CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Deep Learning in State of the Art Airline Crew Rostering Algorithms

Predicting compatibility between tasks and crew members using deep learning in order to accelerate column generation in airline crew rostering problem algorithms by altering the selection process in the pricing problem.

Master's thesis in Engineering Mathematics and Computational Science

Jonathan Nillius

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2022

Master's thesis 2022

Deep Learning in State of the Art Airline Crew Rostering Algorithms

Predicting compatibility between tasks and crew members using deep learning in order to accelerate column generation in airline crew rostering problem algorithms by altering the selection process in the pricing problem.

Jonathan Nillius

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2022

Deep Learning in State of the Art Airline Crew Rostering Algorithms
Predicting compatibility between tasks and crew members using deep learning in order to accelerate column generation in airline crew rostering problem algorithms by altering the selection process in the pricing problem.
JONATHAN NILLIUS

Supervisor: Jonah Brown-Cohen, Department of Computer Science and Engineering
Advisor: Andreas Gunnarsson, Jeppesen Systems AB
Examiner: Devdatt Dubhashi, Department of Computer Science and Engineering

Master's Thesis 2022
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2022

Deep Learning in State of the Art Airline Crew Rostering Algorithms
Predicting compatibility between tasks and crew members using deep learning in order to accelerate column generation in airline crew rostering problem algorithms by altering the selection process in the pricing problem.

Jonathan Nillius
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

When distributing work among employees in Airline crew planning a problem called the crew rostering problem is formed. It is a combinatorial optimization problem and solving large problem instances commonly utilize column generation. This thesis investigates utilizing machine learning predictions instead of reduced costs in the pricing problem. The machine learning model predicts how likely it is that a task is assigned a crew in a supervised learning fashion, by being trained on historical planning problems. The aim is to then utilize the model to improve computational speed in solving future problems.

This thesis presents results suggesting that it is conceptually possible to improve computational time of state of the art crew rostering algorithms with accurate predictions. Training a deep learning model able to make such accurate predictions is found to be very difficult given the techniques and data experimented with. Thus the thesis concludes that further research for improving this concept is needed in two main directions, feature extraction and model techniques.

Keywords: airline crew rostering, machine learning, deep learning, combinatorial optimization, column generation, pricing problem, resource-constrained shortest path problem.

Acknowledgements

Firstly I would like to address a big thank you to my supervisor at Jeppesen, Andreas Gunnarsson, for always taking time to guide me through Jeppesens software and for interesting discussions.

I also want to thank everyone else at Jeppesen, especially the crew rostering team, who helped me navigate and understand their software framework.

Finally, thank you to my supervisor at Chalmers, Jonah Brown-Cohen, for all the great discussions about machine learning.

Jonathan Nillius, Gothenburg, January 2022

Contents

List of Figures

List of Tables

1
Introduction

In recent decades the air transportation industry has grown immensely and naturally along with it so has airlines. The size of the major airlines has developed a larger demand than ever for efficient planning solutions where a small fractal of percentage improvements can lead to millions of dollars in savings. The discipline that studies methods for logistics and planning is known as Operations Research (OR) and was first developed in order to optimize resource allocation in the second world war. Since then many algorithms producing optimal or near-optimal solutions have been developed but because of the complexity of these problems, they are heavily reliant on heuristics. In a more recent time span, the field of machine learning (ML) has been taking large steps forward, especially as a practice since computers have become more powerful. Because of its principled and optimized way of approaching decision making it looks like a promising candidate in uniting with OR to improve state of the art solution methods for planning problems. Thus this thesis aims to investigate techniques for merging ML into state of the art optimization algorithms for what today is one of the most complex planning industries, the airline industry.

Disregarding the last two years (2020-2021), during which a pandemic has put the market of transportation in a state of exception, one can observe a continuous growth of revenue in the airline industry. Airlines are the key actors when it comes to all forms of quick long-distance transportation of both humans and inanimate cargo. When deciding and formulating in detail how these transportations are to be executed multiple planning problems arise. What routes should be flown? Which airplane should fly which route? Which employees should operate the flights? Because of the many different quality aspects of a desirable solution and the scale of these problems they are all very complex to solve.

The company Boeing has a subsidiary called Jeppesen which specialize in the development of software tools for airline planning. Their products are used by major airlines all over the world. In their Swedish office in Gothenburg, the main focus is on software for Crew and Operations management. An important part of Crew planning is that of assigning work to individual employees, i.e. creating personalized work schedules which in the industry is referred to as a *roster*. The problem of creating a roster for each crew member is called the *Crew Rostering Problem* (CRP). In this master's thesis, the focus will be on the CRP and the project is a collaboration with Jeppesen.

1.1 Crew Rostering

In this section, an introduction to the CRP will be presented in order to understand its context, purpose and importance. The CRP is a subproblem of Crew Scheduling which is the problem of distributing work among employees. The problems of Crew Scheduling is generally solved in two phases out of which the first phase is the problem of grouping together flight legs forming an anonymous schedule where all flight legs require the same competence and number of crew members. A group of flight legs is called a *pairing* (even though they may contain more flight legs than a pair) and hence it has the name: *The Crew Pairing Problem*. The second phase is the CRP which has the objective to create personalized schedules that consist of pairings but also other tasks like: ground duties, reserve duties and off-duty blocks. A personalized schedule is called a *roster* and is usually planned on a monthly basis.

Finding good solutions to the CRP is of great importance for both the individual crew members and the airline's economy. For crew, good solutions mean that their preferences are satisfied. For airline's quality solutions can allow for huge economic savings, in fact, next to fuel costs, crew costs represent the largest single cost factor for airlines [7]. It also seems reasonable to assume that if crew members are satisfied with their work they are unlikely to call in sick or even quit their job, perform assigned tasks better etc. which could also result in large savings. Thus it is of great importance for airlines to keep their employees satisfied.

There are typically four kinds of objectives used when formulating a CRP and in practice, they are commonly combined into one objective function. The first potential objective is real costs which consider transportation costs, hotel costs and more. The second is the robustness of the solution which is basically planning with margins i.e. avoiding solutions where real-world disruptions in the form of for example a crew becoming sick could cause a need for large changes in the plan. The third is related to particular roster attributes and can be for instance how equally distributed the workload is. The fourth and last is related to individual crew preferences such as when a given member wants time-off. These objectives and the CRP is described in greater detail by Kohl and Karisch [14]

Since there are so many rules and regulations to follow in the CRP the number of mathematical constraints is expected to be large. Kohl and Karisch divide these into three categories which we will adopt in this report. The first category is called *Horizontal rules* and regards only a single roster. These handle the attributes of the crew member which the roster concerns as well as properties of the assigned tasks. The second category is called *Vertical rules* and, opposite of the previously mentioned category, these rules deal only with restrictions for more than one roster. A commonly used type of rule in this category is called *crew complement* and handles requirements on additional (more than one) crew members being required for a given task. Another vertical rule is *qualification-type constraints* that deals with linked competencies like for example a given flight requiring at

least one member speaking a certain language. The last category of rules is called *Artificial rules* and aims to impose constraints that restrict solution space in a way that only unattractive solutions are pruned. An example of this is the previously mentioned solution robustness.

1.2 Background and aim

The CRP is a Combinatorial Optimization problem which means that an exhaustive search is not a tractable option. This is because the number of rosters is too large for even being stored on a computer. If a problem has for example have 300 pairings the number of possible ways to combine them for each crew into 2^{300} different (legal and illegal) rosters which is a number larger than the number of atoms in the observable universe($\sim 10^{80}$). Hence this problem is solved using the method of *column generation* which is a technique that allows you to iteratively improve the solution without explicitly considering all information in the problem. In this section, a conceptual overview of this method is presented and in Section 3 further details and mathematical notation is given. A brief introduction to the proposed idea for how this method could be improved is also described.

The columns when using column generation to solve the CRP are each related to a roster, the rosters are in turn related to decision variables which given the problem formulation we aim to optimize their relationship measured by an objective function. Firstly, in order to obtain desirable mathematical properties, the integer constraint is relaxed such that partially using rosters becomes mathematically feasible. However in the reality, since a crew member either works or does not, this is of course inconceivable. With this relaxation, a problem called the *master problem* (MP) is obtained. Considering all possible rosters is not possible as the number grows combinatorially with the number of tasks. Thus restrictions are imposed on which rosters to use, i.e. only a subset of rosters is considered, in a problem called the *restricted master problem* (RMP). This is a solvable problem but since it does not contain all information it is unlikely that it has a desirable solution. In order to find out how this solution could be improved the so-called *pricing problem* (PP) is solved. This problem utilizes the *dual variables* generated by solving the RMP to compute *reduced cost* which contains information of how the inclusion of a column will impact the solution of the RMP. The column which improves our solution of the RMP the most is included and the process of solving it and the PP is repeated until no further possible improvement can be made. When an unimprovable solution is found it is still possible that the solution is undesirable since the scheme started off with a continuous relaxation. In order to address this, a *Branch and Bound* methodology is implemented which when combined with column generation in this way is called *Branch and Price*. The idea behind this algorithm is to fixate parts of the solution that are non-integer which is known as *branching* and create a search tree. In each node of the search tree, column generation is applied.

In the CRP algorithm used in this thesis, the PP is modelled as a

resource-constrained shortest path problem (RCSPP). The objective of the problem is to for a given crew member find which set of tasks that person should be assigned to improve the current solution the most. If the number of tasks is large this problem can become very slow to solve, hence the algorithm considers only a subset of tasks. The process of choosing this subset is, as often in OR, controlled by heuristics. In this thesis, the idea of using ML to try and find patterns in what tasks are interesting for a given crew member is investigated. In other words, we aim to use a ML model to predict the likelihood of tasks being assigned to a given crew member and then utilize these when picking the subset of tasks to include in the RCSPP. This idea was first explored by Quesnel et al. [18] and our relation to their paper is explained in Section 1.4.2.

The complexity of deciding which crew is likely to be assigned which tasks stem not only from that person's preferences and limitations but also from the interaction between those factors for all crew members. For example how to prioritize a pairing when it looks good in terms of vertical constraints but has other components which are disliked by the crew member? Furthermore, it may be the case that some pairing is disliked by all crew members but it must still be considered in some PPs.

The ML model is trained in a supervised fashion where features are extracted from the costs and constraints in the CRP. The intended use case for the model is that if it is able to learn the behaviour of our problem we could use its predictions in future problems to increase solution speed. As airline planning is often done month by month the model should be trained on previous months problems and output predictions that can be utilized to improve the solving of the upcoming month's problems. As planning problems are solved to or near to optimality the main metric that is aspired to improve is computation time.

In airline planning computation time is of great importance as it allows for more solutions to be computed and more time to be spent elsewhere. The reason finding more and not necessarily objectively better solutions is important stems from the problem complexity which can make solutions with similar objective value completely different in some of the desired metrics. The person responsible for the planning can thus use the extra time to start new problem runs with tweaked input parameters given observations made from earlier runs. Crew can also use the extra time to express their preferences in the bid line approach if that is used. Another benefit from solving the problem quicker is energy savings as the hardware has less work.

When discussing this thesis project it is important to keep in mind that the algorithms at Jeppesen have been developed and tuned for many years which given the heuristic relying nature of the solution process to the very complex problems in question one might implement something that in essence is an improvement but in the context, it does not increase any desired metrics. However, this was taken into consideration when choosing to proceed with this thesis idea as the concept does not directly interfere or change any core features of the algorithm but instead

provides options in a selection problem that already is part of the solution method. Acknowledging this opens up for a discussion that will be continued in the chapter for which this is a critical insight, Chapter 6.

1.3 Limitations

This thesis will be limited to only investigating the usage of ML to accelerate the solving of the PP in the existing framework of Jeppesen's crew rostering algorithm which utilizes column generation. This is to be done by recognizing patterns in which tasks are likely to be operated by a given crew member in a desirable solution. These patterns are to be identified with the use of ML algorithms, more specifically by the training of a neural network.

1.4 Related work

There any many different ways of solving the CRP but as specified in the previous section, the work of this thesis is limited to only investigating Jeppesen's column generation approach. In this section, the aim is to give a background on relevant papers which investigates or discusses similar topics. In the first subsection, a presentation of work related to ML in OR is given and the second subsection contains a more thorough discussion regarding the paper that sparked the idea for this thesis project.

As this thesis involves column generation and the branch-and-price scheme, which is presented with the aim to expand on the concepts and to contextualize in Chapter 3, work that has laid the foundation for these algorithms and also provides a more comprehensive introduction to them is that of Desrosiers and Lübbecke [6] and Barnhart et al. [3] respectively.

1.4.1 Machine learning in operations research algorithms

Figuring out the role of how and where to best utilize ML in OR is still very much a work in progress. Recently papers surveying the current state of research in this middle ground have received a lot of attention. For this thesis which focuses on ML in combinatorial optimization, the most notable is probably that of Bengio et al. [4]. The authors conclude their paper by remarking that they strongly believe in the usage of ML in CO algorithms and that this is the beginning of a new era for CO. Another interesting paper discussing the state of the art for using ML in meta-heuristics (MH) is that of Karimi-Mamaghan et al. [10]. The authors present a categorisation of how ML is used in MH today and use examples of recently published papers to expand on these categories. Categorizing this thesis into any of these does however feel quite forced, the closest related categories are probably *initialization*, as we are precomputing probabilities that aim to find subproblem solutions, or some kind of *fitness evaluation*, as one can view this

thesis as estimating a function which in a similar fashion to reduced costs strive to prune search space.

When utilizing a search-tree creating algorithm like branch-and-price it is interesting to investigate branching strategies, i.e. which subproblem to solve to find the best solution. A strategy known to be reliable is *strong branching* which solves all subproblems and chooses to proceed with the one producing the largest improvement. The reliability of the strategy is counterbalanced by its computation time, in an effort to get the best of both worlds Alvarez et al. [1] investigates approximating strong branching with supervised ML. Another paper that also aims to guide the decision making in the branch-and-price scheme is that of Kruber et al. [15], where ML is utilized to try and predict if it is beneficial to apply a Dantzig-Wolfe decomposition to solve a mixed-integer program and also which decomposition to choose if several options are possible.

An approach that deals with non-integer solutions in the pairing problem is that of Tahir et al. [19] where a method jumping from integer to integer solutions is presented. They improve on previous integer column generation algorithms by utilizing supervised ML to predict pairings that are to be performed consecutively by the same crew member. These predictions are made using a neural network and they are used in a set of reduced subproblems.

A paper that, similar to this thesis, aims to accelerate the branch-and-price algorithm by pruning search space in the pricing problem is that of Václavík et al. [20]. This is done in an online fashion by predicting an upper bound for the subproblem. This upper bound aims to be as tight as possible without making the subproblem infeasible, i.e. pruning as much search space as possible. This is interesting when the subproblem can utilize an upper bound, which is not the case in the algorithm of this thesis as our subproblem is modelled as a resource-constrained shortest path problem.

In the paper by Morabit et al. [16] the authors present a methodology utilizing a graph neural network to predict whether or not generated column in the standard column generation scheme is to be included in the RMP or not. A common strategy in practice when it comes to selecting columns to include in the RMP is to pick a larger number than needed as extra computation time required in the RMP tends not to outweigh the overall improvement in solution speed due to the following iterations of the pricing problem finding increasingly better columns. The column selection strategy presented by the authors is therefore suitable when solving the RMP required more computational time than the PP. Similarly to this thesis, the authors use ML to alter the selection process for inclusion of information in a particular stage of a column generation problem.

1.4.2 Deep-learning-based partial pricing in a branch-and-price algorithm for personalized crew rostering

The current, as of autumn 2021, published paper which is the closest related to the work of this thesis is that of Quesnel et al. [18] which in fact sparked the idea for the direction of the thesis. In this paper, a deep learning neural network is trained for predicting the likelihood that a crew is assigned a pairing which sounds exactly like the core idea of this thesis but there are some key differences. Most importantly the data used in the report is not directly taken from the real world but instead, they derive instances of datasets from Kasirzadeh et al. [12] too which they artificially add language constraints. The crew data is generated using a random procedure similar to the one in Quesnel et al. [17]. Their different instances have the same set of flights and language constraints but each had its own set of crew members. The authors claim that since each instance has a unique set of crew their instances are more dissimilar and thus also more difficult than real-world problems. This claim is very interesting and will be further discussed in Chapter 7. In this thesis, through the collaboration of Jeppesen, access to real-world data have been granted. This data is, because of its complexity (mostly in regards to constraints), substantially different from the data generated by Quesnel et al. and further details about it is presented in Chapter 4.

Another important aspect when discussing the difference between this thesis and the paper by Quesnel et al. is the algorithm in which the predictions are utilized. This is further detailed in Chapter 6 in which methods for prediction utilization that Quesnel et al. did not explore is presented. The algorithm used by Jeppesen for solving CRPs is state of the art and thus naturally explicit details on how it works are not given in this thesis because it is intellectual property. It is interesting to investigate if the more complex algorithm of Jeppesen can benefit as much as the simpler branch-and-price algorithm experimented with by Quesnel et al. It is important to realize that if Jeppesen can utilize ML in this fashion to enhance their algorithm then it seems likely that this strategy has potential to improve other similar solution methods. Worth noting when considering where else this algorithm has potential is that more businesses than airlines solve complex planning problems using similar methodology, for example in the railway industry [9, 8] and scheduling in hospitals [2].

1.5 Thesis outline

In this section, I give a brief overview of what is explained in the different chapters. Firstly a more detailed explanation of the CRP is presented in Chapter 2. Then the method used to solve this combinatorial optimization problem is presented with details on the subproblem in which the ML predictions are to operate in Chapter 3. In Chapter 4 the data used for the ML is presented and in Chapter 5 the ML itself is presented. Chapter 6 details how the predictions made my the ML model are intended to be used. Then a selection of experiments, including the ML model performance, are presented and discussed in Chapter 7.

Lastly, everything is concluded in Chapter 8.

2

The crew rostering problem

The objective of the CRP is to assign work to employees where the work, known as *tasks*, consists of: pairings, training activities, reserves and ground duties. These tasks are to be formed into rosters that are not only legally workable by individual crew members but also desirable for them. The desirability aspect takes into consideration of individual crew members preferences regarding: off-time, colleagues, layovers and more.

In this chapter, the crew rostering problem will be expanded upon utilizing mathematical notation aiming to contextualize and concretize the outline presented in Chapter 1.1.

2.1 Problem formulation

This problem consists of assigning a set of tasks $\mathcal{T}$ to crew members $k \in \mathcal{C}$, where $\mathcal{C}$ is the set of all crew members, such that no tasks are left unassigned, i.e.

$$\mathcal{T} = \mathcal{R}_1 \cup \mathcal{R}_2 \cup \mathcal{R}_3 \cup ... \cup \mathcal{R}_{|\mathcal{C}|} \tag{2.1}$$

Here $\mathcal{R}_k$ represents a roster or scheme of which tasks are assigned to crew member k. Assigning a crew member to a roster is associated with a cost c_k and the objective is to minimise the sum of these costs. The CRP is in compact form thus expressed in the following way:

$$\begin{aligned} \min \quad & \mathbf{c}^\top \mathbf{x} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} = \mathbf{1} \\ & \mathbf{x} \in \{0,1\}^n \end{aligned} \tag{2.2}$$

The columns $\mathbf{a}_i \in R^{m \times 1}$ of the matrix $\mathbf{A} \in R^{m \times n}$ represents a roster for some crew member. The binary variable x is a decision variable that decides if this roster is to be used or not. To deal with both choosing a crew member and selecting tasks, the columns consists of two parts:

$$\mathbf{a}_j = \begin{bmatrix} \mathbf{e}_k \\ \mathbf{p}_j \end{bmatrix}, \quad j \in \mathcal{J}_k \tag{2.3}$$

where $e_k \in \{0,1\}^{|\mathcal{C}|}$ is a unit vector with 1 at position k and p_j is a binary vector. Hence the first $|\mathcal{C}|$ rows of the constraint matrix $\mathbf{A}$ are called *assignment constraints* since they ensure that each crew member is assigned tasks. The following $|\mathcal{T}|$ number of rows are called *activity constraints* since they express which activities the selected

crew member are to perform. Thus the **1** on the right-hand side of the constraints in Problem 2.2 demands that both all crews are to be assigned tasks and all tasks are to be assigned exactly once. The set $\mathcal{J}_k$ contains the indices corresponding to the potential schedules of crew member k, which means that the set has the following mathematical properties:

$$\mathcal{J} = \bigcup_{k \in \mathcal{C}} \mathcal{J}_k = \{1, 2, ..., n\} \tag{2.4}$$

and

$$\mathcal{J}_k \cap \mathcal{J}_{k'} = \emptyset, \quad \forall k \neq k', \quad k, k' \in \mathcal{C} \tag{2.5}$$

where $n = |\mathcal{J}|$ is the total number of variables in the Master problem which corresponds to all possible rosters for all crew in the CRP. The simple Model 2.2 only considers constraints related to one roster. It could be of interest to add constraints on more than one roster, those kinds of constraints are called *vertical constraints*. An example of this is crew complement where the goal is to allow for tasks that need more than one crew to be operated, the model is then extended to:

$$\begin{aligned} \min \quad & \mathbf{c}^\top \mathbf{x} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \in \{0, 1\}^n \end{aligned} \tag{2.6}$$

where $\mathbf{b}$ is a vector of positive integers. Of course for elements $i = 1, 2, ..., |\mathcal{C}|$ elements $\mathbf{b}_i = 1$ since these still should correspond to how many rosters any given crew is allowed to operate. The assignment constraints remain but the activity constraints are now allowed to ask for more crew to operate the activities. For example if task $t \in \mathcal{T}$ requires two crew members then $b_{|\mathcal{C}|+t} = 2$. Further extensions such as: qualification constraints, pilot experience, certain crew must fly together and more are presented by Kohl and Karisch in [14]. The problem in Equation 2.6 forms an IMP which we solve using the branch-and-price algorithm presented in Section 3.2. The pricing problem which is the interest of this thesis thus consists of generating columns or possible rosters for each individual crew member.

3

Column generation for large scale problems

In this chapter, we expand on the theory behind the solution method commonly utilized when solving the CRP such that the context of where our ML is implemented is clarified. In order to do this, we start by explaining the scheme of column generation. Then that theory is related to the branch-and-price algorithm which is a union of column generation and a typical branch-and-bound algorithm. Lastly, we focus on the specific subproblem in which our ML-models predictions will operate.

3.1 Column Generation scheme

Consider a general linear problem with m constraints and n decision variables:

$$\begin{aligned} \min \quad & \mathbf{c}^\top \mathbf{x} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq 0 \end{aligned} \tag{3.1}$$

where $\mathbf{c} \in R^n$, $\mathbf{x} \in R^n$, $\mathbf{A} \in R^{m \times n}$ and $\mathbf{b} \in R^m$. In the context of column generation this problem is called the *master problem*(MP) and due too n being large enough to make considering all columns an inconceivable option we restrict this number by only considering a subset. This is the idea behind the *restricted master problem*(RMP) which we can formulate as:

$$\begin{aligned} \min \quad & \sum_{j \in \mathcal{J}'} c_j x_j \\ \text{s.t.} \quad & \sum_{j \in \mathcal{J}'} \mathbf{a}_j x_j = \mathbf{b} \\ & x_j \geq 0 \qquad \forall j \in \mathcal{J}' \end{aligned} \tag{3.2}$$

where $\mathbf{a}_j \in R^m$ are the columns of the $\mathbf{A} \in R^{m \times n}$ and x_j, c_j are elements of $\mathbf{x} \in R^n$ and $\mathbf{c} \in R^n$ respectively. Most importantly $\mathcal{J}' \subseteq \mathcal{J}$ which allows us to find solutions to this problem with for example the simplex method [11, pg.23]. The concept of the simplex algorithm is to move in steps between what is called basic feasible solutions until no step improves the current solution and thus the optimal solution is found.

It relies on a starting point, a basic feasible solution $\mathbf{x}_B$ that satisfies all constraints i.e. $\mathbf{Ax} = \mathbf{b}$ and $\mathbf{x} \geq 0$. Using this solution we can compute the dual variables: $\boldsymbol{\pi}^\top = \mathbf{c}_B^\top B^{-1}$ where the basis matrix $\mathbf{B}$ and the cost vector $\mathbf{c}_B$ is formed by the columns related to $\mathbf{x}_B$. These can then be used to compute the reduced costs for a given variable i as follows:

$$\bar{c}_i = c_i - \boldsymbol{\pi}^\top \mathbf{a}_i \tag{3.3}$$

The reduced cost gives information about how the objective value would change if column i is included in the basis B. The index i could thus be any $i \in \mathcal{J}$ where $\mathcal{J} = \{1, 2, .., n\}$. The same concept is used when deciding what columns to include in our RMP. The problem of finding which columns are related to the greatest improvement in our objective function is then formulated as follows:

$$\bar{c}_j^* = \min\{c_j - \bar{\boldsymbol{\pi}}^\top \mathbf{a}_j \mid j \in \mathcal{J}\} \tag{3.4}$$

which is known as the *pricing problem* (PP). Because the RMP was formulated as a minimization problem the only columns of interest are those with negative reduced costs, i.e columns that reduce our objective value. This enables mathematical identification of no possible further improvement as $\bar{c}_j^* > 0$. These formulations can be made without loss of generality as with a maximization problem the signs can simple be flipped.

The solution procedure is thus iterating between solving the RMP to get the dual variables which then can be used in the pricing problem to find new columns to include in the RMP. You repeat these steps until no column inclusion can improve the objective value and hence the optimal solution is found. In Figure 3.1 one can observe the same relationship between the MP, RMP and PP as described in this section but with the extension of the algorithm explained in the related section, Section 3.2.

3.2 Branch and Price algorithm

When looking for integer solutions to very large scale optimization problems one commonly combines the method of column generation with the branch-and-bound algorithm. This idea was first proposed by Barnhart et al. [3] and is called the branch-and-price method. Consider the integer minimization problem:

$$\begin{aligned} \min \quad & \mathbf{c}^\top \mathbf{x} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \in \{0,1\}^n \end{aligned} \tag{3.5}$$

In the scheme of branch-and-price this problem is called the *integer master problem* (IMP) and with relaxation of the binary constraints on the variables the master problem, presented in Equation 3.1 in obtained. This is then solved as discussed in the related Section 3.1. If the solution is integer the solution satisfies the IMP and hence the algorithm stops, but if it is non-integer some branching rule is applied which in our context commonly mean fixation of a subset of variables as follows:

$$\sum_{i \in \alpha} x_i = d, \quad d \in \{0,1\} \tag{3.6}$$

where $\alpha \subseteq \mathcal{J}'$ i.e a subset of the variables considered in the current iteration of the RMP. d represents the decision of fixating the variables which means forcing the rosters of α to either be used or not. When searching the created search tree related to these constraints column generation is used at every node. The complete scheme of the algorithm is illustrated in Figure 3.1.

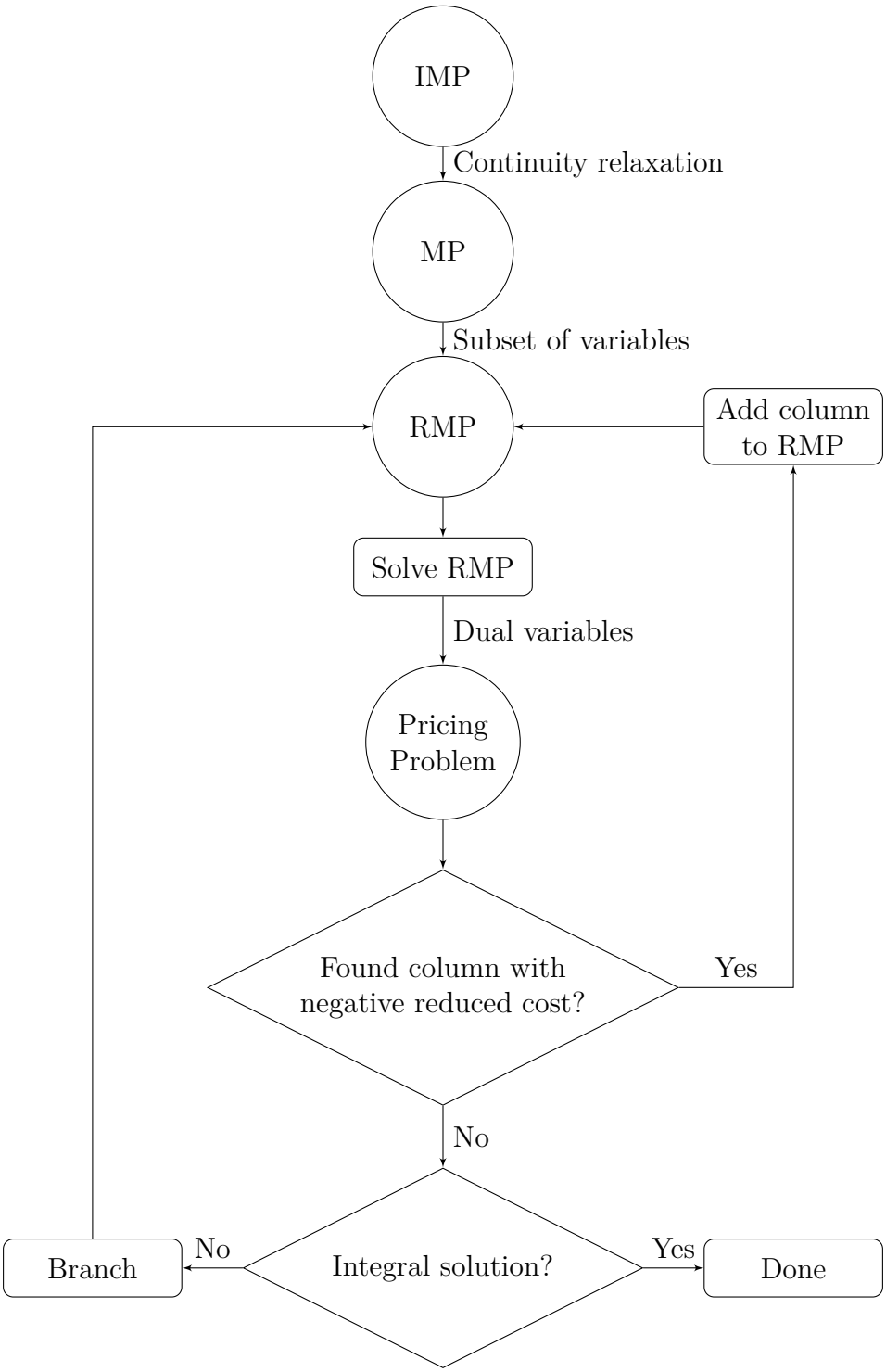

Figure 3.1: Diagram of the branch-and-price scheme.

3.3 Pricing problem

When solving the pricing problem, which in our CRP context means finding a roster for a given crew member that improves our objective value, there are three methods that are frequently used. Which one out of the three is chosen for a given iteration is commonly iteration-depth dependent. The first method, which we will call *simple assign*, is used in early iterations when assigning any legal tasks to a crew member improves the objective value. This method simply picks a selected number of tasks that has the lowest reduced cost and tries to assign them to a crew member. The second method creates a graph network with which a *resource-constrained shortest path problem* is formulated. This method is better suited for finding adjustments to a roster and how it works is detailed in the following section. The final method is a local search that deals with fine-tuning and is used when the other methods struggle to find legal rosters. Since this method only cares about finding similar rosters to what has already been found it does not seem appropriate to guide it away from this purpose with probabilities. Hence the two methods of interest are simple assign and the shortest path method. In these two methods which utilize a reduced costs based selection when choosing which tasks to consider this thesis project aims to change the selection such that they utilize ML predictions instead. This may seem naive at first as the reduced cost directly correlate with improving objective value and should hence be a good indicator of what column to include but because of the way rule modelling is handled the reduced cost disregard important constraints that may make the roster illegal. On top of that, the reduced cost does only take information in the current iteration of the RMP into account. Thus there might be potential in ML predictions which could learn from information not considered in the reduced cost.

3.3.1 Resource-constrained shortest path problem

When solving the pricing problem, which in the CRP means finding a roster that when included improves the cost of our current solution, we can formulate a RCSPP of which the goal is to construct such a roster. The core idea is to formulate a RCSPP on a directed acyclic graph/network in which nodes represent actions and edge represents related costs. Finding a solution, i.e. which actions are to be done and in what order, to such a network is a potential roster.

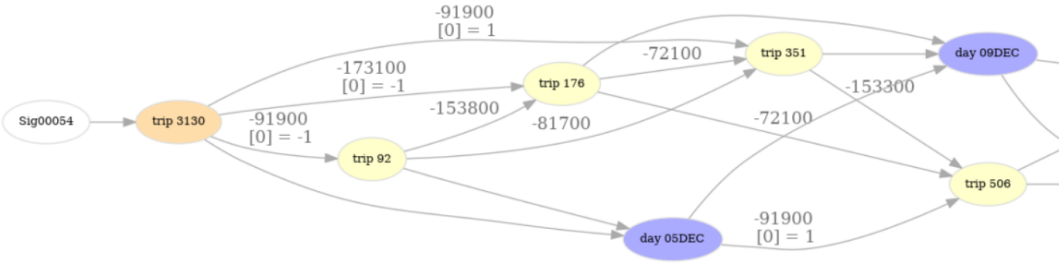

Figure 3.2: First part of a RCSSP network diagram.

The nodes in Figure 3.2 represent either the passing of time without work or the assignment of work in the form of a pairing. The time passing nodes are blue and there are two different assigned work-nodes: there are nodes that and compulsory (orange) and work-nodes that are optional to assign (yellow). All networks are crew member specific and hence the edge costs represent an aggregation of the crew's individual preferences as well as the actual cost of assigning the pairing. If the solution path contains a sequence of time passing nodes that means the crew in question will have time off during the period which those nodes represent. In the figure, a lot of time-nodes have been pruned by network optimization.

When deciding which pairings to consider in the RCSPP different heuristics are used. Usually, this selection process is dominated by one cost, the reduced cost. An important parameter when building the network is how many pairings to include. This is a fine balance act as including too many nodes will slow down the computation and including too few could make us not find a roster that improves our RMP solution. The size of how many pairings to include is commonly controlled by heuristics. In Chaper 6 we discuss further details on how the predictions are to be utilized.

4
Data

In this chapter, we will discuss what data is used and how it was collected. The keyword for this section is that the data is collected from the *real world* which is possible since this thesis is done in collaboration with Jeppesen. However, with respect to GDPR, we will keep details that risk disclosing private information anonymous or rounded.

In order to replicate how this method would be used in the real world, it is important that the data is collected from two consecutive planning periods, in airline planning a period is commonly equivalent to a month. The reason this is important is because the intended use case of the ML model is to predict future planning in a supervised ML fashion by learning from previous plans. Naturally, the plans do not have to be perfectly consecutive for this but since change usually happens slowly in big airlines which should make consecutive months similar it seems reasonable to utilize at least the most recent month to plan ahead. However, it may be that for example for a holiday month like December when airline travel pattern changes it may not be optimal to learn from the neighbouring month but instead perhaps from the same month last year. In practice airlines have access to more training data than just the single previous month therefore our data set is likely to be more difficult to predict as our data set only contain two months.

The data sets have several variables which are all extracted with consideration of how to capture the behaviour of our problem, this is further discussed in Chapter 5. Each sample represents the relation between a task and a crew member thus the data set could have up to $|T| * |C|$ number of samples where the terms are number of tasks and number of crew. However, all tasks are not feasible for all crew members thus the total number of data points is much lower. All samples have two labels out of which of course only one is used at a time. The first label type is a binary label which represents if the task is assigned the specified crew member in any of the solutions or not. The second label type is a ratio between zero and one which represents how frequent among a selection of solutions a task is assigned a specific crew member. These two can be mathematically defined as follows:

$$y_1^{kt} = \min(0, s_i^{kt}) \quad \forall i \in \mathcal{S} \tag{4.1}$$

$$y_2^{kt} = \frac{\sum_{i \in \mathcal{S}} s_i^{kt}}{|\mathcal{S}|} \tag{4.2}$$

where y_1^{kt} and y_2^{kt} are the two labels respectively, $\mathcal{S}$ is the index set of selected solutions and $s_i \in \{0, 1\}$ is a binary variable representing if the task is assigned to the crew member in solution i or not. The set $\mathcal{S}$ is built up by solving the problem using Jeppesen's untouched CRP algorithm where all solutions are the best solution found given different randomization seeds.

The data set is from a major airline's planning period of two consecutive months, namely December and January which as mentioned may be one of the more difficult periods to plan. For the two months, out of which the ML model will be trained on the first, 15 solutions have been computed and processed into the labels formulated in Equation 4.1, i.e. $|\mathcal{S}| = 15$.

4.1 Month 1

If we look at Figure 4.1 we see the distribution of the binary labels in our training data. It is clearly unbalanced (about 10% of samples have $y_1 = 1$) which is something that has to be dealt with when training ML models on the data, this is expanded upon in Section 5.3. This unbalance is expected as many tasks are likely comparatively inefficient to assign a crew member and they may also not match the crew members preferences.

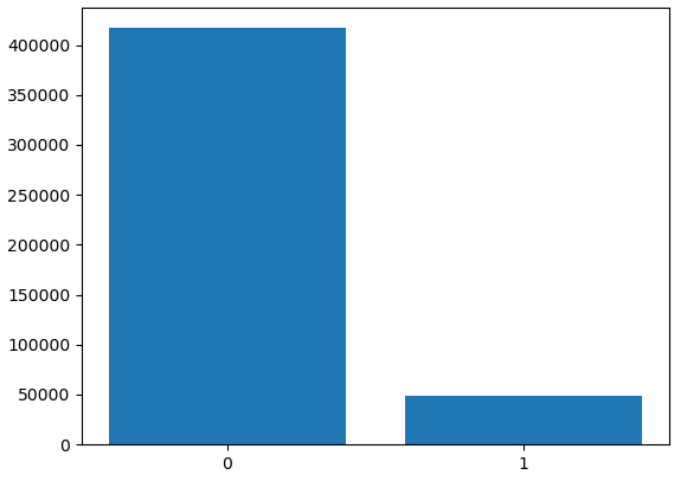

Figure 4.1: Plot of binary label distribution in training data.

Looking at Figure 4.2 we see that the majority of non-zero labels are corresponding to tasks that are assigned to a certain crew member only once in all of the solutions. This is good as it indicates that the solution set contains dissimilar solutions, something especially important when it comes to gaining general knowledge about the compatibility between crew members and tasks which is the exact purpose of our ML model.

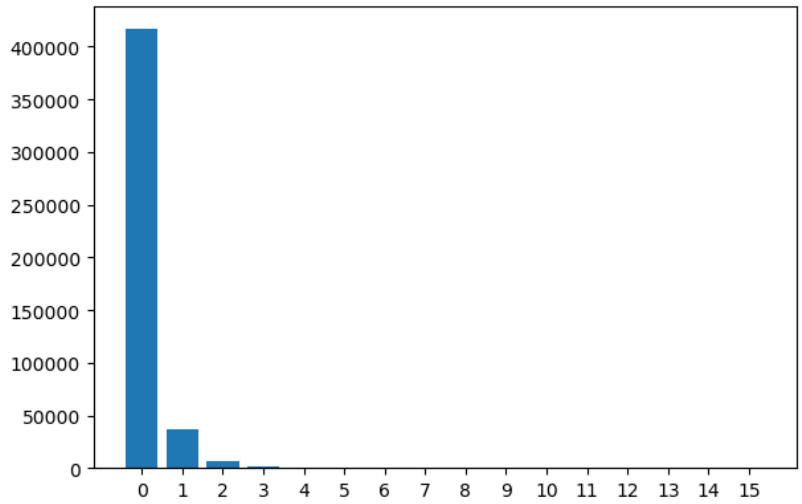

Figure 4.2: Plot of inverted ratio-label distribution, i.e. how frequently a task was assigned to a crew member in all 15 solutions.

4.2 Month 2

Looking at Figure 4.1 we observe that this month is also very unbalanced, which again is expected. However this month contain a lot more data points, 650 000 compared to 470 000. This indicates that the two months are dissimilar planning periods which will be more difficult for the ML model but also makes the problem relevant as if the model manages to learn it seems likely that this approach could be used successfully on simpler planning periods.

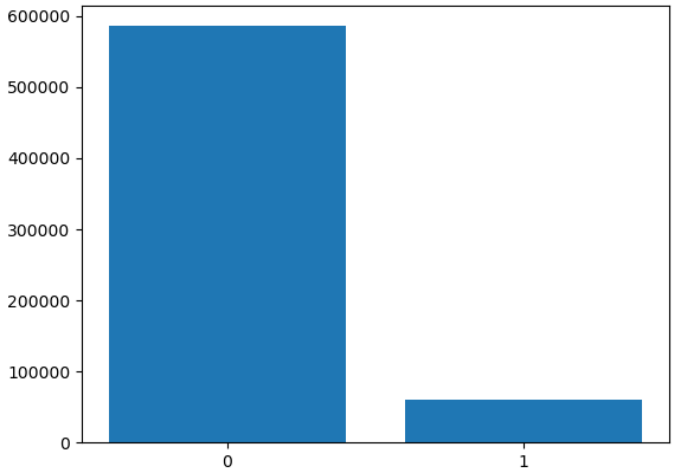

Figure 4.3: Plot of binary label distribution in testing data.

In Figure 4.4 the distribution of also this month shows that there seem to be many ways to construct rosters in a near-optimal way. Thus a ML model could potentially identify patterns in what these tasks have in common.

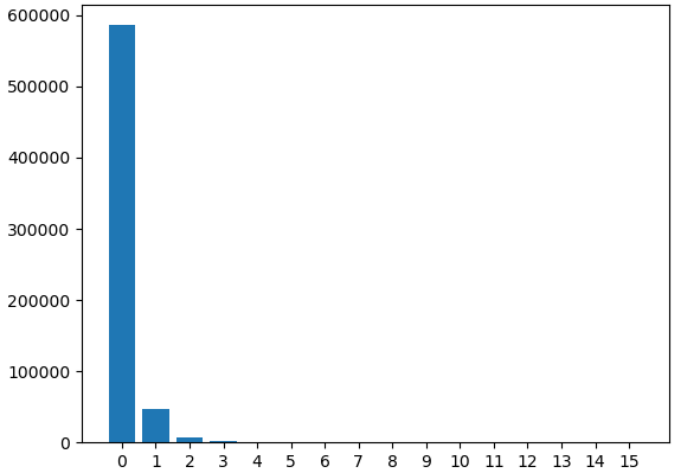

Figure 4.4: Plot of inverted ratio-label distribution, i.e. how frequently a task was assigned to a crew member in all 15 solutions for testing data.

5
Machine learning

In this chapter, the ML model and its features will be discussed. In the first section, a description of intentions is given to further clarify the context in which the model will operate. In the following section, the selected features are presented along with the motivation of why they should capture the behaviour of our problem. Lastly, we express details and parameters investigated when training the deep neural network which is the main ML model of this thesis project.

5.1 Overview

The ML model will operate offline in a supervised fashion by utilizing selected features from a planning problem and a given crew members competencies and preferences to predict the probability that a specific task is assigned to the given crew member. Therefore this predicted probability can also be interpreted as a degree of compatibility between the crew member and the task. These probabilities are then used to choose which tasks should be included in the PP. The selection process for task inclusion in the PP is thus simple: sort the tasks in descending probability order and pick the first ones. Further details on the different strategies for exactly when the probabilities are used is explained in Chapter 6.

The data set contains two consecutive months and the aim is to train the model on the first month and test it on the following month. By having a data set consisting of two parts like this the process of splitting the data into test and training becomes simple as it aims to reflect the real-world scenario where this method is intended to operate.

5.2 Features

In this section, the features will be presented but only to the extent that the intellectual property limits of Jeppesen allow. The approach for how to extract features is however freely discussed and this is arguably the most interesting aspect.

The features are extracted by outputting data from the CRP algorithms formulation. If we look back at problem Formulation 2.6 there are two main categories of information that could be of interest: the costs and the constraints.

The cost is an aggregation of the true costs; for example, layover costs like hotel nights and transportation, but also fictional costs like crew preferences and solution robustness. These are common factors in the cost but naturally different airlines have different prioritization. Since the aggregated cost is handled by the rule modelling language Rave extracting the separate components is difficult and since it may also be airline-specific it is most general to use the complete aggregated cost. Thus the first feature is a cost that takes both fictional and true costs into account.

Secondly, we should include information about how the tasks relate to the limitations and expectations on the roster. This is mathematically expressed by the constraints. Since the aim is to predict for each crew and task combination the constraints of interest are the activity constraints, as these relate the assigning of a task to constraints for each crew. An example of a constraint is the seniority rostering constraint which imposes that an inexperienced crew member must always be assigned tasks together with a senior crew member. In total there are 18 constraints in common between the first and second month in the data set, thus 18 constraint features are extracted.

Lastly, we need a feature that can relate tasks to how big of a commitment assigning of them are. For this purpose, the task duration is extracted. This feature could help prevent assigning one long task instead of assigning two shorter tasks that together are better but worse separately. This leads to a total of 20 features.

5.3 Model training and architecture

In this section, we present the hyperparameters of the network and model training procedure. All training was implemented in Python using the PyTorch library. All training is done in a supervised fashion where month 1 is used as training data and month 2 is used as test data, see Chapter 4 for details. In order to deal with the heavily unbalanced data oversampling is used such that non-zero labels represent 40% of the total data. The network is a feedforward fully connected deep neural network and conforms with standard practice by having: the size of neurons in the hidden layers decrease from input to output, the activation functions between the layers are all the rectified linear unit functions (ReLU). The output layer uses a sigmoid function such that the prediction is on the interval [0,1]. Besides the mean squared error (MSE) loss function, Binary cross-entropy (BCE) is tested which is usually used for binary classification but does also work for our prediction interval. The optimizers experimented with are the classic stochastic gradient descent (SGD)[5] and Adam[13]. The regularization method used to deal with overfitting is called drop out and works by giving all neurons a probability to be ignored in the training with the current batch. All these hyperparameters are tested in an experimental fashion where a bunch of models are trained and parameters of models that perform well are further investigated. In training network size was gradually increased, starting off as linear regression. The reason behind investigating using a deep neural work is because it was proved to successfully

capture the behaviour of a similar problem by Quesnel et al. [18]. The range of the hyperparameters is stated in Table 5.1.

Hyperparameter	Range
Number of hidden layers	{2, 3}
Size of hidden layer 1	[50, 1000]
Size of hidden layer 2	[50, 500]
Size of hidden layer 3	[20, 200]
Optimizer	{SGD, Adam}
Loss function	{BCE, MSE}
Drop out probability	[0, 0.3]
Learning rate	$[10^{-6}, 10^{-3}]$
Batch size	[50, 200]
Batch normalization	[yes, no]

Table 5.1: Hyperparameters of deep neural network.

To measure the performance of a model we use, besides loss, top-k-score which we define as follows:

$$\text{top-k-score}^k = \frac{\sum_{i \in \mathcal{A}_t} y_i^k}{\sum_{i \in \mathcal{B}_t} y_i^k} \tag{5.1}$$

where $\mathcal{A}_t$ is the index set of the first t elements in a list of the binary label y_i^k sorted by the predicted probabilities and $\mathcal{B}_t$ is the set of the first t elements in a list of the binary label y_i^k sorted by the true labels. Both lists are sorted in descending order. This metric thus describes how many of the first t tasks that we predict to be in a solution are actually in a solution and operate on the range [0,1]. If top-k-score = 1 then the model has made perfect predictions.

6
Prediction utilization

In this chapter, we discuss how to utilize the predictions generated by our ML model in the designated pricing problem subproblems, simple assign and the RCSPP.

Let's first consider when it is most appropriate to utilize ML predictions instead of reduced costs. The strength of reduced costs is that they are similar to that of a gradient in the way that they point us in a direction, a direction of improvement. The weakness is that the point from which they declare this direction may be a poor starting point as the reduced costs are computed with information from the RMP which especially in the early iterations is very limited. Therefore an interesting hypothesis to test is if utilizing the ML predictions instead of reduced costs is most beneficial in the early iterations.

6.1 Simple assign

In the early iterations of column generation, the pricing problem is solved using the Simple assign method which attempts to greedily assign tasks to the given crew member based purely on reduced costs. When utilizing the ML predictions we adopt the same framework. The tasks are picked in order one-by-one from Θ which is a task list sorted by predicted probabilities in descending order. How many tasks are considered in the subproblem is controlled by a heuristic but if the predictions are reliable then perhaps it could be interesting to lower the size of many tasks to pick. But as sorting and picking the tasks are not very computationally heavy operations this may not be as interesting as in the other, more complex, subproblem.

6.2 RCSPP

In the PP the RCSPP is utilized after the early simple assign iterations unless it struggles to produce legal rosters, in that case, a local search approach is applied instead. Setting up and solving the RCSPP is a relatively large part of the computation time in the pricing problem, which in turn is a large part of column generation.

If our ML model is capable of finding tasks that are more deeply compatible with crew than reduced costs indicates then the RCSPP network size could potentially

be scaled down. This is interesting to investigate as the size of the network is naturally closely related to the computational time of setting up and solving the RCSPP. There are different ways to implement network size reduction. Currently, the size is controlled by heuristics. It could be interesting to only include tasks that have predictions better than some threshold as this would ensure only good predictions are considered in the subproblem. But if the model struggles to output high probability predictions this could lead to some crew having very small networks. Another strategy is to lower the limit of the maximum allowed tasks to be considered in the subproblem. This approach doesn't have the problem that the network would be empty if no probabilities are high. If the model is accurate then this seems like a reasonable approach as it could be expected to be able to have a smaller number of probable tasks than tasks with low reduced costs (as, especially in early iteration, many assignments of work are likely to improve the current objective value).

Even if the ML predictions are used to decide which tasks to include in the RCSPP the reduced cost is still an important factor of the problem as they are part of the aggregated arc costs. Thus the ML predictions can include tasks that are to be in the optimal roster for a crew member but these tasks may not be included in the solution roster because their respective arc costs are not good enough. A potential strategy for dealing with this is to manipulate the arc costs so that they take the prediction into account. This problem is important to remember as it may help describe odd results when experimenting with using accurate predictions.

7
Results and Discussion

In this section, a selection of experiments is presented and discussed. In the first section, experiments are conducted without ML to provide a proof of concept in a base case scenario. In the second section, the ML training and performance is analyzed.

7.1 Best case scenario

In this section, no experiments use any machine learning instead we extract perfect predictions which are computed directly from the column generation solution to the problem. This is done as a proof of concept, i.e. to test if the ML predictions are perfect is this able to improve the algorithm. Perfect predictions are identified as follows:

$$p_i^k = \begin{cases} 1, & \forall i \in \mathcal{S}_k \\ 0, & \forall i \notin \mathcal{S}_k \end{cases} \tag{7.1}$$

where $\mathcal{S}_k$ is the index set of tasks assigned to crew member k in a solution to the selected problem.

In the experiments of this section, multiple converge graphs of column generation are presented. In these graphs, there are symbols relating to what kind of solution the algorithm finds. For our purpose, the symbol of interest is the circle which represents the finding of an integer solution. In these plots, there are also always two graphs where the green one is the standard run without any altering done. The red graph is always the one related to the experiment as it shows how the behaviour of the algorithm changed with the changes described in the related subsection.

7.1.1 Always using perfect predictions

In this section, we present the results from the first month in data set 1 when perfect predictions are fed into the algorithm. These perfect predictions are computed as detailed in Equation 7.1 and utilized as described in Chapter 6.2. The size of the subproblems along with when which type of pricing method is used when remains unchanged for this experiment.

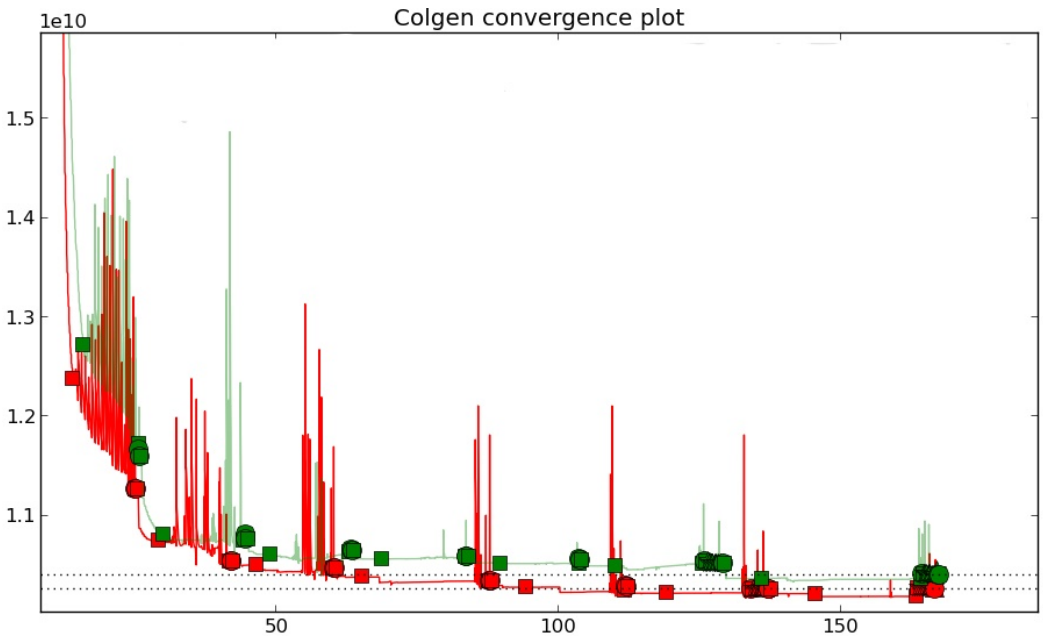

Figure 7.1: Convergence plot of an algorithm utilizing perfect predictions at all times in red and original algorithm in green. X-axis is execution time and Y-axis is cost.

As Figure 7.1 shows the utilization of predictions can make things better and not necessarily quicker. The reason for it not being quicker is likely because the subproblem size is kept the same thus the RCSPPs are of equal size and thus take similar computing time. Another potential reason, discussed in Chapter 6, is that the arc costs of the network are still reduced cost based which when combined with larger networks allows the RCSPP to not generate rosters including the probable tasks.

7.1.2 Always using perfect predictions and limiting network size

In this experiment, the limit of how many trips are allowed in the network is reduced to about half of the standard value. Perfect predictions are utilized at every iteration.

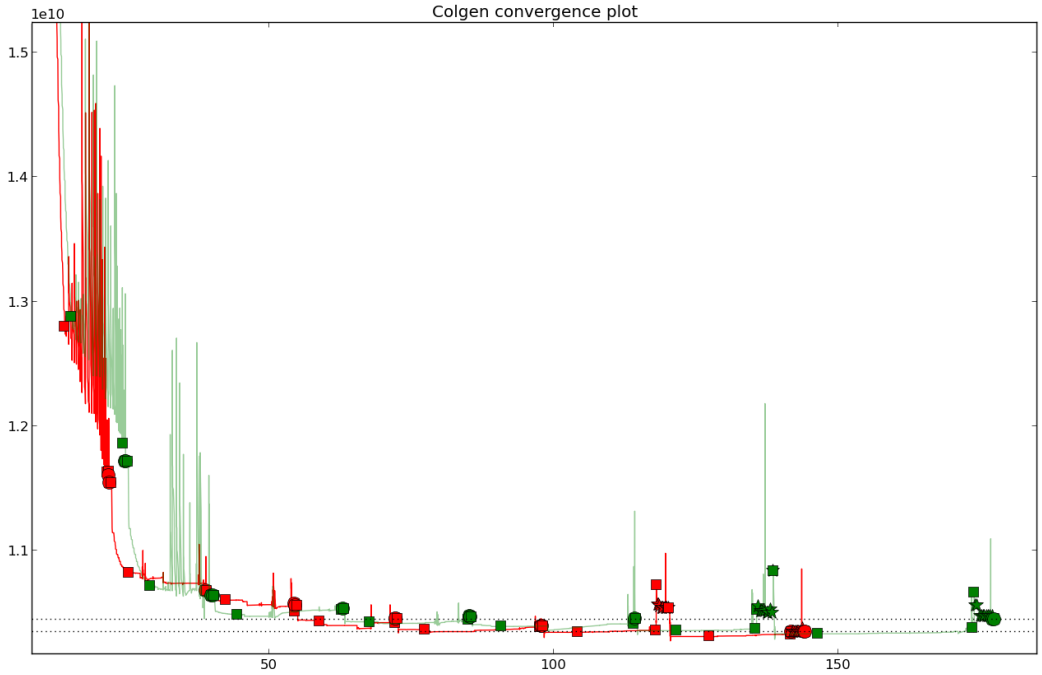

Figure 7.2: Column generation convergence plot of using perfect predictions and lowered limit on maximum allowed tasks in RCSPP. X-axis is execution time and Y-axis is cost.

In Figure 7.2 we observe a decrease of about 20% in computational time. On average over 15 runs there was an increase in speed of 12.5% and 0.356% better solution was found. This is interesting if we compare with the results in Section 7.1.1 we can confirm that limiting network size is in fact useful for speeding up the problem run.

7.1.3 Always using perfect predictions and minimally sized subproblems

This subsection presents the result from an experiment where we push the minimization of network size to the maximum. This is done in order to investigate the importance of that parameter. The parameter is changed as follows:

$$SP_k^{min} = \left| \left\{ \{p_i^k\} \middle| p_i^k > 0, i \in \{1, 2, ..|T|\} \right\} \right| \tag{7.2}$$

which means that in any crew members subproblem only tasks that have a prediction > 0. Since perfect predictions are being used i.e. $p_i^m \in \{0, 1\}$ this means that the size of the subproblem is minimal while considering all tasks in the solution.

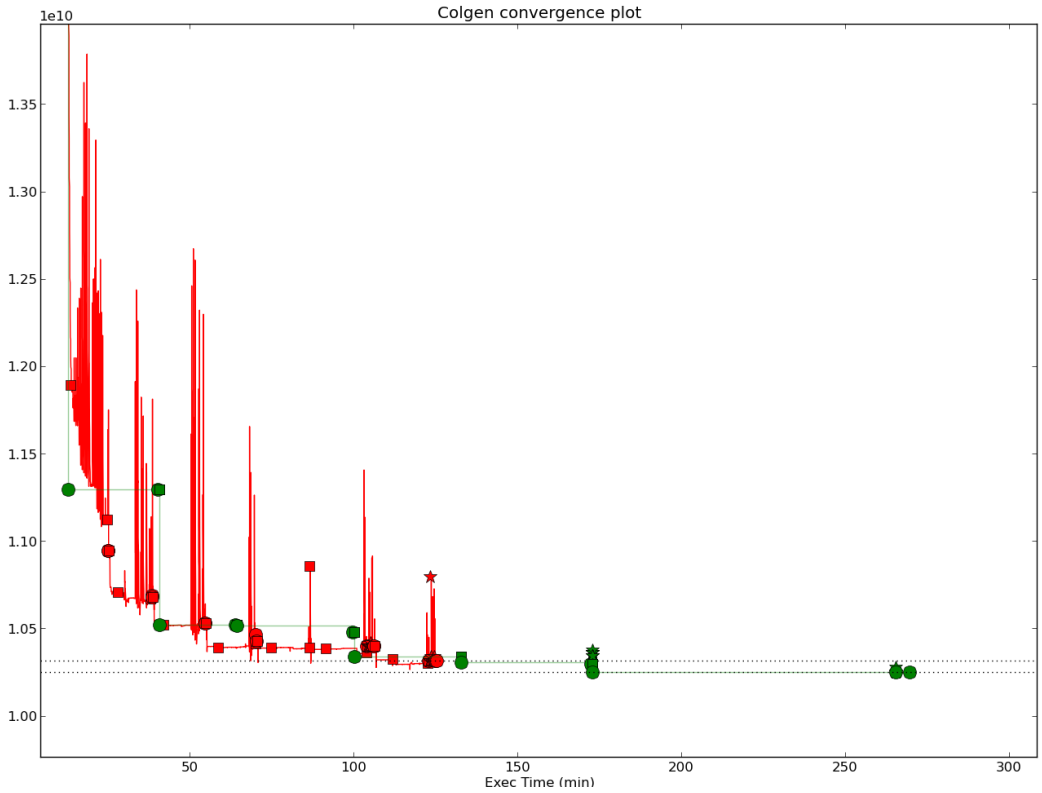

Figure 7.3: Perfect predictions and only tasks with $p_i^m = 1$ are included in the network. X-axis is execution time and Y-axis is cost.

In this graph we observe the solution is slightly worse but it is found much quicker. A likely reason for not finding the solution used to make the predictions is that the heuristics that control when to stop the search thinks that too little improvement is made or can be made. Thus we recognize that finding a balanced network size is important.

7.1.4 Perfect predictions and limited network size for 50 iterations

In this experiment, the perfect predictions are use utilized in a way that was hypothesized before implementation to be the most probable way one could utilize predictions. That is, as further detailed in Chapter 6, by using the predictions only on the early iterations and limiting the network size while the predictions are utilized. Predictions are thus utilized in both simple assign and RCSPP.

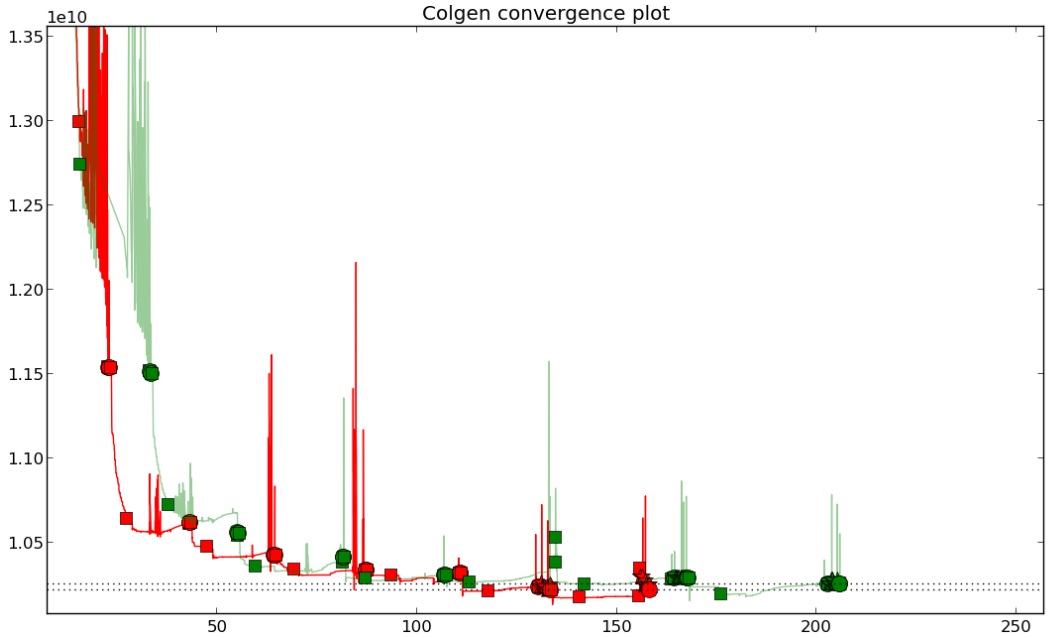

Figure 7.4: Column generation convergence plot of using perfect predictions and lowered limit on maximum allowed tasks for 50 iterations in the pricing problem. X-axis is execution time and Y-axis is cost.

This experiment resulted in an average increase of 6% solution speed over 15 problem runs with different randomization seeds. However notable about the performance in those 15 runs are that either there was no change or a 20% increase in computation speed. As Figure 7.4 shows utilizing the predictions in the early iterations can allow for a more stable solution process that converges earlier and notably also does not converge to worse solutions.

7.2 ML model training

In order to first verify that the model is able to learn from the data, we conduct experiments where the aim is to overfit a model. This can usually be done by making larger networks and not utilizing any regularization methods. Table 7.1 shows the hyperparameters of one of the models which performed the best in terms of training loss and training top-k-score. Note that as this top-k-score was computed on the batches it no longer is reasonable to compute it per crew as our data set contains around 700 crew members and the batch size is always smaller than that. This should make it easier to get a god top-k-score as when sorting only the most probable tasks will affect the top k-score and obviously, there are more tasks with higher probability with a larger number of crews.

Hyperparameter	Model 1
Number of hidden layers	3
Size of hidden layer 1	800
Size of hidden layer 2	400
Size of hidden layer 3	200
Optimizer	Adam
Loss function	BCE
Drop out probability	0
Learning rate	10^{-5}
Batch size	100
Batch normalization	yes

Table 7.1: Hyperparameters of model 1.

As can be seen in Figure 7.5 the model rapidly converges to a noisy level of training loss. In other models where instead the SGD was used to optimize the conversion was slower but to similar values, i.e. 0.33 with lots of noise. In Figure 7.6 we see that even though this top-k-score should be higher than the crew specific top-k-score it is very low. As the model struggles to learn the training data it seems unlikely that it would perform well on the testing data. This hypothesis is verified in Figure 7.7 where the test loss barely changes as the model learns. The test and training loss are very similar but the test loss is much less noisy. The test top-k-score can be seen in Figure 7.8 and shows consistent improvement over the training period but very little in total. To test the actual purpose of the model predictions on the second month is made and compared with randomly ordering the tasks instead of ordering them by probability. This comparison confirms that there is no significant improvement in using probability sorting in terms of top-k-score.

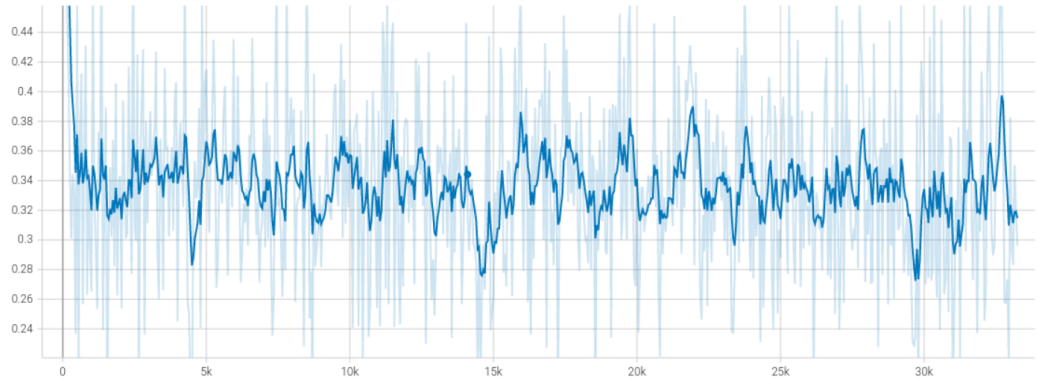

Figure 7.5: Training loss for model 1. Dark blue graph is the smoothed curve and light blue is unsmoothed.

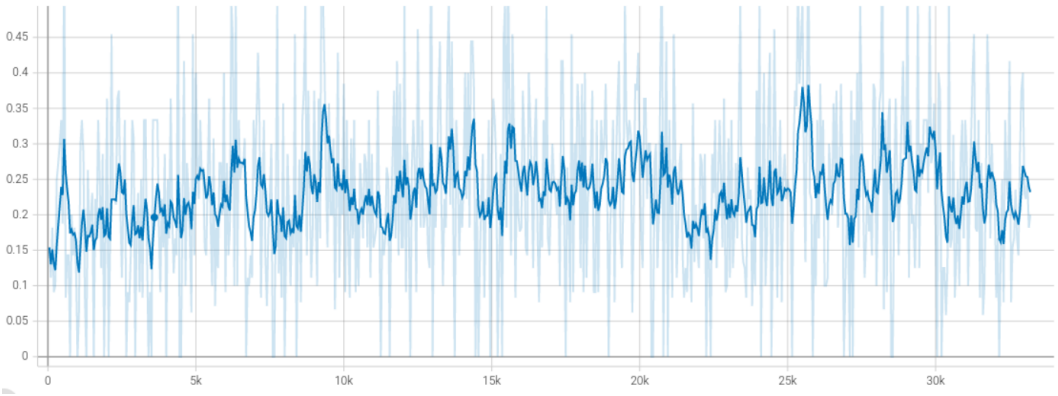

Figure 7.6: Top-k-score of batch for model 1 with k = 20%. Dark blue graph is the smoothed curve and light blue is unsmoothed.

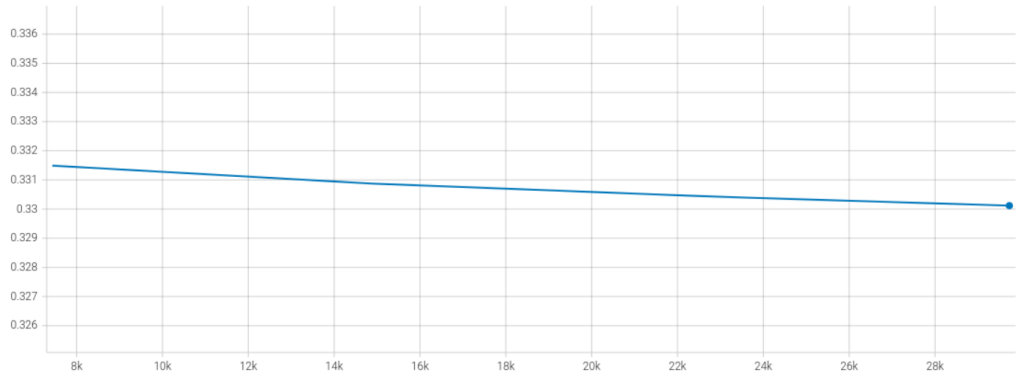

Figure 7.7: Testing loss for model 1.

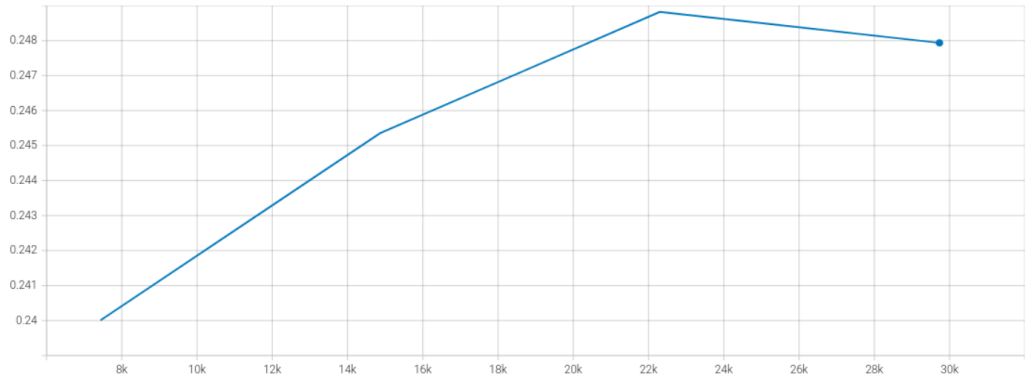

Figure 7.8: Batch top-k-score for model 1 on test data with k = 20%.

There are two possible explanations for none of the ML models being able to properly learn the behaviour of the data. The first explanation is that the complex relationship that exists between the features are not capturable by the choice of model. It could be that the deep learning approach to this problem is not able to identify the patterns in the data and thus for further research on the topic we recommend investigating other methods as well. The second explanation is that perhaps there exists no, however complex, the relationship between our features and labels. If this is the case perhaps more features could be extracted. It is also possible that there is some combination of the two. Comparing these results with the paper of Quesnel et al. [18] one might conclude that perhaps the key difference is that their data has a lower degree of complexity compared to that of data from the real world.

8

Conclusion

The problem in which the thesis operates is the crew rostering problem and the solution algorithm of that problem is provided by Jeppesen, with which this thesis was made in collaboration. Crew rostering concerns constructing rosters or personal work schedules for individual crew members and is modelled as a combinatorial optimization problem that commonly is solved using column generation. In this thesis, an approach based on deep learning predictions to alter the selection process in the pricing problem of column generation is presented. The concept of the machine learning model is to predict how likely it is that a task is assigned to a specific crew member. These predictions are intended to replace how the reduced costs are used in the different stages of column generation. In early iterations, the predictions are used in a greedy and direct fashion where the most probable tasks are attempted to be assigned to the given crew member. In later iterations, where the pricing problem is commonly modelled as a resource-constrained shortest path problem the predictions are used in the selection process of which tasks to include in the network.

The data used in the thesis is from an anonymous real world airline, consisting of two consecutive planning periods. Airline planning is usually done month by month and our planning periods and from December and January which is considered extra difficult as travel patterns are disrupted when there are big holidays, as in December.

The two planning periods are split into training and testing data to replicate how the machine learning model would be used. Thus it is trained on historical data, in a supervised fashion, and outputs predictions for what could be future planning problems. The features for the model is extracted from costs and constraints used in the optimization problem formulation. The cost is an aggregation of real-world cost and crew preferences. Crew competence and trip properties are explained by the constraint features. There is also a trip duration feature that aims to assess the degree of commitment related to assigning the trip. The ML model is a deep learning network and a wide range of hyperparameters are experimented with in the training stage. To measure the performance of predictions a metric explaining the ratio of how many accurately sorted tasks the predictions yielded. Unfortunately, none of the models was able to significantly outperform randomized order. Two possible explanations for this are concluded. Either the features used are not able to capture the behaviour of the problem or the models experimented with are not capable of learning the complex relationship between the features and

the labels. It may also be a combination of the reasons thus we suggest that further research on the topic should both investigate what further information can be extracted from the problem formulation and also experiment with other ML techniques.

To investigate the potential of accurate predictions experiments were conducted with predictions computed directly from a given solution. These experiments show a proof of concept as utilizing predictions only in a relatively few early column generation iterations increase computational speed significantly.

To conclude, the deep learning models experimented with in this thesis were not able to consistently find patterns in the given real-world data but results show that the concept of utilizing accurate predictions to increase computational speed in state of the art crew rostering algorithms is possible. For further research on the topic we suggest two directions: investigation of what further information can be extracted from the problem formulation to create more features and experimentation with other ML techniques.

Bibliography

[1] Alejandro Alvarez et al. “A Machine Learning-Based Approximation of Strong Branching”. In: *INFORMS Journal on Computing* 29 (Jan. 2017), pp. 185–195. DOI: 10.1287/ijoc.2016.0723.

[2] Jonathan F. Bard and Hadi W. Purnomo. “Preference scheduling for nurses using column generation”. In: *Eur J Oper Res* 164.2 (July 2005), pp. 510–534. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2003.06.046.

[3] Cynthia Barnhart et al. “Branch-and-Price: Column Generation for Solving Huge Integer Programs”. In: *Oper Res* (June 1998). URL: https://pubsonline.informs.org/doi/10.1287/opre.46.3.316.

[4] Yoshua Bengio et al. “Machine learning for combinatorial optimization: A methodological tour d’horizon”. In: *European Journal of Operational Research* 290.2 (2021), pp. 405–421. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2020.07.063. URL: https://www.sciencedirect.com/science/article/pii/S0377221720306895.

[5] Léon Bottou. “Large-Scale Machine Learning with Stochastic Gradient Descent”. In: *Proceedings of COMPSTAT’2010*. Ed. by Yves Lechevallier and Gilbert Saporta. Heidelberg: Physica-Verlag HD, 2010, pp. 177–186. ISBN: 978-3-7908-2604-3.

[6] Jacques Desrosiers and Marco E. Lübbecke. “A Primer in Column Generation”. In: *Column Generation*. Ed. by Guy Desaulniers et al. Boston, MA: Springer US, 2005, pp. 1–32. ISBN: 978-0-387-25486-9. DOI: 10.1007/0-387-25486-2_1. URL: https://doi.org/10.1007/0-387-25486-2_1.

[7] Balaji Gopalakrishnan and Ellis. L. Johnson. “Airline Crew Scheduling: State-of-the-Art”. In: *Ann Oper Res* 140.1 (Nov. 2005), pp. 305–337. ISSN: 1572-9338. DOI: 10.1007/s10479-005-3975-3.

[8] Silke Jütte et al. “Optimizing railway crew schedules with fairness preferences”. In: *J Sched* 20.1 (Feb. 2017), pp. 43–55. ISSN: 1099-1425. DOI: 10.1007/s10951-016-0499-4.

[9] Silke Jütte et al. “Optimizing Railway Crew Scheduling at DB Schenker”. In: *Interfaces* (Apr. 2011). URL: https://pubsonline.informs.org/doi/pdf/10.1287/inte.1100.0549.

[10] Maryam Karimi-Mamaghan et al. “Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art”. In: *European Journal of Operational Research* 296.2 (2022), pp. 393–422. ISSN: 0377-2217. DOI:

https://doi.org/10.1016/j.ejor.2021.04.032. URL: https://www.sciencedirect.com/science/article/pii/S0377221721003623.

[11] Howard Karloff. *Linear Programming*. Boston, MA, USA: Birkhäuser, 1991. DOI: 10.1007/978-0-8176-4844-2.

[12] Atoosa Kasirzadeh et al. "Airline crew scheduling: models, algorithms, and data sets". In: *EURO Journal on Transportation and Logistics* 6.2 (June 2017), pp. 111–137. ISSN: 2192-4376. DOI: 10.1007/s13676-015-0080-x.

[13] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *arXiv* (Dec. 2014). eprint: 1412.6980. URL: https://arxiv.org/abs/1412.6980v9.

[14] Niklas Kohl and Stefan E. Karisch. "Airline Crew Rostering: Problem Types, Modeling, and Optimization". In: *Ann Oper Res* 127.1 (Mar. 2004), pp. 223–257. ISSN: 1572-9338. DOI: 10.1023/B:ANOR.0000019091.54417.ca.

[15] Markus Kruber et al. "Learning When to Use a Decomposition". In: *Integration of AI and OR Techniques in Constraint Programming*. Ed. by Domenico Salvagnin and Michele Lombardi. Cham: Springer International Publishing, 2017, pp. 202–210. ISBN: 978-3-319-59776-8.

[16] Mouad Morabit et al. "Machine-Learning–Based Column Selection for Column Generation". In: *Transportation Science* (June 2021). URL: https://pubsonline.informs.org/doi/10.1287/trsc.2021.1045.

[17] Frédéric Quesnel et al. "A branch-and-price heuristic for the crew pairing problem with language constraints". In: *European Journal of Operational Research* 283.3 (2020), pp. 1040–1054. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2019.11.043. URL: https://www.sciencedirect.com/science/article/pii/S0377221719309555.

[18] Frédéric Quesnel et al. "Deep-learning-based partial pricing in a branch-and-price algorithm for personalized crew rostering". In: *Computers & Operations Research* 138 (Feb. 2022), p. 105554. ISSN: 0305-0548. DOI: 10.1016/j.cor.2021.105554.

[19] Adil Tahir et al. "An Improved Integral Column Generation Algorithm Using Machine Learning for Aircrew Pairing". In: *Transportation Science* (Sept. 2021). URL: https://pubsonline-informs-org.eu1.proxy.openathens.net/doi/10.1287/trsc.2021.1084.

[20] Roman Václavík et al. "Accelerating the Branch-and-Price Algorithm Using Machine Learning". In: *European Journal of Operational Research* 271 (May 2018). DOI: 10.1016/j.ejor.2018.05.046.