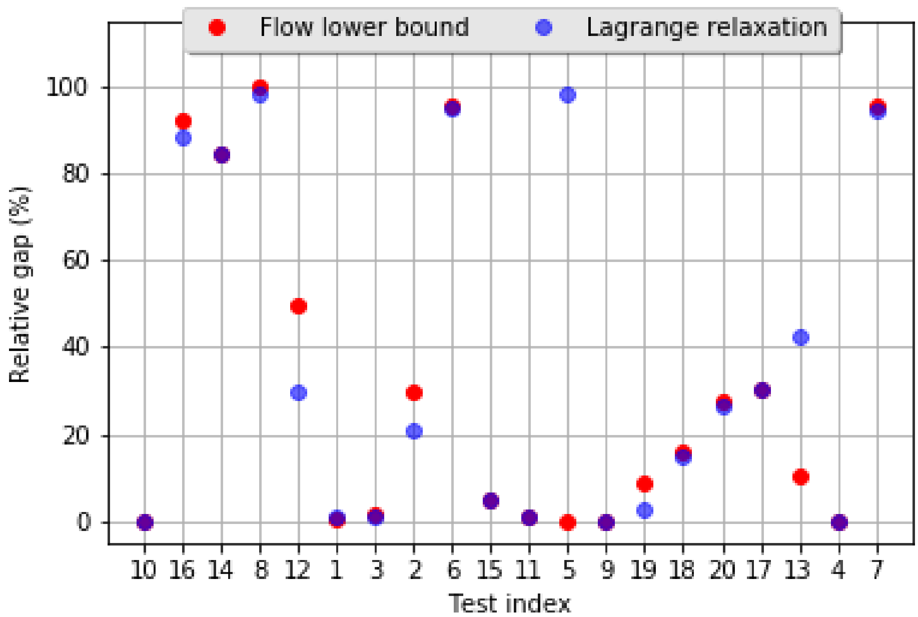


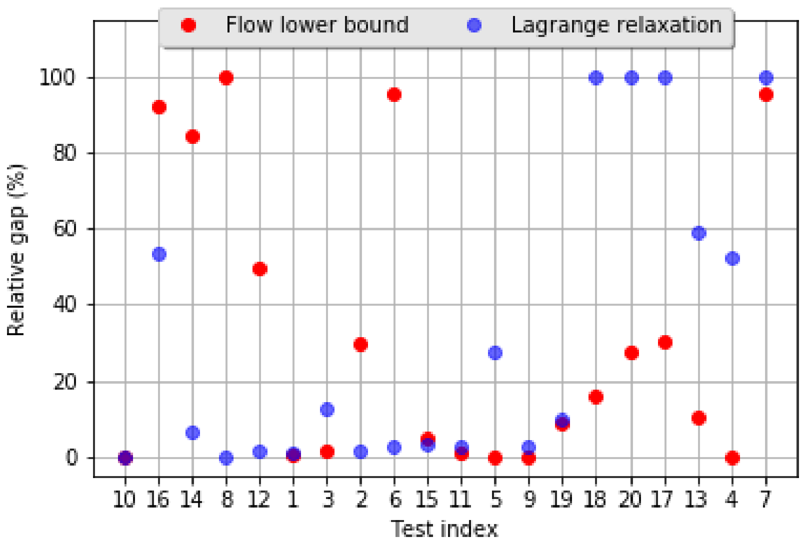
Figure 6.3: Relative gap (5.1) for Lagrange relaxation and flow lower bound. Polyak step size (2.14) parameters: target $\bar{q}$ starts at flow lower bound, both $\bar{q}$ and β changes based on the progress of q_{best}.

Next in Figure 6.4 is results from the computational experiments performed with:

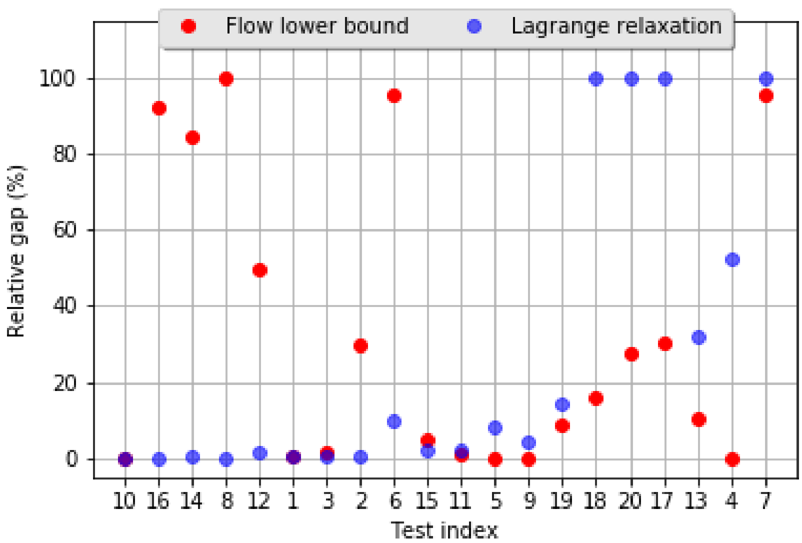
- $\bar{q}$ set to upper bound,
- 10^4 iterations,

and β for the different experiments was chosen as:

- Figure 6.4a: β changes based on progress of q_{best},
- Figure 6.4b: β decreases linearly from 2 to 0.0001.



(a) For this test setup β changes based on progress of q_{best}.



(b) For this test setup β decreases linearly from 2 to 0.0001.

Figure 6.4: Relative gap (5.1) for Lagrange relaxation and flow lower bound. Target in Polyak step size (2.14) is set to upper bound.

Results from when an upper bound from an input solution was used as target $\bar{q}$ in the Polyak formula can be observed in Figure 6.4. Combining these results with the information in Figure 5.2, it appears that the method based on Lagrange relaxation fails to produce a better lower bound when the relative gap of $\bar{q}$ is more than 5000%. However, given a decent input solution this method shows promise for calculating a strong lower bound. Comparing Figures 6.4a and 6.4b, using a linearly decreasing value of β appears to more consistently result in a decent lower bound. The difference between these two methods indicates that different alternatives for choosing β can result in better or worse lower bounds.

Preliminary results with another initialization to the test setup, which generates an upper bound by creating an initial solution instead of using the input solution from airlines, resulted in some test cases performing better. This could indicate that it would be worth spending time on finding a better initial solution to improve the performance of the algorithm.

6.1.3 Comparisons of Lower Bounds

The resulting mean relative gap (5.1) of each of the methods can be observed in Table 6.1 where "upper" and "lower" is which of the methods of choosing $\bar{q}$ mentioned in Section 4.2.3.

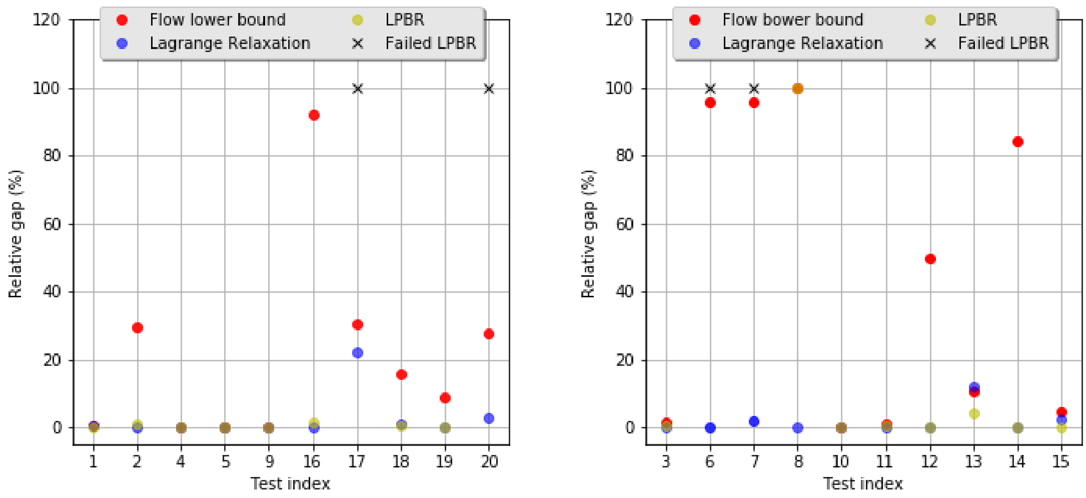
Table 6.1: Mean resulting relative gap when using the different bounds.

Lower bound	Mean relative gap (5.1) (%)
Flow lower bound	32.7
Lagrange, from test setup displayed in Figure 6.3	37.1
Lagrange, from test setup displayed in Figure 6.4a	32.4
Lagrange, from test setup displayed in Figure 6.4b	27.0
Idealistic Lagrange	2.2

Given that the mean relative gap of the idealistic Lagrange relaxation is only 2.2%, the Lagrange relaxation shows a high potential. Even though the current realistic implementations have comparable results to the flow lower bound, computing this new lower bound is slower and these implementations are therefore not quite useful in their current state. An improved implementation of solving the Lagrange dual problem could be useful as an additional method for computing a lower bound.

6.2 Comparison to an LP-based Relaxation

The relative gap of LPBR, explained in Section 4.1, compared to the flow lower bound and Lagrange relaxation can be observed in Figure 6.5. For most cases the lower bound obtained by LPBR is better than the flow lower bound. Disregarding the test cases which were terminated due to their size, the lower bounds from LPBR in Figure 6.5a are very close to z_{best}.



(a) Without resource constraints. **(b)** With resource constraints.

Figure 6.5: Relative gap for the lower bound from LPBR, idealistic Lagrange relaxation, and flow lower bound. x indicates that LPBR failed for that test case. For some test cases the dots overlap, which is displayed by blended colors.

Recall from Section 4.1 that resource constraints are not present when computing the LPBR bound. As expected LPBR does not perform as consistently well on the test cases with resource constraints.

Compare the results of Lagrange relaxation and LPBR in Figure 6.5b. Observe that there are two test cases with a visibly larger gap from Lagrange relaxation than from LPBR. Had the subgradient optimization converged to q^*, the best found value of the Lagrange dual function should never be lower than LPBR. Observing that $q_{\text{best}} < z^*_{\text{LPBR}}$, is consistent with the hypothesis in Section 6.1.1 that these test cases did not converge to an optimal value.

When reading in data into Mosel, four of the 20 test cases were either too big to write to file properly, or the time to read in data to the model was too long to be computationally viable. These cases were terminated and the relative gap was set to 100. Table 5.1 shows that the failed test cases have a combination of a lot of aircraft, flights, and high arc density, which could potentially explain the issues reading data into the model by the test cases being too big.

When looking at test case 8 in Figure 6.5b, it might appear that this test case failed, but it has a relative gap of 99.98% which means it only has a really low, i.e. weak, lower bound compared to z_{best}. This test case has resource constraints and 313 unassigned flights in the best solution found. However, if the resource constraints are removed from this test case the best solution has no unassigned flights, which would give a much lower value of z_{best}.

7

Conclusion and Further Research

7.1 Conclusion

The Lagrangian relaxation shows promise as the idealized tests resulted in a mean relative gap of 2.2%. This could be improved even further with a better convergence of q_{best}. Given the several improvements which could be made both to the input parameters to subgradient optimization and its implementation, Lagrange relaxation combined with subgradient optimization could potentially be useful in the future.

However, the realistic implementations of solving the Lagrange dual problem currently do not perform consistently well. Even the best performing parameters of a linearly decreasing β and an upper bound as a target $\bar{q}$ only worked decently when the relative gap of $\bar{q} < 5000\%$. If we cannot know beforehand how big the relative gap is, this method of obtaining a lower bound before optimization is most likely not usable in its current state.

7.2 Further Research

The execution time of obtaining a lower bound via Lagrange relaxation is currently too slow to be viable for commercial use. However, further improvements to the solution algorithms should increase the performance. For example faster convergence would result in faster execution time. Methods to improve the convergence could also result in more consistent results since the optimization of the dual problem would be more likely to get close to optimality before being terminated.

Subgradient optimization often has a tendency to exhibit a zigzag behaviour which causes a slower convergence. One way to handle this behaviour is by deflection, which is a method to take into account the subgradients from previous iterations when computing the search direction instead of just using the latest found subgradient. It would be interesting to incorporate some different deflection techniques, perhaps inspired by methods in [13].

The Lagrange relaxation appears to be promising in most instances, but none of the realistic implementations for solving the Lagrange dual problem works for all test cases. Further investigating how to choose β and $\bar{q}$ could result in more consistently strong lower bounds, for example by using $\bar{q}$ closer to the objective value. Even though it is not possible to determine the gap of the input solution before calculating the lower bound, spending a little more time to find a better initial solution could significantly improve the reliability of the subgradient optimization.

Since a lot of difficulties appears to come from the target $\bar{q}$ not being close enough to q^*, it could be useful to investigate methods for obtaining a better target $\bar{q}$. Blomgren [14] conducted a master thesis investigating several methods for producing initial solutions; some of which most likely could be used to improve the subgradient optimization. It would also be interesting to investigate these methods in the context of improving computation time, as it appears her focus was investigating their effect on costs in a previous testing environment.

Another possibility, which does not require a target at all, would be to instead investigate alternative methods for non-smooth optimization. Two such examples are the bundle method, see e.g. Brännlund, Kiwiel, and Lindberg [15], and a method introduced by Chambolle and Pock [16]. These methods do not require a target and can therefore be useful when q^* is unknown.

Furthermore, in contrast to improving convergence, parallelization via threads is commonly used to increase performance of code. Since each resource-constrained shortest path problem (4.6a)–(4.6f) can be solved independently of each other; the inner loop in Algorithm 2 is very suitable for parallelization. The separability for each commodity means that the decrease in computational time of this loop should be proportional to the number of threads used.

7.2.1 Out of Scope Research

As mentioned in Section 1.3, global constraints have not been taken into account in this thesis. However it would be interesting to see how well the Lagrange relaxation works on test cases with this additional feature, and whether or not further implementation is required to support these test cases.

This thesis was also limited to only investigating methods for computing lower bounds. However, given the framework developed to optimize the Lagrange dual problem, it also opens up for developing a so called *Lagrangian heuristic* [17] to obtain an upper bound. More information about this type of heuristic can be found in e.g. [18]. Such an heuristic is to use the properties of the subgradient $g_i(\boldsymbol{x})$ to obtain a better upper bound. For example, it can also be possible to make minor adjustments to $\boldsymbol{x}$ and $\boldsymbol{v}$ to recover feasibility and obtain a feasible solution.

Bibliography

[1] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations,* R. E. Miller, J. W. Thatcher, J. D. Bohlinger, Ed., Boston, MA: Springer US, 1972, pp. 85−103.

[2] E. Tardos, "A strongly polynomial algorithm to solve combinatorial linear programs," *Operations Research*, vol. 34, no. 2, pp.250−256, 1986.

[3] N. Karmarkar, "A new polynomial-time algorithm for linear programming," *Combinatorica,* vol. 4, pp. 373–395, Murray Hill, NJ: AT&T Bell Laboratories, 1984.

[4] D. Goldfarb, "On the Complexity of the Simplex Method", in *Advances in Optimization and Numerical Analysis,* S. Gomez, JP. Hennart, Ed., Dordrecht, NL: Springer Netherlands, 1994, pp. 25−38.

[5] N. Andréasson, A. Evgrafov, M. Patriksson, E. Gustavsson, Z. Nedělková, Kin Cheong Sou, M. Önnheim, *An Introduction to Continuous Optimization,* 3rd ed., Lund, SE: Studentlitteratur AB, 2022.

[6] A. Strömberg, "Lagrangean duals and subgradient methods for linear and mixed-binary linear optimization", *Course literature for the course TMA522 Large-Scale Optimization*, Chalmers, 2022. Available: `https://chalmers.instructure.com/courses/20879/files/folder/Literature?preview=2419548`

[7] N. Z. Shor, *Minimization Methods for Non-differentiable Functions,* Heidelberg, GE: Springer Berlin, 1985.

[8] B. T. Polyak, "Minimization of unsmooth functionals," *USSR Computational Mathematics and Mathematical Physics*, vol. 9, no. 2, pp. 14−−29, 1969.

[9] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows*, Englewood Cliffs, NJ: Prentice Hall, 1993.

[10] M. Grönkvist, "The Tail Assignment Problem," Ph.D. dissertation, Chalmers University of Technology, 2005. Available: `https://ww1.jeppesen.com/documents/aviation/commercial/crtr0502_final.pdf`

[11] J. Kjerrström, "A Model and Application for the Resource Constrained Shortest Path Problem with Reset Possibilities", M.S. thesis, Chalmers University of Technology and University of Gothenburg, 2003.

[12] P. Sjögren, "Solving the Master Linear Program in Column Generation Algorithms for Airline Crew Scheduling using a Subgradient Method," M.S. thesis, Chalmers University of Technology and University of Gothenburg, 2009. Available: `http://www.math.chalmers.se/Math/Research/Optimization/reports/masters/PerSjogren-final.pdf`

[13] R. Belgacem, A. Abdessamad, "A new modified deflected subgradient method," *Journal of King Saud University - Science,* vol. 30, no. 4, pp. 561−−567, 2018.

[14] E. Blomgren, "Creating Initial Solution for the Tail Assignment Problem," M.S. thesis, Chalmers University of Technology and University of Gothenburg, 2018. Available: `https://odr.chalmers.se/server/api/core/bitstreams/bf5b435f-dfd7-4590-bcee-f073ccaa147a/content`

[15] U. Brännlund, K. C. Kiwiel, P. O. Lindberg, "A descent proximal level bundle method for convex nondifferentiable optimization," *Operations Research Letters,* vol. 17, no. 3, pp. 121−−126, 1995.

[16] A. Chambolle, T. Pock, "A First-Order Primal-Dual Algorithm for Convex Problems with Applications to Imaging," *Journal of Mathematical Imaging and Vision,* vol. 40, no. 1, pp. 120−−145, 2011.

[17] M. L. Fisher, "The Lagrangian Relaxation Method for Solving Integer Programming Problems," *Management Science,* vol. 20, no. 12, pp. 1861−−1871, 2004.

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

DEPARTMENT OF MATHEMATICAL SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY