university of groningen / faculty of science and engineering / mathematics and applied mathematics

An integer programming model for timetabling at the University of Groningen

Bachelor's Project Mathematics

June 2024

Student: Roberto Schininà

First supervisor: prof. dr. Juan Peypouquet

Second assessor: prof. dr. ir. Bart Besselink

Abstract

University Course Timetabling is a big challenge that many universities must go through before the start of each academic year. This is a very complex administrative task that sometimes takes months to complete. Due to the many constraints and limited resources, it is not always possible to satisfy all requests. In this thesis, we propose a model for the curriculum-based university course timetabling problem at the University of Groningen (RUG) that guarantees parent-child relationships between lectures and other events such as tutorials and computer laboratories. While doing so, we also optimize the number of rooms used and wasted seating spaces in the classrooms. This model is then applied to all the courses given by the Department of Mathematics of the RUG in the academic year 2023/2024.

Contents

1. Introduction

Before the start of each academic year, educational institutions must perform the difficult task of allocating courses' events to rooms and timeslots while avoiding scheduling conflicts: this is known as the educational timetabling problem (ETP) (Ceschia, Di Gaspero & Schaerf, 2023). It is often solved manually and, depending on the institution's size, it may take a long time to complete (Soyemi, John Lekan & Oloruntoba, 2017). For example at the University of Groningen, it takes months to schedule the activities of the Faculty of Science and Engineering (Groningen, 2023). Moreover, this method is prone to mistakes and does not guarantee optimality.

ETPs have been studied extensively since the first half of the 20th century (Schmidt & Ströhlein, 1980) and they come in three different varieties (Ceschia, Di Gaspero & Schaerf, 2023) [1]:

- High-School Timetabling (HTT) which consists of scheduling, without any overlap, all lectures while avoiding the double allocation of teachers.
- University Examination Timetabling (ETT). This consists of scheduling all the exams without overlapping the ones with common students.
- University Course Timetabling (CTT) whose objective is to schedule lectures and other activities (for example tutorials and computer laboratories) without any overlap of events that have common students

While this subject has been researched for many years, its output was not oriented towards real-life application (RLA): researchers created models that could not be applied due to oversimplification and picked data sets carefully, ensuring that they worked with the models in question (Mccollum, 2006). This created a gap between research and RLA. Thankfully, the International Timetabling Competitions (ITCs) were able to push the research towards solving complex course timetabling problems without any shortcuts (Müller, Rudová & Müllerová, 2024). An example of the impact that the ITCs had, can

[1] Different institutions have different requirements so the following definitions may vary.

be found at Purdue University where the timetabling for all departments is automated (Rudová, Müller & Murray, 2011).

It is important to note that timetabling problems are proven to be NP-hard (Mahlous & Mahlous, 2023). Because of this, heuristic methods are often used to deal with these problems. However, there are other methods as well, such as operational research methods and hybrid approaches (Chen et al., 2021).

In this thesis, we propose a model to solve the CTT at the University of Groningen (RUG) using integer linear programming (which is a special type of operational research method). Particularly we focus on creating a timetable for the Mathematics Department. One of the main problems schedulers at the RUG have is not assigning lectures, but scheduling tutorials and laboratories. This is for two main reasons:

- lectures, tutorials, and laboratories must respect a parent-child relationship. This means that tutorials and labs must be scheduled after their corresponding lectures in order to adhere to the pattern "Lecture, Tutorial/Lab, Lecture, Tutorial/Lab".
- Course attendees are split into groups when attending tutorials and laboratories. This puts a strain on the limited number of resources (e.g rooms) that are available.

Therefore, the aim of this thesis is to create a schedule that satisfies this specific parent-child relationship. The inspiration for our model came from the work of Havås et al., 2013. However, the model they proposed does not have a parent-child relationship as strict as we have in Groningen. This necessitates the creation of a different model that guarantees a higher control over our timetable.

The main body of the thesis is divided into 3 Chapters. In Chapter 2 we provide the basic definitions and results from optimization, linear algebra, and the geometry of polyhedrons. From this, in Chapter 3, we present the main results of Linear programming; in particular, we prove the Fundamental Theorem of Linear Programming and we discuss how to solve (integer) linear programs using the Simplex algorithm and Branch and Cut methods. In Chapter 4 we define our model and we test it using data collected from the University of Groningen.

2. Preliminaries

In this chapter, we present some definitions and properties that will be used throughout the rest of the thesis. We only provide proofs of results that aid the understanding and the intuition behind the fundamental theorem of linear programming. Other proofs are omitted as they do not enter the scope of this thesis.

2.1. Optimization

Since timetabling can be modeled as optimization problems, we recall some basic definitions of optimization theory that are taken from (Rader, 2010).

An optimization model is a problem that can be written in the following way:

$$\underset{x \in C}{\text{minimize/maximize}}\ f(x)$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is the objective function, x is a vector formed by n independent variables (that we call decision variables) and C is the feasible set (or set of constraints). The constraints are of two types:

- variable bounds: they specify the values for which the decision variables x_i (for $i \in \{1, ..., n\}$) have meaning.
- General constraints: which specify all other requirements of the problem.

In order to stay consistent with the literature, throughout this thesis, every optimization problem will be written in the following form:

$$\begin{aligned} &\underset{x}{\text{minimize/maximize}} && f(x) \\ &\text{subject to} && \text{(general constraints)}, \\ & && \text{(variable bounds)}. \end{aligned}$$

Ultimately we wish to find solutions to the optimization problem. The x that satisfies the constraints (i.e. $x \in C$) is known as a feasible solution. A feasible solution $\hat{x}$

that satisfies $f(\hat{x}) < f(x)$ for all $x \in C$ is called an optimal solution (this is for the minimization problem, in the other case the sign is flipped). Optimization problems can have multiple global solutions or none at all. The latter can happen for one of the following two reasons:

- The set of feasible solutions is empty. Then the problem is called infeasible.
- For any feasible solution $x \in C$ we can always find a $y \in C$ that improves the value objective function more than x. These problems are called unbounded.

2.2. Linear Algebra

We assume the reader has some basic knowledge of linear algebra and therefore we report only one result that will be necessary when describing the geometry of polyhedrons. First, we present the definition of active constraints which was adapted from Definition 2.8 of (Bertsimas & Tsitsiklis, 1997).

Definition 1. *Let A be a matrix in $\mathbb{R}^{m\times n}$ and b a vector in $\mathbb{R}^m$. If a vector $\hat{x} \in \mathbb{R}^n$ satisfies $a_i^T\hat{x} = b_i$ (where a_i^T is the i^{th} row of A) for some i, we say that the corresponding constraint is active at $\hat{x}$. Furthermore, we define the set of indices of constraints that are active at $\hat{x}$ as follows:*

$$I(\hat{x}) = \{i \in \{1, ..., m\} \mid a_i^T\hat{x} = b_i\}$$

If there are n constraints that are active at $\hat{x}$, then $\hat{x}$ satisfies a system of n linear equations. The uniqueness of the solution of this system is discussed in the next theorem (for the proof see Theorem 2.2 from (Bertsimas & Tsitsiklis, 1997)).

Theorem 1. *Let $\hat{x} \in \mathbb{R}^n$ and $I(\hat{x})$ as in the definition above. Then, the following are equivalent:*

- *There exist n vectors in the set $\{a_i | i \in I(\hat{x})\}$, which are linearly independent.*
- *The span of the vectors a_i, $i \in I(\hat{x})$, is equal to $\mathbb{R}^n$*
- *The system of equations $a_i^T x = b_i$, $i \in I(\hat{x})$, has a unique solution.*

2.3. The geometry of polyhedrons

This section is essential for the development of linear programming, as the feasibility region of a linear program is a polyhedron. Almost all these results are taken from (Bertsimas & Tsitsiklis, 1997). Whenever a result from Bertsimas is used, we will write its corresponding location (theorem, proposition, etc.) in brackets. We begin with the standard definition of hyperplanes and halfspaces.

Definition 2. *(From Definition 2.3) Let $a \in \mathbb{R}^n$ be a nonzero vector and let b be a scalar.*

- *The set $\{x \in \mathbb{R}^n \mid a^T x = b\}$ is called a hyperplane.*
- *The set $\{x \in \mathbb{R}^n \mid a^T x \geq b\}$ is called a halfspace.*

By combining multiple halfspaces we obtain polyhedrons. Formally:

Definition 3. *(Adapted from Definition 2.1) A polyhedron $P \subseteq \mathbb{R}^n$ is a set that can be described as*

$$P = \{x \in \mathbb{R}^n | Ax \geq b\}$$

where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$.

Therefore a polyhedron is obtained from the the intersection of finitely many halfspaces. This allows us to easily prove that polyhedrons are convex.

Proposition 1. *Polyhedrons are convex.*

Proof. First, we show that any halfspace is convex. To this end let $H = \{x \mid a^T x \geq b\}$ be an arbitrary halfspace for some $a \in \mathbb{R}^n$ and $b \in \mathbb{R}$. Consider two arbitrary points $x, y \in H$. Then, for all $\lambda \in [0, 1]$:

$$a^T(\lambda x + (1 - \lambda)y) = \lambda a^T x + (1 - \lambda)a^T y \geq \lambda b + (1 - \lambda)b = b$$

Thus $(\lambda x + (1 - \lambda)y) \in H$ for any $x, y \in H$ and $\lambda \in [0, 1]$, which implies that halfspaces are convex.
Since any polyhedron can be described as an intersection of finitely many halfspaces we can conclude that polyhedrons are convex [1]. □

[1] Recall from convex analysis that the intersection of any finite number of convex sets is a convex set.

As we will see in Chapter 3 the optimal solution(s) of a linear programming problem occur on the corner(s) of a polyhedron. Three equivalent definitions characterize these corners. We begin with a geometric definition. In this case, we call the corner an extreme point.

Definition 4. *(From Definition 2.6) Let P be a polyhedron. A vector $x \in P$ is an extreme point of P if we cannot find two vectors $y, z \in P$, both different from x, and a scalar $\lambda \in [0, 1]$, such that:*

$$x = \lambda y + (1 - \lambda)z$$

We could also define a corner from an optimization perspective. In this case, we call the corner a vertex.

Definition 5. *(From Definition 2.7) Let $P \subseteq \mathbb{R}^n$ be a polyhedron. A vector $x \in P$ is vertex of P if it exists $c \in \mathbb{R}^n$ such that*

$$c^T x < c^T y$$

for all $y \in P$ with $y \neq x$.

Unfortunately, while the two definitions above are very important and will be used throughout the thesis, they are not ideal when describing algorithmic procedures. Hence, we provide an algebraic definition of corners.

Definition 6. *(From Definition 2.9) Consider a polyhedron $P \subseteq \mathbb{R}^n$ defined by linear equality and inequality constraints and let $\hat{x} \in \mathbb{R}^n$. Then:*

1. *The vector $\hat{x}$ is a basic solution if all equality constraints are active at $\hat{x}$ and n of them are linearly independent.*
2. *If $\hat{x}$ is a basic solution that satisfies all constraints (equalities and inequalities), we say it is a basic feasible solution.*

In geometry, vertexes and extreme points are often interchangeable so it may not be surprising that they represent the same object. Nonetheless, since we defined corners also algebraically, we want to prove the equivalence of all three definitions.

Theorem 2. *(From Theorem 2.3) Let $P = \{x \in \mathbb{R}^n | Ax \geq b\}$ be a nonempty polyhedron and $\hat{x} \in P$ with $I = \{i \in \{1, ..., m\} \mid a_i^T \hat{x} = b_i\}$. Then, the following are equivalent:*

1. *$\hat{x}$ is a vertex.*
2. *$\hat{x}$ is an extreme point.*
3. *$\hat{x}$ is a basic feasible solution.*

Proof.

(1) $\Rightarrow$ (2) We assume that $\hat{x}$ is a vertex. Then by Definition 5 there exists $c \in \mathbb{R}^n$ such that $c^T \hat{x} < c^T y$ for all $y \in P$ with $y \neq \hat{x}$.
Suppose $\hat{x}$ was not an extreme point. From Definition 4, there exists a $\lambda \in [0, 1]$ such that $\hat{x} = \lambda y + (1 - \lambda)z$ for some $y, z \in P$ both different from $\hat{x}$. If we multiply $\hat{x}$ by c^T we obtain the following:

$$c^T \hat{x} = c^T(\lambda y + (1 - \lambda)z) = \lambda c^T y + (1 - \lambda)c^T z$$

using the vertex assumption:

$$c^T \hat{x} = \lambda c^T y + (1 - \lambda)c^T z > \lambda c^T \hat{x} + (1 - \lambda)c^T \hat{x} = c^T \hat{x}$$

This is a contradiction, which implies that $\hat{x}$ is also an extreme point.

(2) $\Rightarrow$ (3) Assume $\hat{x}$ is an extreme point. Then by Definition 4 we cannot find $y, z \in P$ (with $y, z \neq \hat{x}$) such that $\hat{x} = \lambda y + (1 - \lambda)z$ for any $\lambda \in [0, 1]$. Suppose that $\hat{x}$ is not a basic feasible solution. Then the number of linearly independent active constraints a_i^T ($i \in I$) is less than n. Because of this there must exist a vector $d \in \mathbb{R}^n$ such that

$$a_i^T d = 0 \quad \text{for all} \quad i \in I. \tag{2.1}$$

Consider now two vectors y, z such that, for some $\epsilon > 0$ we have:

$$y = \hat{x} - \epsilon d$$

$$z = \hat{x} + \epsilon d$$

By choosing [2] ϵ such that $\epsilon |a_i^T d| < a_i^T \hat{x} - b_i$ we get that for $i \notin I$

$$a_i^T y = a_i^T \hat{x} - \epsilon a_i^T d > a_i^T \hat{x} - a_i^T \hat{x} + b_i = b_i$$

[2] We can do this as $a_i^T \hat{x} > b_i$ for all $i \notin I$

$$a_i^T z = a_i^T \hat{x} + \epsilon a_i^T d > a_i^T \hat{x} - a_i^T \hat{x} + b_i = b_i$$

On the order side, for any $i \in I$, it follows from (2.1):

$$a_i^T y = a_i^T \hat{x} - \epsilon a_i^T d = b_i - 0 = b_i$$

$$a_i^T z = a_i^T \hat{x} + \epsilon a_i^T d = b_i - 0 = b_i$$

Therefore, since y, z satisfy all active and non-active constraints, it follows that $y, z \in P$. In particular we have $\hat{x} = \frac{y+z}{2}$ with $y, z \neq \hat{x}$. This contradicts the assumption that $\hat{x}$ is an extreme point of P.
We conclude that $\hat{x}$ is also a basic feasible solution.

(3) $\Rightarrow$ (1) Assume $\hat{x}$ is a basic feasible solution. Define $c = \sum_{i \in I} a_i$. Then if we multiply c^T and $\hat{x}$ we get:

$$c^T \hat{x} = (\sum_{i \in I} a_i^T)\hat{x} = \sum_{i \in I} (a_i^T \hat{x}) = \sum_{i \in I} b_i$$

Moreover, for any $y \in P$ and $i \in I$, we have $a_i^T y \geq b_i$. Thus:

$$c^T y = (\sum_{i \in I} a_i^T) y = \sum_{i \in I} (a_i^T y) \geq \sum_{i \in I} b_i \tag{2.2}$$

This implies that $\hat{x}$ is a (not necessarily unique) global minimizer of the function $c^T x$ over P. Moreover, since $a_i^T x \geq b_i \; \forall \, i \in I$, it follows that the equality in (2.2) holds if an only if $a_i^T x = b_i \; \forall \, i \in I$.

Since $\hat{x}$ is a basic feasible solution, there exist n vectors in the set $\{a_i^T \mid i \in I\}$, which are linearly independent. Therefore by Theorem 1 there is a unique solution (i.e. $\hat{x}$) to the system of equations $a_i^T x = b_i$, $i \in I$. Thus we have that

$$c^T y > \sum_{i \in I} b_i = c^T \hat{x}$$

for all $y \in P$ with $y \neq \hat{x}$. This implies that $\hat{x}$ is the unique global minimizer of $c^T x$ over P and by Definition 5 it is a vertex.

□

This equivalence relation allows us to prove the following property.

Corollary 1. *A polyhedron $P = \{x \in \mathbb{R}^n \mid Ax \geq b\}$, where $A \in \mathbb{R}^{m \times n}$, has a finite number of vertices.*

Proof. If $m < n$ then, by Definition 6, we cannot have any basic feasible solution and, by Theorem 2, not a single vertex. If $m \geq n$ then we can have at most $k = \binom{m}{n}$ sets of indices of active constraints. Since $k < \infty$ it follows that we have at most k feasible solutions and again, by Theorem 2, at most k vertexes. □

This result is significant. As stated above, in Chapter 3 we will show that optimal solutions are achieved on the vertexes; since they are finite we can limit our search of the optimal solution by only checking the vertexes.

Until now we discussed extreme points without giving criteria for their existence. It can be shown that the existence of extreme points is related to the concept of containing lines which we define as follows:

Definition 7. *(From Definition 2.12) A polyhedron $P \subseteq \mathbb{R}^n$ contains a line if it exists a vector $x \in P$ and $d \in \mathbb{R}^n \backslash \{0\}$ such that $(x + \lambda d) \in P$ for all $\lambda \in \mathbb{R}$.*

We report the theorem concerning the existence of extreme points but we do not prove it as it goes beyond the scope of this thesis.

Theorem 3. *(Adapted from Theorem 2.6) A nonempty polyhedron has a vertex if and only if it does not contain a line.*

We conclude the preliminaries by providing a definition that will be used in one of the proofs of Chapter 3.

Definition 8. *(Defined in the proof of Theorem 2.8) A polyhedron $P \subseteq \mathbb{R}^n$ contains a half-line if it exists a vector $x \in P$ and $d \in \mathbb{R}^n \backslash \{0\}$ such that $(x + \lambda d) \in P$ for all $\lambda \in \mathbb{R}_{>0}$.*

Remark. *A polyhedron either contains one point or infinitely many. The reason is that if it contains two points then, as polyhedrons are convex, it must also contain the segment that connects them. In particular, this segment contains infinitely many points.*

3. Linear Programming

Linear programming is not as new as people think. The first example is attributed to Fourier, who in 1823 wrote a paper discussing how to determine if a polyhedron is nonempty. However, until the 1930s, there was no further research due to lack of application (Chakraborty, Chandru & Rao, 2020). Then, in 1939, Leonid Vitaliyevich Kantorovich published *Mathematical Methods in the Organization and Planning of Production* which is the first formal resource in Linear programming (Kutateladze, 2012). Unfortunately, due to the Cold War, his work was unknown in the West. Meanwhile, in the United States, Wassily Leontief formulated the Inter-industry Model of the American Economy. This model was later generalized by George Dantzig (known as the father of Linear programming), who at the time was working on logistical problems for the US Air Force. This generalization is now known as the Linear Programming [1] Model. (Albers & Reid, 1986).

3.1. Definitions and main results

Definition 9. *A linear program (LP) is an optimization problem in which the objective and the constraints are linear functions. It can be written in the following form:*

$$\begin{aligned} &\textit{minimize} \quad c^T x \\ &\textit{subject to} \quad Ax \geq b \end{aligned} \tag{3.1}$$

Before presenting the proof of the Fundamental Theorem we prove a lemma that is cardinal for the proof of the theorem. The statements of the lemma and theorem are taken from (Roma, 2019, Lemma 5.2.1, Theorem 5.2.1) which is published in Italian. There are equivalent statements in (Bertsimas & Tsitsiklis, 1997) (which we specify in parenthesis as we did in Chapter 2) but we report the "Italian" results. The reason for this choice is to enhance clarity.

[1] Interestingly the term programming has nothing to do with coding or computer science. It is a military term that refers to planning schedules efficiently or deploying men optimally (Dias Rasteiro, 2020).

Lemma 1 (Adapted from Theorem 2.8)**.** *Consider the following LP*

$$\begin{aligned} &\textit{minimize} && c^T x \\ &\textit{subject to} && Ax \geq b \end{aligned}$$

and assume that $P = \{\mathbf{x} \in \mathbb{R}^n | Ax \geq b\}$ *is nonempty, does not contain a line, and that it is bounded below. Then if* $y \in P$ *is not a vertex of* P *we can find a vector* $z \in P$ *such that* $c^T z \leq c^T y$ *and the number of linearly independent active constraints of* z *is larger than the one of* y*.*

Proof. Suppose that $y \in P$ is not a vertex. Then, by Theorem 2, the number of linearly independent active constraints at y is less than n, say k. Because of this it must exist a non-zero vector $d \in \mathbb{R}^n$ such that:

$$a_i^T d = 0$$

for all $i \in I = \{i \,|\, a_i^T y = b_i\}$. Without loss of generality, we assume that $c^T d \leq 0$ (if not then we can pick $-d$ instead of d). We distinguish two cases:

- First case: $\mathbf{c^T d < 0}$. Consider the half line $z(\lambda) = y + \lambda d$ with $\lambda \in \mathbb{R}_{>0}$. Then for all $i \in I$ we have:

  $$a_i^T z(\lambda) = a_i^T y + \lambda a_i^T d = a_i^T y + 0 = a_i^T y = b_i$$

  If $(y + \lambda d) \in P$ for all $\lambda \in \mathbb{R}_{>0}$ (that is, the whole half-line is contained in P) we would have:

  $$\lim_{\lambda \to \infty} c^T (y + \lambda d) = -\infty$$

  This would violate the assumption that the problem is bounded from below. Therefore the half-line must, eventually, exit P. Thus there exists a "last" $\hat{\lambda} > 0$ such that, for some $j \notin I$:

  $$a_j^T (y + \hat{\lambda} d) = a_j^T y + \hat{\lambda} a_j^T d = b_j$$

  Geometrically this means that the half line intersects the hyperplane $a_j^T x = b_j$. Using these information, and the fact that $c^T d < 0$, we have:

  $$c^T z(\hat{\lambda}) = c^T y + \hat{\lambda} c^T d < c^T y$$

  Note that, for $j \notin I$, we must have $a_j^T d \neq 0$ (as $a_j^T y \neq b_j$ and $a_j^T (y + \lambda d) = b_j$). Since d is orthogonal to all a_i^T $(i \in I)$, it must also be orthogonal to any of their linear combinations. We conclude that a_j is linearly independent from all the $a_i's$

($i \in I$). Therefore we have $c^T z(\hat{\lambda}) < c^T y$ and $z(\hat{\lambda})$ has at least $k+1$ linearly independent active constraints.

- $\mathbf{c^T d = 0}$: Consider the line $z(\lambda) = y + \lambda d$. Since P does not contain any line it means that there exists $\hat{\lambda} \in \mathbb{R}$ and $j \notin I$ such that $a_j^T(y + \hat{\lambda} d) = b_j$. In this case we have:

$$c^T z(\hat{\lambda}) = c^T y + \hat{\lambda} c^T d = c^T y$$

Using the same reasoning as before, a_j is independent from all other $a_i's$ for $i \in I$, and therefore there are at least $k+1$ linearly independent active constraints at $z(\hat{\lambda})$.

Since we found a vector z (namely $z(\hat{\lambda})$) with a number of linearly independent active constraints larger than y and $c^T z \leq c^T y$ the proof is concluded. □

Finally, we are ready to prove the theorem that Dantzig first proved in 1951 (Dantzig, 1951).

Theorem 4 (Fundamental Theorem of Linear Programming from Theorem 2.8 and Corollary 2.3)**.** *Consider the following LP*

$$\begin{aligned} &\textit{minimize} && c^T x \\ &\textit{subject to} && Ax \geq b \end{aligned}$$

and assume that $P = \{x \in \mathbb{R}^n | Ax \geq b\}$ *does not contain a line. Then only one of the following holds:*

- *The problem is infeasible.*
- *The problem is unbounded from below.*
- *The problem admits optimal solutions and at least one of them is a vertex of* P*.*

Proof. Note that the 3 statements are mutually exclusive. Therefore we can prove the theorem by showing that at least one of them is true. We assume that $P \neq \emptyset$ and that the problem is bounded from below. Then we consider two cases:

- Assume that $P = \{x\}$. Since P does not contain a line it follows that P must contain a vertex. As x is the only point, it must be the vertex and also the optimal solution.

- Assume P contains infinitely many points [2]. Since the problem is feasible and bounded from below we know there is at least one feasible solution. Call this solution x. We distinguish two cases:

  1. Assume x is a vertex. By Corollary 1 the number of vertices is finite (we denote the set of vertices by V) and therefore we can find that vertex $\hat{v}$ such that:
  $$c^T\hat{v} \leq c^T v$$
  for all $v \in V$.

  2. If $x \in P$ is not a vertex then we can use Lemma 1 a (finite) number of times until we find a vector $v \in P$ with n linearly independent active constraints and $c^T v \leq c^T x$ (we cannot use Lemma 1 more than n times because in $\mathbb{R}^n$ we have at most n linearly independent vectors). But then this vector v is a basic feasible solution and therefore a vertex. Then again, using the fact that the set of vertices is finite, we can find a vertex $\hat{v}$ such that:
  $$c^T\hat{v} \leq c^T v \leq c^T x$$
  for all $v \in V$.

  This shows that the vertex $\hat{v}$ is an optimal solution to the problem. Therefore statement 3 is true and the proof is complete.

□

[2] From the Remark at the bottom of the preliminaries we know that a polyhedron either contains one point or infinitely many.

3.2. LP in standard form

In this section, we do not consider general polyhedrons but polyhedrons in standard form (which are defined by hyperplanes only). We show that it is always possible to write a linear program in general form to an equivalent one in standard form. The reason behind this choice is that the Simplex method, which we use to solve linear programs, is defined using the standard form.

Definition 10. *A linear program is said to be in standard form if:*

- *All constraints are equalities.*
- *All of the elements of vector b are non-negative.*
- *All decision variables in the model are non-negative.*

It can be written in the following form:

$$\begin{aligned} &\textit{minimize} && z = c^T x \\ &\textit{subject to} && Ax = b, \\ & && x \geq 0 \end{aligned} \tag{3.2}$$

Now we show how to transform a general LP into an equivalent one in standard form.

1. **The objective function**
   If needed, we transform the maximization problem into a minimization one. Recall that the maximum value of the function z is equivalent to the minimum of $-z$. Therefore:
   $$\max(z) = c^T x \quad \text{becomes} \quad \min(-z) = -c^T x$$

2. **Constraints**
   If $b_i \leq 0$ for some i, we multiply by (-1) the corresponding row of A (i.e. a_i^T). Then inequality constraints can be converted into equality constraints by adding slack and surplus variables:
   $$\sum_{j=0}^{n} a_{ij} xj \leq b_i \quad \text{becomes} \quad \sum_{j=0}^{n} a_{ij} xj + s_1 = b_i$$
   and
   $$\sum_{j=0}^{n} a_{ij} xj \geq b_i \quad \text{becomes} \quad \sum_{j=0}^{n} a_{ij} xj - s_2 = b_i$$

where $s_1, s_2 \geq 0$ are the slack and surplus variable respectively.

3. **Variables**
   Lastly, decision variables need to be non-negative. In order to satisfy this requirement we consider the following transformations;
   - If $x_i \leq 0$ then we just substitute x_i with $-x_i$.
   - If x_i is a free variable then we define two variables $x_i^+, x_i^- \geq 0$ such that $x_i = x_i^+ - x_i^-$.

We want to stress the fact that while the optimal value is the same for both problems (in standard and in general form), the optimal solution is different as we are considering two different polyhedrons.

Throughout the rest of Chapter 3, we will always assume that matrix A is full rank and that $m < n$ [3] (if $m \geq n$ then our system will have at most one solution. This is not interesting in an optimization problem). The full rank assumption is without loss of generality from an Operation Research perspective. When building an LP model, scientists do not include repetitive constraints which therefore guarantees that the the constraint matrix A is full rank.

Now that we have rewritten the problem, we can provide a different definition of basic (feasible) solutions in order to simplify our treatment of the Simplex algorithm. For the remainder of section 3.2 and all 3.3 we use as reference (Andreasson, Evgrafov & Patriksson, 2005). As we have done before, we place between parentheses the location of the corresponding results.

Definition 11. *(See page 215)*
Consider problem (3.2) with $rank(A) = m$ *(with* $A \in \mathbb{R}^{m \times n}$*) and* $m < n$*. A point* $\hat{x}$ *is a basic solution of (3.2) if:*

- $A\hat{x} = b$*; and*
- *the columns of* A *corresponding to the non-zero components of* $\hat{x}$ *are linearly independent.*

If, moreover, $\hat{x}$ *satisfies* $\hat{x} \geq 0$ *then we call it a basic feasible solution.*

[3] The reader may recall that in Corollary 1 we required $m \geq n$. Now we want $m < n$ because we are redefining the definition of a basic (feasible) solution. However, all theorems and results still hold with this new definition.

Since A is full rank we can solve the system $Ax = b$ by partitioning matrix A in two. The first sub-matrix, which we denote as N, is composed of the $n - m$ columns of A corresponding to components of x that are set to 0; we call them non-basic variables and we denote them by the sub-vector x_N. The second sub-matrix, denoted by B, is formed by m linearly independent columns of A (which then guarantees that B is non-singular). Then, the remaining components of x are known as basic variables and are denoted by x_B.

Proposition 2. *(See page 215) Let* $x = \begin{pmatrix} x_B \\ x_N \end{pmatrix} = \begin{pmatrix} x_B \\ 0 \end{pmatrix}$ *and* $A = \begin{pmatrix} B & N \end{pmatrix}$ *defined as above. If* $x_B = B^{-1}b$ *then* x *is a basic solution.*

Proof. By construction, the columns of A corresponding to the non-zero components of x are linearly independent: thus $rank(A) = m$. We only need to show that $Ax = b$. Indeed:

$$Ax = \begin{pmatrix} B & N \end{pmatrix} \begin{pmatrix} x_B \\ x_N \end{pmatrix} = Bx_B + Nx_N =$$

Since $x_N = 0$ by construction and $x_B = B^{-1}b$ we get:

$$= Bx_B = BB^{-1}b = b$$

Therefore x is a basic solution. □

Corollary 2. *(See page 215) Let* x *be as above. If* $x_B = B^{-1}b \geq 0$ *then* x *is a basic feasible solution*

In the next subsection, we show a way to characterize adjacent extreme points. But why are we interested in it? From the Fundamental Theorem, we know that if a problem has an optimal solution then we can find it on an extreme point; instead of going through all the vertexes we "jump" from vertex to an adjacent vertex (this is effectively the idea behind the Simplex Method).

3.2.1. Adjacent extreme points

Definition 12. *(Algebraic characterization of adjacency from Definition 8.12) Two extreme points* x, y *of a polyhedron* P *are adjacent if all points* z *on the line segment between* x *and* y *satisfy the following property: if* $z = \lambda u + (1 - \lambda)w$ *(for some* $\lambda \in (0, 1)$ *and* $u, w \in P$*) then* u, w *must be on the line segment between* x *and* y*.*

Consider the linear program in standard form and let x be a basic feasible solution corresponding to the partition $A = \begin{pmatrix} B_1 & N_1 \end{pmatrix}$. We can rewrite the two sub-matrices in the following way

$$B_1 = (b_1, .., b_m) \quad \text{and} \quad N_1 = (n_1, .., n_{n-m})$$

If we swap the columns b_1 and n_1 (this is done without loss of generality since we can always reorder the constraints) we get a new partition of A. In particular $A = \begin{pmatrix} B_2 & N_2 \end{pmatrix}$ where:

$$B_2 = (n_1, .., b_m) \quad \text{and} \quad N_2 = (b_1, .., n_{n-m})$$

Using this construction we can prove an important result, central to the simplex algorithm's treatment.

Proposition 3 (Algebraic characterization of adjacency from Proposition 8.13)**.** *Let $u, v \in P$ be two extreme points that correspond, respectively, to the partitions, $\begin{pmatrix} B_1 & N_1 \end{pmatrix}$ and $\begin{pmatrix} B_2 & N_2 \end{pmatrix}$ defined above. Then u, v are adjacent extreme points.*

Proof. First, assume that the variables of u and v are ordered in the same way (without loss of generality since it is a matter of reordering). Then we can write:

$$u = (u_1, ..., u_m, 0, ..., 0)^T$$

and

$$v = (0, v_2, ..., v_{m+1}, 0, ..., 0)^T$$

Let z be an arbitrary point in the segment between u and v. That is:

$$x = \lambda u + (1 - \lambda)v$$

for some $\lambda \in (0, 1)$. In order to prove the proposition we must show that if $x = \alpha y_1 + (1 - \alpha)y_2$ (for some $y_1, y_2 \in P$ and $\alpha \in (0, 1)$) then y_1 and y_2 must belong to the segment connecting u and v.

Since y_1 and y_2 are feasible solution they must satisfy the following system:

$$\begin{aligned} x_1 b_1 + x_2 b_2 + ... + x_{m+1} n_1 &= b \\ x_{m+2}, ..., x_n &= 0 \\ x &\geq 0 \end{aligned}$$

or

$$\begin{aligned} \begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix} B + x_{m+1} n_1 &= b \\ x_{m+2}, ..., x_n &= 0 \\ x &\geq 0 \end{aligned}$$

If $x_{m+1} = 0$ then the unique solution (recall that B is invertible) of the above system is $x = u$ while if $x_{m+1} = v_{m+1}$ the unique solution of the above system is $x = v$. This shows that y_1 and y_2 belong to the segment connecting u and v which allows us to conclude that u, v are adjacent. □

3.3. Solving linear programming problems: the Simplex Method

From the Fundamental Theorem of Linear Programming, we know that if a linear program has a solution then we can find it on a vertex of the polyhedron. Therefore we could iterate over all the possible vertices and find the global minimum.

This may seem like a good method. However, the number of vertices could be considerably large. For example, the n dimensional hypercube has 2^n many vertices; computationally this would be quite expensive as n increases. Therefore we want to pick a subset of the vertices in a "smart way". To do this we will use the notion of adjacency we presented in the previous section. Starting from one vertex we jump to an adjacent one only if the value of the objective function improves: this is known as the Simplex Method. The method is composed of two phases: Phase I allows us to determine our initial vertex while Phase II determines the "jumping" part. We begin with the latter.

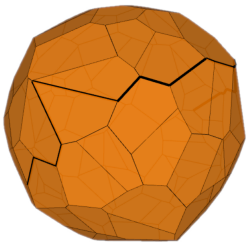

Figure 3.1.: A simplex path on a convex polyhedron (Huiberts, 2022)

3.3.1. Phase II of the Simplex Method

As reference we use section 9.1.1 of Andreasson, Evgrafov and Patriksson, 2005. Consider a linear program in standard form:

$$\begin{aligned} \text{minimize} \quad & z = c^T x \\ \text{subject to} \quad & Ax = b, \\ & x \geq 0 \end{aligned} \tag{3.3}$$

Step 1

Let $x = \begin{pmatrix} x_B \\ x_N \end{pmatrix}$ be an extreme point corresponding to the partition $A = (B, N)$. First, we rewrite the constraints using the partition:

$$Ax = (B, N) \begin{pmatrix} x_B \\ x_N \end{pmatrix} = Bx_B + Nx_N = b$$

or, equivalently,

$$x_B = B^{-1}b - B^{-1}Nx_N \tag{3.4}$$

We can also rewrite the objective function using the partition. Rearrange the components of c such that $c = \begin{pmatrix} c_B \\ c_N \end{pmatrix}$. Then if we multiply the partition of c and x we get:

$$c^T x = c_B^T x_B + c_N^T x_N =$$

substituting equation (3.4) we obtain:

$$= c_B^T(B^{-1}b - B^{-1}Nx_N) + c_N^T x_N$$

$$= c_B^T B^{-1} b + (c_N^T - c_B^T B^{-1} N) x_N \tag{3.5}$$

Then we define $\tilde{c}_N := (c_N^T - c_B^T B^{-1} N)$: this is known as the reduced cost vector of the non-basic variable. Using $\tilde{c}_N$ we can rewrite Equation (3.4) as:

$$c^T x = c_B^T B^{-1} b + \tilde{c}_N x_N \tag{3.6}$$

Step 2

In this step, we check if our current extreme point is an optimal solution. In order to do this we increase the non-basic vector $(x_N)_j$ from 0 to 1. Now consider the j^{th} component of $\tilde{c}_N$:

$$(\tilde{c}_N)_j = ((c_N^T - c_B^T B^{-1} N) x_N)_j = (c_N^T - c_B^T B^{-1} N)_j \cdot 1 = (c_N^T - c_B^T B^{-1} N)_j$$

If $(\tilde{c}_N)_j \geq 0$ for all $j = 1, .., n - m$, there does not exist any adjacent extreme point that improves the objective value. This implies that x is an optimal solution (see Theorem 5 at the end of this sub-chapter for the proof).

Step 3

If there is at least one negative reduced cost, then we need to keep searching: i.e. determine a new vertex. In particular Proposition 3 tells us that we determine this point by swapping one column of B and one column of N. In order to minimize the function the quickest we choose the nonbasic variable with the most negative reduced cost: the column of N corresponding (same index) to that nonbasic variable will be swapped (in this case we say that the variable $(x_N)_j$ enters the basis).

We are left to determine which column of B will be swapped. Assume that the variable $(x_N)_j$ has entered the basis and consider the extreme point of step 1. Then, as $(x_N)_j$ increases from 0 to 1 we will move along the following half line:

$$l(\alpha) = \begin{pmatrix} B^{-1} b \\ 0 \end{pmatrix} + \alpha \begin{pmatrix} -B^{-1} N_j \\ e_j \end{pmatrix} \quad \alpha \geq 0$$

To satisfy the nonnegativity constraints we must have $l(\alpha) \geq 0$ for all $\alpha \geq 0$. Moreover, the unboundedness constraints are satisfied only if $B^{-1} N_j \in \mathbb{R}^m$ has at least one positive element (see Theorem 6 for the proof).

The maximal value of α that maintains feasibility is given by the following minimization

problem:

$$\hat{\alpha} = \min_{i \in \{k | (B^{-1}N_j)_k > 0\}} \frac{(B^{-1}b)_i}{(B^{-1}N_j)_i}$$

Remark. *This pricing rule is known as the minimum ratio test. We consider this because it is the standard one, but other rules exist and can be implemented according to the problem's needs (Ploskas & Samaras, 2014).*

Then, the index of the column of B that will be swapped is given by:

$$i = \operatorname*{argmin}_{i \in \{k | (B^{-1}N_j)_k > 0\}} \frac{(B^{-1}b)_i}{(B^{-1}N_j)_i}$$

Step 4

Construct a new partition by swapping the j^{th} column of N with the i^{th} column of B.

Next, we present the optimality conditions of the Simplex Method. The proofs we use are taken from Theorem 6.4.1 and Corollary 6.4.1 of (Roma, 2019).

Theorem 5 (Optimality condition of the simplex method)**.** *[Adapted from Proposition 9.1] Consider a linear program in standard form and let* $\hat{x} = \begin{pmatrix} B^{-1}b \\ 0 \end{pmatrix}$ *be the extreme point that corresponds to the partition* $A = (B, N)$*. If the reduced cost* $(\tilde{c}_N)_j \geq 0$ *for all* $j = 1, ..., n - m$*, then* $\hat{x}$ *is an optimal solution.*

Proof. Let $x = \begin{pmatrix} x_B \\ x_N \end{pmatrix}$ be an arbitrary feasible point. Then, from equation (3.6) we have:

$$c^T x = c_B^T x_B + c_N^T x_N = c_B^T B^{-1} b + \tilde{c}_N x_N$$

Since $(\tilde{c}_N)_j \geq 0$ for all $j = 1, ..., n - m$ and $x_N \geq 0$, it follows that:

$$(c^T x)_j \geq (c_B^T B^{-1} b)_j = (c_B^T B^{-1} b + c_N^T 0)_j = (c^T \hat{x})_j$$

for all $j = 1, ..., n - m$. Therefore $\hat{x}$ is an optimal solution for the problem □

Corollary 3 (Adapted from Proposition 9.10)**.** *Consider a linear program in standard form and let* $\hat{x} = \begin{pmatrix} B^{-1}b \\ 0 \end{pmatrix}$ *be the extreme point that corresponds to the partition* $A = (B, N)$*. If the reduced cost* $(\tilde{c}_N)_j > 0$ *for all* $j = 1, ..., n-m$*, then* $\hat{x}$ *is the unique optimal solution.*

Proof. Using the same reasoning as in Theorem 5 we obtain that :

$$(c^T x)_j > (c_B^T B^{-1} b)_j = (c_B^T B^{-1} b + c_N^T 0)_j = (c^T \hat{x})_j$$

for all $j = 1, ..., n-m$.The strict inequality implies that $\hat{x}$ is the unique optimal solution □

Lastly, we consider the unboundedness criterion of the simplex method that is needed to justify the choices of step 3. In Andreasson, Evgrafov and Patriksson, 2005 only the statement is provided on page 228. The proof comes from Theorem 6.4.2 of (Roma, 2019).

Theorem 6. *Consider problem (3.3) and let (B, N) be a valid partition of A. If there exists some $j \in \{1, .., n-m\}$ such that:*

- $(\tilde{c}_N)_j < 0$.
- $B^{-1} N_j \leq 0$.

then the problem is unbounded from below

Proof. Consider the half line defined in step 3. That is:

$$l(\alpha) = \begin{pmatrix} B^{-1}b \\ 0 \end{pmatrix} + \alpha \begin{pmatrix} -B^{-1}N_j \\ e_j \end{pmatrix} \quad \alpha \geq 0$$

Using this we define:

$$l_N(\alpha) := \alpha e_j$$

$$l_B(\alpha) := B^{-1}b - \alpha B^{-1} N_j$$

Then we have:

$$Al(\alpha) = Bl_B(\alpha) + Nl_N(\alpha) = b$$

Clearly have $l_N(\alpha) \geq 0$ and, using the second assumption, we obtain:

$$l_B(\alpha) = B^{-1}b - \alpha B^{-1} N_j \geq 0$$

Therefore all the constraints of (3.3) are satisfied. Now consider the value of the objective function at $l(\alpha)$; from equation (3.6) we obtain:

$$z(l(\alpha)) = c_B^T B^{-1} b + \tilde{c}_N l_N(\alpha) = c_B^T B^{-1} b + \alpha(\tilde{c}_N)_j$$

Then as $\alpha \to \infty$ we have $z(l(\alpha)) \to -\infty$ which shows that the problem is unbounded from below. □

3.3.2. Phase I of the Simplex Method

But how can we solve Phase II if we do not have an initial extreme point? In two or three dimensions we may determine it by plotting the polyhedron. But what if we are solving a problem with a dimension larger than three? In that case, we apply Phase I. Again, consider the LP in standard form:

$$\begin{aligned} \text{minimize} \quad & z = c^T x \\ \text{subject to} \quad & Ax = b, \\ & x \geq 0 \end{aligned} \tag{3.7}$$

then its correspondent Phase I problem is:

$$\begin{aligned} \text{minimize} \quad & w = \sum_{i=1}^{m} a_i \\ \text{subject to} \quad & I^m a + Ax = b, \\ & x \geq 0 \\ & a \geq 0 \end{aligned} \tag{3.8}$$

where the elements of $a \in \mathbb{R}^m$ are known as artificial variables. Notice that this problem always admits an optimal solution as:

- $\begin{pmatrix} a \\ x \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}$ is a solution of (3.8). Therefore the feasible set (or the polyhedron) is not empty.
- The problem is bounded from below as each element of a is nonnegative.
- A vertex exists as the partition (I^m, A) corresponds to a basic feasible solution (with $N = A$ and $I^m = B$).

Therefore, by the Fundamental Theorem of Linear Programming, problem (3.8) admits an optimal solution.
Knowing that an optimal solution exists for the Phase I problem allows us to determine if the initial problem is solvable or not. This is discussed in the next theorem.

Theorem 7 (Adapted from Theorem 6.5.1)**.** *The point $\hat{x}$ is a feasible solution of the LP (3.7) if and only if the optimal solution $\begin{pmatrix}\hat{a}\\ \hat{x}\end{pmatrix}$ of (3.8) is equal to $\begin{pmatrix}0^m\\ \hat{x}\end{pmatrix}$.*

Proof.

($\Rightarrow$) Assume that $\hat{x}$ is a feasible solution of (3.7) and that $\begin{pmatrix}\hat{a}\\ \hat{x}\end{pmatrix}$ is the optimal solution of the Phase I problem for some $\hat{a} \in \mathbb{R}^m$ and $a \geq 0$. Suppose there is some $j \in \{1, .., m\}$ such that $(\hat{a})_j > 0$. Then, the optimal value is larger than 0. Since $\hat{x}$ is a solution of (3.7) we must have $A\hat{x} = b$. Then the vector $\begin{pmatrix}0^m\\ \hat{x}\end{pmatrix}$ is a solution for (3.8) and the value of the objective function at this point is 0. We have found a solution that is better than the optimal one. This is a contradiction and therefore the optimal solution of (3.8) is $\begin{pmatrix}0^m\\ \hat{x}\end{pmatrix}$.

($\Leftarrow$) Assume $\begin{pmatrix}0^m\\ \hat{x}\end{pmatrix}$ is an optimal solution for (3.8). Then:

$$I^m\hat{a} + A\hat{x} = 0 + A\hat{x} = b$$

which implies that $\hat{x}$ is a feasible solution for (3.7) □

Corollary 4 (Adapted from Corollary 6.5.1)**.** *Let $\hat{w}$ be the optimal value of (3.8). If $\hat{w} = 0$ then the problem (3.7) has a basic feasible solution. If $\hat{w} > 0$ then the original problem is infeasible.*

Therefore, if $\hat{w} = 0$, an extreme point can be found by solving the Phase I problem which can later be used as the starting vertex for the Phase II problem.

3.3.3. Convergence of the Simplex Method

We conclude the treatment of the Simplex method by discussing its convergence properties. To this end, we need to define degenerate solutions.

Definition 13. *(From Remark 8.4) If more than $n - m$ variables are zero at a basic solution x, then the corresponding partition is not unique. Such a basic solution is called degenerate.*

If there is a degenerate solution it is possible that the algorithm cycles between some vertices and thus never terminates. We do not consider this problem in the thesis as it

does not enter the main focus [4]. However, it is worth noting that there exist rules that avoid cycling; an example is Bland's rule (Bland, 1977).
However, if there are no degenerate solutions we can easily prove that the algorithm terminates in a finite number of steps.

Theorem 8. *(From Theorem 9.11) If all of the basic feasible solutions are non-degenerate, then the simplex algorithm terminates after a finite number of iterations.*

Proof. Recall that one of the conditions of problem (3.3) is $x \geq 0$. Therefore if the basic feasible solution x is non degenerate, it has exactly m positive components. Then if we consider the minimum ratio test we have that:

$$\hat{\alpha} = \min_{i \in \{k | (B^{-1}N_j)_k > 0\}} \frac{(B^{-1}b)_i}{(B^{-1}N_j)_i} > 0$$

Therefore, at each iteration, the objective value decreases and thus we never visit the same vertex twice (i.e. we never cycle).
Since the number of vertices is finite the Simplex Method terminates after a finite number of iterations. □

[4] Note that cycling from degeneracy does not seem to occur often in practical application (Andreasson, Evgrafov & Patriksson, 2005).

3.4. Solving (Mixed) Integer Programming problems

A mixed integer program (MIP) is a linear program in which we impose integrality constraints on at least one decision variable. If we require integrality for all the decision variables we then call it a pure integer program.
A naive method to solve MIPs would be finding the optimal solution (e.g. using the Simplex Method), and then rounding the decision variables to the nearest integer. However, this method may not give us the optimal solution, or, in the worst case, it might not even be feasible. Therefore, to solve this kind of problem, we use two different methods [5] (which are often combined): branch and bound, and cutting planes.

3.4.1. Branch and bound

We introduce this method by employing an example (for reference see Chapter 23 section 5 of Vanderbei, 2020). Consider the following IP problem.

$$\begin{aligned} \text{minimize} \quad & z = x_1 + 2x_2 \\ \text{subject to} \quad & -x_1 + x_2 \leq \frac{16}{3} \\ & 2x_1 + x_2 \leq \frac{23}{2} \\ & x_1 \leq \frac{11}{2} \\ & x_2 \leq \frac{15}{2} \\ & x \in \mathbb{Z}^n \end{aligned} \tag{3.9}$$

[5] The methods are valid for both MIPs and pure integer programs.

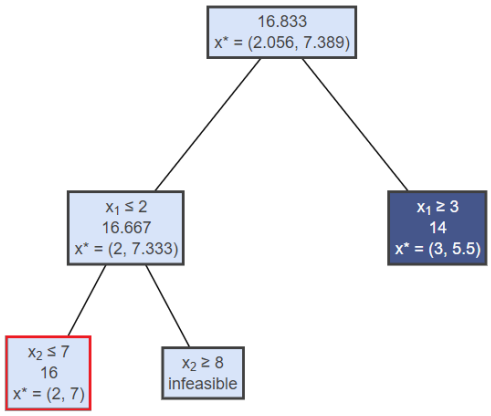

Figure 3.2.: Branch and Bound method for Example 3.9. The red square represents the optimal integer solution. The dark blue square represents a branch that did not improve the objective function. The image was generated using the gilp python library(Robbins et al., 2023)

.

First, we solve the problem without the integrality constraint (the removal is called LP relaxation). The optimal solution is $(2.056, 7.389)$ with an optimal value of 16.833. Since the solution is not integral (if it was then we would be done) we perform a branching operation. Choose a decision variable that does not satisfy integrality: for example x_1. Then solve two problems with the same objective function of (3.9) but with the following variations:

1. One adding the constraint $x_1 \leq \lfloor 2.056 \rfloor = 2$. In this case, the optimal solution is $(2, 7.333)$ and the optimal value 16.667.

2. The other adding the constraint $x_1 \geq \lceil 2.056 \rceil = 3$. In this case the optimal solution is $(3, 5.5)$ and the optimal value 14

Now we solve subproblem (1): we get two nodes: one that gives an infeasible solution and one that gives an integral solution. The latter gives us an optimal value that is larger than the right-hand branch (the blue square in the picture). Therefore we do not need to explore the right branch and the optimal solution of (3.9) is $(2, 7)$.
In our example could we have branched on x_2 first rather than x_1? The answer is yes, but this choice was not dictated by us but by Robbins et al., 2023 who wrote

the program. In particular, there are different strategies for branching, searching (the order in which subproblems in the tree are solved), and pruning (rules that prevent exploration of suboptimal regions of the tree) (Morrison et al., 2016). We do not explore these different strategies as they do not enter the focus of this thesis.

3.4.2. Cutting planes

The cutting planes method consists of considering additional constraints to the IP problem. These additional constraints are known as valid cuts. As a reference for this section we used (Lessard, 2017-18)

Definition 14. *Let P denote the feasible set for (3.8). We call an inequality a valid inequality (or valid cut) if and only if it does not eliminate any feasible integer solutions.*

The cutting planes method is similar to the Branch and Bound one.

1. Solve the LP relaxation of (3.9).
2. If the solution is integral then we are done.
3. If it is not integral then execute a valid cut.
4. Add the new cut constraint to (3.8) and go to step 1

It is very important to note that, since the LP relaxation's feasible set of (**??**) is convex, we can always find a valid cut. This is a consequence of the hyperplane separation theorem.

As in Branch and Bound, there are different strategies for creating valid cuts in order to reduce the computations needed to find the optimal solutions. Since this is not the focus of the thesis we refer the reader to Cornuéjols, 2008.

4. Timetabling at the University of Groningen

As stated in the introduction, this thesis aims at creating a model for university course timetabling. There are two main variants for this problem (Ceschia, Di Gaspero & Schaerf, 2023):

- Post-Enrolment course timetabling (PE-CTT): where students can enroll in any courses they wish to.
- Curriculum-Based course timetabling (CB-CTT): in which students must follow a predetermined set of courses.

At the University of Groningen, we focus on CB-CTT as students are enrolled in three different tracks: applied mathematics, general mathematics, and probability and statistics. Before delving into the mathematical formulation we want to provide an overview of the different elements that we need to take into consideration when creating the timetable.

4.1. Problem description

- Blocks. The academic year at the university is divided into four blocks (trimesters): 1A, 1B, 2A, and 2B. Each block has a different number of activities that must be scheduled.
- Rooms. In our model, we include 56 rooms: 44 of them are for lectures and tutorials while the remaining 12 are for computer laboratories.
- Lecturers. They are in charge of teaching courses. Sometimes they may teach multiple courses in a single block or multiple teachers may teach the same course.
- Students. There are two main types of mathematics students at the University of Groningen: bachelor's and master's. First-year bachelor's students follow a common path while second and third years can specialize in the three tracks described

above. Master students on the other side are split in general and applied. Within these two tracks, there are several specializations. However, as it is also possible to follow a specialization-free master we do not consider any specialization, only the two tracks.

- Courses. As our university uses a Curriculum Based Timetabling system we effectively ignore the single students because it is implied that they are following the courses assigned to them. Therefore all the constraints put in place to avoid conflict will be based on the tracks and not on the students.
- Time. Our model creates a weekly schedule that will be therefore repeated for the rest of the trimester. Each week has five days (from Monday to Friday) and on each day there are 5 timeslots of 2 hours each: 9 to 11, 11 to 13, 13 to 15, 15 to 17, and 17 to 19.

Now we can list all the constraints. These are the same constraints that schedulers at the University of Groningen must use. We take this opportunity to thank them for the information provided.

Hard Constraints

- All events for each course are scheduled.
- Each room can hold at most one event in a day and timeslot.
- Courses of the same specialization/track must not collide.
- Lecturers must teach the courses they are assigned to.
- Lecturers may teach at most one course at a given timeslot and day.
- Computer rooms cannot be used for tutorials and lectures. Similarly, lecture and tutorial rooms cannot be used for computer labs.
- Each room must be large enough for all students participating in a specific event.
- Lectures cannot be scheduled on Friday and two lectures should be at least one day apart.
- If a course has several groups attending tutorials and/or labs then they must all follow it at the same time.

- Parent-child relationship (the pattern "Lecture, Tutorial/Lab, Lecture, Tutorial/Lab") must be respected.
- Laboratories must be scheduled in the timeslot after the tutorial.

Objectives

Our model considers three sub-objectives that we wish to minimize. The first two are the same that the timetablers try to minimize while the third one is proposed by us.

- The number of events happening in the timeslot 17 to 19.
- The number of unused seats.
- The number of rooms used.

4.2. Proposed integer programming model

Now we list all sets, parameters, and variables that will be used in our implementation.

4.2.1. Definition of sets and parameters

- $\mathcal{C}_B^1$: Courses offered to all first-year bachelor students.
- $\mathcal{C}_{BA}^2$: Courses offered to second-year bachelor students specializing in Applied Mathematics.
- $\mathcal{C}_{BG}^2$: Courses offered to second-year bachelor students specializing in General Mathematics.
- $\mathcal{C}_{BP}^2$: Courses offered to second-year bachelor students specializing in Probability and Statistics.
- $\mathcal{C}_{BA}^3$: Courses offered to third-year bachelor students specializing in Applied Mathematics.
- $\mathcal{C}_{BG}^3$: Courses offered to third-year bachelor students specializing in General Mathematics.
- $\mathcal{C}_{BP}^3$: Courses offered to third-year bachelor students specializing in Probability and Statistics.

- $\mathcal{C}_{MG}$: Courses offered to master students specializing in General Mathematics.
- $\mathcal{C}_{MA}$: Courses offered to master students specializing in Applied Mathematics.
- $\mathcal{C}$: set of all offered courses.
- $\mathcal{D} = \{1, 2, 3, 4, 5\}$: Set of days from Monday to Friday.
- $\mathcal{T} = \{1, 2, 3, 4, 5\}$: Set of time slots in a day.
- $\mathcal{R}_{LT}$: Set of rooms used for lectures and tutorials.
- $\mathcal{R}_{LAB}$: Set of rooms used for computer labs.
- $\mathcal{R}$: set of all rooms available.
- $\mathcal{L}$: sets of all lecturers.
- $\mathcal{S} := \mathcal{C} \times \mathcal{D} \times \mathcal{T} \times \mathcal{L}$ (where $\times$ represents the Cartesian product): set that represents all possible events.
- n_r: number of available seating places in room $r \in \mathcal{R}$.
- n_c: number of students enrolled in course $c \in \mathcal{C}$.
- $n1_{lec,c} = \begin{cases} 1, & \text{if the first lecture for course } c \in \mathcal{C} \text{ takes place.} \\ 0, & \text{otherwise} \end{cases}$
- $n2_{lec,c} = \begin{cases} 1, & \text{if the second lecture for course } c \in \mathcal{C} \text{ takes place.} \\ 0, & \text{otherwise} \end{cases}$
- $n1_{tut,c} = \begin{cases} 1, & \text{if the first tutorial for course } c \in \mathcal{C} \text{ takes place.} \\ 0, & \text{otherwise} \end{cases}$
- $n2_{tut,c} = \begin{cases} 1, & \text{if the second tutorial for course } c \in \mathcal{C} \text{ takes place.} \\ 0, & \text{otherwise} \end{cases}$
- $n1_{lab,c} = \begin{cases} 1, & \text{if the first lab for course } c \in \mathcal{C} \text{ takes place.} \\ 0, & \text{otherwise} \end{cases}$
- $n2_{lab,c} = \begin{cases} 1, & \text{if the second lab for course } c \in \mathcal{C} \text{ takes place.} \\ 0, & \text{otherwise} \end{cases}$

- $n_{g,c}$: number of tutorial groups for course $c \in \mathcal{C}$.
- k_{present}: coefficient that represents the proportion of students that are present at an event. It is equal to 0.9 for blocks 1A and 1B while it is 0.8 for blocks 2A and 2B.
- M: matrix in $\mathbb{R}^{|\mathcal{L}|\times|\mathcal{C}|}$ with the following assignment rule:

$$M_{lc} = \begin{cases} 1, & \text{if professor } l \in \mathcal{L} \text{ is teaching course } c \in \mathcal{C}. \\ 0, & \text{otherwise.} \end{cases}$$

4.2.2. Definition of decision variables

As stated in the introduction, the inspiration for our model comes from Havås et al., 2013. Their model uses only three variables to represent lectures, tutorials, and labs which are $x_{c,d,t,r}, y_{c,d,t,r}$and $z_{c,d,t,r}$ respectively. However, their model needs to be modified as their university has a different parent-child relationship (less strict) than ours. To do this, as in the mathematics department of the RUG there are at most two lectures, tutorials, and labs per week, we use two variables for each of the three types of events. This allows us to order the variables in a way that increases control over the timetable (and satisfies the pattern required).

- $x1_{c,d,t,r,l}$: binary variable that takes 1 if course $c \in \mathcal{C}$ has the first lecture on day $d \in \mathcal{D}$ in the time slot $t \in \mathcal{T}$ in room $r \in \mathcal{R}$ taught by lecturer $l \in \mathcal{L}$ and it is 0 otherwise.
- $x2_{c,d,t,r,l}$: binary variable that takes 1 if course $c \in \mathcal{C}$ has the second lecture on day $d \in \mathcal{D}$ in the time slot $t \in \mathcal{T}$ in room $r \in \mathcal{R}$ taught by lecturer $l \in \mathcal{L}$ and it is 0 otherwise.
- $y1_{c,d,t,r}$: binary variable that takes 1 if course $c \in \mathcal{C}$ has the first tutorial on day $d \in \mathcal{D}$ in the time slot $t \in \mathcal{T}$ in room $r \in \mathcal{R}$ and it is 0 otherwise.
- $y2_{c,d,t,r}$: binary variable that takes 1 if course $c \in \mathcal{C}$ has the second tutorial on day $d \in \mathcal{D}$ in the time slot $t \in \mathcal{T}$ in room $r \in \mathcal{R}$ and it is 0 otherwise.
- $z1_{c,d,t,r}$: binary variable that takes 1 if course $c \in \mathcal{C}$ has the first computer lab on day $d \in \mathcal{D}$ in the time slot $t \in \mathcal{T}$ in room $r \in \mathcal{R}$ and it is 0 otherwise.
- $z2_{c,d,t,r}$: binary variable that takes 1 if course $c \in \mathcal{C}$ has the second computer lab on day $d \in \mathcal{D}$ in the time slot $t \in \mathcal{T}$ in room $r \in \mathcal{R}$ and it is 0 otherwise.

- $u1_{c,d,t}$: binary variable that takes 1 if course $c \in \mathcal{C}$ has a tutorial on day $d \in \mathcal{D}$ in the time slot $t \in \mathcal{T}$ and it is 0 otherwise (used to force first tutorials to happen at the same time)
- $u2_{c,d,t}$: binary variable that takes 1 if course $c \in \mathcal{C}$ has a tutorial on day $d \in \mathcal{D}$ in the time slot $t \in \mathcal{T}$ and it is 0 otherwise (used to force second tutorials to happen at the same time)
- $v1_{c,d,t}$: binary variable that takes 1 if course $c \in \mathcal{C}$ has a lab on day $d \in \mathcal{D}$ in the time slot $t \in \mathcal{T}$ and it is 0 otherwise (used to force first labs to happen at the same time)
- $v2_{c,d,t}$: binary variable that takes 1 if course $c \in \mathcal{C}$ has a lab on day $d \in \mathcal{D}$ in the time slot $t \in \mathcal{T}$ and it is 0 otherwise (used to force second labs to happen at the same time)
- used_r: binary variables that takes 1 if room r is used at least once and it is 0 otherwise.

4.2.3. Constraints

Lecturers-Courses constraints

Lecturers can only teach courses that are assigned to them. This is enforced by equation (4.1) and (4.2). Moreover, since lecturers may teach more than one course in a given trimester, we must ensure that, at any given time, a lecturer is holding at most one lecture. This is enforced by equations (4.3) and (4.4).

$$\sum_{d\in\mathcal{D}}\sum_{t\in\mathcal{T}}\sum_{r\in\mathcal{R}_{LT}} x1_{c,d,t,r,l} = M_{lc}\cdot n1_{lec,c} \quad \forall \quad c\in\mathcal{C}, l\in\mathcal{L} \tag{4.1}$$

$$\sum_{d\in\mathcal{D}}\sum_{t\in\mathcal{T}}\sum_{r\in\mathcal{R}_{LT}} x2_{c,d,t,r,l} = M_{lc}\cdot n2_{lec,c} \quad \forall \quad c\in\mathcal{C}, l\in\mathcal{L} \tag{4.2}$$

$$\sum_{c\in\mathcal{C}}\sum_{r\in\mathcal{R}_{LT}} x1_{c,d,t,r,l} \leq 1 \quad \forall \quad d\in\mathcal{D}, t\in\mathcal{T}, l\in\mathcal{L} \tag{4.3}$$

$$\sum_{c\in\mathcal{C}}\sum_{r\in\mathcal{R}_{LT}} x2_{c,d,t,r,l} \leq 1 \quad \forall \quad d\in\mathcal{D}, t\in\mathcal{T}, l\in\mathcal{L} \tag{4.4}$$

Courses-Rooms constraints

Lectures and tutorials must not be assigned to rooms that are designed for computer labs. This is enforced by equations (4.5) to (4.8). Similarly, computer labs must be scheduled in rooms with computers. This is enforced by equations (4.9) and (4.10).

$$x1_{c,d,t,r,l} = 0 \quad \forall \quad c \in \mathcal{C}, d \in \mathcal{D}, t \in \mathcal{T}, r \in \mathcal{R}_{LAB}, l \in \mathcal{L} \tag{4.5}$$

$$x2_{c,d,t,r,l} = 0 \quad \forall \quad c \in \mathcal{C}, d \in \mathcal{D}, t \in \mathcal{T}, r \in \mathcal{R}_{LAB}, l \in \mathcal{L} \tag{4.6}$$

$$y1_{c,d,t,r} = 0 \quad \forall \quad c \in \mathcal{C}, d \in \mathcal{D}, t \in \mathcal{T}, r \in \mathcal{R}_{LAB} \tag{4.7}$$

$$y2_{c,d,t,r} = 0 \quad \forall \quad c \in \mathcal{C}, d \in \mathcal{D}, t \in \mathcal{T}, r \in \mathcal{R}_{LAB} \tag{4.8}$$

$$z1_{c,d,t,r} = 0 \quad \forall \quad c \in \mathcal{C}, d \in \mathcal{D}, t \in \mathcal{T}, r \in \mathcal{R}_{LT} \tag{4.9}$$

$$z2_{c,d,t,r} = 0 \quad \forall \quad c \in \mathcal{C}, d \in \mathcal{D}, t \in \mathcal{T}, r \in \mathcal{R}_{LT} \tag{4.10}$$

Rooms capacity constraints

The following six equations guarantee that an event can be assigned to a room only if that room has enough space. Since not all students that are enrolled in the course will show up we multiply the number of enrolled students by the coefficient k_{present}. Moreover in equation (4.13) to (4.16) we divide the number of students enrolled by the number of groups. We do this because the course audience is divided into groups for tutorials and labs.

$$x1_{c,d,t,r,l} \cdot k_{\text{present}} \cdot n_c \leq n_r \quad \forall \quad c \in \mathcal{C}, d \in \mathcal{D}, t \in \mathcal{T}, r \in \mathcal{R}_{LT}, l \in \mathcal{L} \tag{4.11}$$

$$x2_{c,d,t,r,l} \cdot k_{\text{present}} \cdot n_c \leq n_r \quad \forall \quad c \in \mathcal{C}, d \in \mathcal{D}, t \in \mathcal{T}, r \in \mathcal{R}_{LT}, l \in \mathcal{L} \tag{4.12}$$

$$y1_{c,d,t,r} \cdot k_{\text{present}} \cdot \left\lceil \frac{n_c}{n_{g,c}} \right\rceil \leq n_r \quad \forall \quad c \in \mathcal{C}, d \in \mathcal{D}, t \in \mathcal{T}, r \in \mathcal{R}_{LT} \tag{4.13}$$

$$y2_{c,d,t,r} \cdot k_{\text{present}} \cdot \left\lceil \frac{n_c}{n_{g,c}} \right\rceil \leq n_r \quad \forall \quad c \in \mathcal{C}, d \in \mathcal{D}, t \in \mathcal{T}, r \in \mathcal{R}_{LT} \tag{4.14}$$

$$z1_{c,d,t,r} \cdot k_{\text{present}} \cdot \left\lceil \frac{n_c}{n_{g,c}} \right\rceil \leq n_r \quad \forall \quad c \in \mathcal{C}, d \in \mathcal{D}, t \in \mathcal{T}, r \in \mathcal{R}_{LAB} \tag{4.15}$$

$$z2_{c,d,t,r} \cdot k_{\text{present}} \cdot \left\lceil \frac{n_c}{n_{g,c}} \right\rceil \leq n_r \quad \forall \quad c \in \mathcal{C}, d \in \mathcal{D}, t \in \mathcal{T}, r \in \mathcal{R}_{LAB} \tag{4.16}$$

Correct number of events

Our timetable model must allocate the correct number of tutorials and labs for each course. Since there are multiple groups for each of these events we multiply the number of events by the number of groups. Note that we do not do this for the lectures because we have already done it in (4.1) and (4.2).

$$\sum_{d\in\mathcal{D}}\sum_{t\in\mathcal{T}}\sum_{\mathcal{R}_{LT}} y1_{c,d,t,r} = n1_{tut,c}\cdot n_{g,c} \quad \forall \quad c\in\mathcal{C} \tag{4.17}$$

$$\sum_{d\in\mathcal{D}}\sum_{t\in\mathcal{T}}\sum_{r\in\mathcal{R}_{LT}} y2_{c,d,t,r} = n2_{tut,c}\cdot n_{g,c} \quad \forall \quad c\in\mathcal{C} \tag{4.18}$$

$$\sum_{d\in\mathcal{D}}\sum_{t\in\mathcal{T}}\sum_{r\in\mathcal{R}_{LAB}} z1_{c,d,t,r} = n1_{lab,c}\cdot n_{g,c} \quad \forall \quad c\in\mathcal{C} \tag{4.19}$$

$$\sum_{d\in\mathcal{D}}\sum_{t\in\mathcal{T}}\sum_{r\in\mathcal{R}_{LAB}} z2_{c,d,t,r} = n2_{lab,c}\cdot n_{g,c} \quad \forall \quad c\in\mathcal{C} \tag{4.20}$$

Ordering of lectures constraints

As stated in the problem description we are splitting the week in two. Therefore the first lecture must happen within the first three days and the second one on the third or fourth day. This is enforced by equations (4.21) and (4.22). Moreover, they must not be scheduled on the same day [1] and there should be at least one day off between the lectures. This is guaranteed by equation (4.23) and (4.24) respectively.

$$x1_{c,d,t,r,l} = 0 \quad \forall \quad c\in\mathcal{C}, d\in\{4,5\}, t\in\mathcal{T}, r\in\mathcal{R}_{LT}, l\in\mathcal{L} \tag{4.21}$$

$$x2_{c,d,t,r,l} = 0 \quad \forall \quad c\in\mathcal{C}, d\in\{1,2,5\}, t\in\mathcal{T}, r\in\mathcal{R}_{LT}, l\in\mathcal{L} \tag{4.22}$$

$$\sum_{t\in\mathcal{T}}\sum_{r\in\mathcal{R}_{LT}}\sum_{l\in\mathcal{L}} x1_{c,3,t,r,l} + x2_{c,3,t,r,l} \leq 1 \quad \forall \quad c\in\mathcal{C} \tag{4.23}$$

$$\sum_{t\in\mathcal{T}}\sum_{r\in\mathcal{R}_{LT}}\sum_{l\in\mathcal{L}} x1_{c,d,t,r,l} + x2_{c,d+1,t,r,l} \leq 1 \quad \forall \quad c\in\mathcal{C}, d\in\mathcal{D}\backslash\{5\} \tag{4.24}$$

[1] The only day that they could be scheduled together is day 3 which is Wednesday.

Unique allocation and collisions constraints

At any given time a room can be used for, at most, one event. This is enforced by (4.25). All other equations guarantee that, for a given track and year, the different subjects' events do not collide.

$$\sum_{c\in\mathcal{C}}\left(\sum_{l\in\mathcal{L}} x1_{c,d,t,r,l}+x2_{c,d,t,r,l}\right)+y1_{c,d,t,r}+y2_{c,d,t,r}+z1_{c,d,t,r}+z2_{c,d,t,r}\leq 1 \quad \forall \quad d\in\mathcal{D},t\in\mathcal{T},r\in\mathcal{R} \tag{4.25}$$

$$\sum_{c\in\mathcal{C}_B^1}\sum_{r\in\mathcal{R}}\left(\sum_{l\in\mathcal{L}} x1_{c,d,t,r,l}+x2_{c,d,t,r,l}\right)+\frac{y1_{c,d,t,r}+y2_{c,d,t,r}+z1_{c,d,t,r}+z2_{c,d,t,r}}{n_{g,c}}\leq 1 \quad \forall \quad d\in\mathcal{D},t\in\mathcal{T} \tag{4.26}$$

$$\sum_{c\in\mathcal{C}_{BG}^2}\sum_{r\in\mathcal{R}}\left(\sum_{l\in\mathcal{L}} x1_{c,d,t,r,l}+x2_{c,d,t,r,l}\right)+\frac{y1_{c,d,t,r}+y2_{c,d,t,r}+z1_{c,d,t,r}+z2_{c,d,t,r}}{n_{g,c}}\leq 1 \quad \forall \quad d\in\mathcal{D},t\in\mathcal{T} \tag{4.27}$$

$$\sum_{c\in\mathcal{C}_{BA}^2}\sum_{r\in\mathcal{R}}\left(\sum_{l\in\mathcal{L}} x1_{c,d,t,r,l}+x2_{c,d,t,r,l}\right)+\frac{y1_{c,d,t,r}+y2_{c,d,t,r}+z1_{c,d,t,r}+z2_{c,d,t,r}}{n_{g,c}}\leq 1 \quad \forall \quad d\in\mathcal{D},t\in\mathcal{T} \tag{4.28}$$

$$\sum_{c\in\mathcal{C}_{BP}^2}\sum_{r\in\mathcal{R}}\left(\sum_{l\in\mathcal{L}} x1_{c,d,t,r,l}+x2_{c,d,t,r,l}\right)+\frac{y1_{c,d,t,r}+y2_{c,d,t,r}+z1_{c,d,t,r}+z2_{c,d,t,r}}{n_{g,c}}\leq 1 \quad \forall \quad d\in\mathcal{D},t\in\mathcal{T} \tag{4.29}$$

$$\sum_{c\in\mathcal{C}_{BG}^3}\sum_{r\in\mathcal{R}}\left(\sum_{l\in\mathcal{L}} x1_{c,d,t,r,l}+x2_{c,d,t,r,l}\right)+\frac{y1_{c,d,t,r}+y2_{c,d,t,r}+z1_{c,d,t,r}+z2_{c,d,t,r}}{n_{g,c}}\leq 1 \quad \forall \quad d\in\mathcal{D},t\in\mathcal{T} \tag{4.30}$$

$$\sum_{c\in\mathcal{C}_{BA}^3}\sum_{r\in\mathcal{R}}\left(\sum_{l\in\mathcal{L}} x1_{c,d,t,r,l}+x2_{c,d,t,r,l}\right)+\frac{y1_{c,d,t,r}+y2_{c,d,t,r}+z1_{c,d,t,r}+z2_{c,d,t,r}}{n_{g,c}}\leq 1 \quad \forall \quad d\in\mathcal{D},t\in\mathcal{T} \tag{4.31}$$

$$\sum_{c\in\mathcal{C}_{BP}^3}\sum_{r\in\mathcal{R}}\left(\sum_{l\in\mathcal{L}} x1_{c,d,t,r,l}+x2_{c,d,t,r,l}\right)+\frac{y1_{c,d,t,r}+y2_{c,d,t,r}+z1_{c,d,t,r}+z2_{c,d,t,r}}{n_{g,c}}\leq 1 \quad \forall \quad d\in\mathcal{D},t\in\mathcal{T} \tag{4.32}$$

$$\sum_{c\in\mathcal{C}_{MG}}\sum_{r\in\mathcal{R}}\left(\sum_{l\in\mathcal{L}} x1_{c,d,t,r,l}+x2_{c,d,t,r,l}\right)+\frac{y1_{c,d,t,r}+y2_{c,d,t,r}+z1_{c,d,t,r}+z2_{c,d,t,r}}{n_{g,c}}\leq 1 \quad \forall \quad d\in\mathcal{D},t\in\mathcal{T} \tag{4.33}$$

$$\sum_{c\in\mathcal{C}_{MA}}\sum_{r\in\mathcal{R}}\left(\sum_{l\in\mathcal{L}} x1_{c,d,t,r,l}+x2_{c,d,t,r,l}\right)+\frac{y1_{c,d,t,r}+y2_{c,d,t,r}+z1_{c,d,t,r}+z2_{c,d,t,r}}{n_{g,c}}\leq 1 \quad \forall \quad d\in\mathcal{D},t\in\mathcal{T} \tag{4.34}$$

Groups have labs and tutorials at the same time

Using the auxiliary variables $u1, u2, v1, v2$, if one group has an event (tutorial or lab) at a given time, then all groups will have the same type of event at that same time.

$$\sum_{r \in \mathcal{R}_{LT}} y1_{c,d,t,r} = u1_{c,d,t} \cdot n1_{tut,c} \cdot n_{g,c} \quad \forall \quad c \in \mathcal{C}, d \in \mathcal{D}, t \in \mathcal{T} \tag{4.35}$$

$$\sum_{r \in \mathcal{R}_{LT}} y2_{c,d,t,r} = u2_{c,d,t} \cdot n2_{tut,c} \cdot n_{g,c} \quad \forall \quad c \in \mathcal{C}, d \in \mathcal{D}, t \in \mathcal{T} \tag{4.36}$$

$$\sum_{r \in \mathcal{R}_{LAB}} z1_{c,d,t,r} = v1_{c,d,t} \cdot n1_{lab,c} \cdot n_{g,c} \quad \forall \quad c \in \mathcal{C}, d \in \mathcal{D}, t \in \mathcal{T} \tag{4.37}$$

$$\sum_{r \in \mathcal{R}_{LAB}} z2_{c,d,t,r} = v2_{c,d,t} \cdot n2_{lab,c} \cdot n_{g,c} \quad \forall \quad c \in \mathcal{C}, d \in \mathcal{D}, t \in \mathcal{T} \tag{4.38}$$

Pattern for tutorials and labs constraint

Lectures and laboratories must occur on the day after their respective lectures. This is enforced by equations (4.39) to (4.42). Equations (4.43) to (4.46) guarantee that tutorials and labs are not scheduled on Mondays.

Equations (4.47) and (4.48) satisfy the constraint "laboratories are scheduled in the timeslot after the tutorial". Equations (4.49) and (4.50) guarantee that, if there is a tutorial and a lab assigned to a lecture, then the lab does not happen in the first timeslot.

$$\begin{cases} \sum_{r \in \mathcal{R}_{LT}} \sum_{t \in \mathcal{T}} x1_{c,d,t,r,l} - \frac{y1_{c,d+1,t,r}}{n_{g,c}} \leq 0, & \text{if } n1_{tut,c} > 0 \\ \text{no constraint}, & \text{otherwise} \end{cases} \quad \forall \quad c \in \mathcal{C}, d \in \mathcal{D} \backslash \{5\}, l \in \mathcal{L} \tag{4.39}$$

$$\begin{cases} \sum_{r \in \mathcal{R}_{LT}} \sum_{t \in \mathcal{T}} x2_{c,d,t,r,l} - \frac{y2_{c,d+1,t,r}}{n_{g,c}} \leq 0, & \text{if } n2_{tut,c} > 0 \\ \text{no constraint}, & \text{otherwise} \end{cases} \quad \forall \quad c \in \mathcal{C}, d \in \mathcal{D} \backslash \{5\}, l \in \mathcal{L} \tag{4.40}$$

$$\begin{cases} \sum_{r \in R} \sum_{t \in \mathcal{T}} x1_{c,d,t,r,l} - \frac{z1_{c,d+1,t,r}}{n_{g,c}} \leq 0, & \text{if } n1_{lab,c} > 0 \text{ and } n1_{tut,c} = 0 \\ \text{no constraint}, & \text{otherwise} \end{cases} \quad \forall \quad c \in \mathcal{C}, d \in \mathcal{D} \backslash \{5\}, l \in \mathcal{L} \tag{4.41}$$

$$\begin{cases} \sum_{r\in R}\sum_{t\in\mathcal{T}} x2_{c,d,t,r,l} - \frac{z2_{c,d+1,t,r}}{n_{g,c}} \leq 0, & \text{if } n2_{lab,c} > 0 \text{ and } n2_{tut,c} = 0 \\ \text{no constraint}, & \text{otherwise} \end{cases} \quad \forall \quad c \in \mathcal{C}, d \in \mathcal{D}\backslash\{5\}, l \in \mathcal{L} \tag{4.42}$$

$$y1_{c,1,t,r} = 0 \quad \forall \quad c \in \mathcal{C}, t \in \mathcal{T}, r \in \mathcal{R} \tag{4.43}$$

$$y2_{c,1,t,r} = 0 \quad \forall \quad c \in \mathcal{C}, t \in \mathcal{T}, r \in \mathcal{R} \tag{4.44}$$

$$z1_{c,1,t,r} = 0 \quad \forall \quad c \in \mathcal{C}, t \in \mathcal{T}, r \in \mathcal{R} \tag{4.45}$$

$$z2_{c,1,t,r} = 0 \quad \forall \quad c \in \mathcal{C}, t \in \mathcal{T}, r \in \mathcal{R} \tag{4.46}$$

$$\begin{cases} \sum_{r\in R} z1_{c,d,t,r} - y1_{c,d,t-1,r} \leq 0, & \text{if } n1_{lab,c} > 0 \\ \text{no constraint}, & \text{otherwise} \end{cases} \quad \forall \quad c \in \mathcal{C}, d \in \mathcal{D}, t \in \mathcal{T}\backslash\{1\} \tag{4.47}$$

$$\begin{cases} \sum_{r\in R} z2_{c,d,t,r} - y2_{c,d,t-1,r} \leq 0, & \text{if } n2_{lab,c} > 0 \\ \text{no constraint}, & \text{otherwise} \end{cases} \quad \forall \quad c \in \mathcal{C}, d \in \mathcal{D}, t \in \mathcal{T}\backslash\{1\} \tag{4.48}$$

$$\begin{cases} z1_{c,d,1,r} = 0, & \text{if } n1_{tut,c} > 0 \\ \text{no constraint}, & \text{otherwise} \end{cases} \quad \forall \quad c \in \mathcal{C}, d \in \mathcal{D}, r \in \mathcal{R}_{LAB} \tag{4.49}$$

$$\begin{cases} z2_{c,d,1,r} = 0, & \text{if } n2_{tut,c} > 0 \\ \text{no constraint}, & \text{otherwise} \end{cases} \quad \forall \quad c \in \mathcal{C}, d \in \mathcal{D}, r \in \mathcal{R}_{LAB} \tag{4.50}$$

Rooms used

The binary variable used_r determines whether room r is used. Note that, for a given combination of c, d, t, and l, at most one among the variables, x's,y's, and z's is equal to one. Therefore, $|S|$ is an upper bound of the right-hand side of the inequality. Thus, if a room is used even once, used_r will be equal to 1 (and 0 otherwise).

$$\sum_{c\in\mathcal{C}}\sum_{d\in\mathcal{D}}\sum_{t\in\mathcal{T}}\left(\sum_{l\in\mathcal{L}} x1_{c,d,t,r,l} + x2_{c,d,t,r,l}\right) + y1_{c,d,t,r} + y2_{c,d,t,r} + z1_{c,d,t,r} + z2_{c,d,t,r} \leq |S| \cdot \text{used}_r \quad \forall \quad r \in \mathcal{R}. \tag{4.51}$$

Objective function

The objectives function is a combination of three different sub-objectives. Equation (4.52) measures the number of events that take place in the last timeslot (i.e 17 to 19). Equation (4.53) quantifies the number of wasted seating spaces while (4.54) the number of rooms that have been used.

$$f_1 = \sum_{c\in\mathcal{C}}\sum_{d\in\mathcal{D}}\sum_{r\in\mathcal{R}}\left(\sum_{l\in\mathcal{L}} x1_{c,d,5,r,l} + x2_{c,d,5,r,l}\right) + y1_{c,d,5,r} + y2_{c,d,5,r} + z1_{c,d,5,r} + z2_{c,d,5,r} \tag{4.52}$$

$$\begin{aligned} f_2 = &\sum_{c\in\mathcal{C}}\sum_{d\in\mathcal{D}}\sum_{t\in\mathcal{T}}\sum_{r\in\mathcal{R}}\left(\sum_{l\in\mathcal{L}} x1_{c,d,t,r,l} + x2_{c,d,t,r,l}\right) \cdot (n_r - k_{\text{present}} \cdot n_c) \\ &+ (n_r - \frac{k_{\text{present}} \cdot n_c}{n_{g,c}}) \cdot (y1_{c,d,t,r} + y2_{c,d,t,r} + z1_{c,d,t,r} + z2_{c,d,t,r}) \end{aligned} \tag{4.53}$$

$$f_3 = \sum_{r\in\mathcal{R}} \text{used}_r \tag{4.54}$$

We want to try to minimize all three. In order to do this we consider the following two objective functions.

$$z_1 = f_1 + f_2 + f3 \qquad \text{(Model 1)}$$

$$z_2 = f_1 + 0.1 \cdot f_2 + f3 \qquad \text{(Model 2)}$$

Model 1 gives an equal weight to all three soft constraints while model two gives more importance to rooms used and events taking place in the last timeslot rather than wasted space.

4.3. Analysis of results

The model was implemented in Python 3.12 using the AMPL Python API (Fourer, Gay & Kernighan, 2003) and it was solved using HiGHS (Huangfu & Hall, 2018), an open solver for optimization problems. All the tests were executed using an Intel Core i7-12700H 4.7GHz and 16GB RAM. The code may be found on https://github.com/scinii/timetabling_RUG.

We test our two models on four different instances. Each instance corresponds to a block (or trimester) at the University of Groningen. Table 4.1 provides a general overview of each block (for a complete description see Appendix A.2).

Block	#Events	#Lectures	#Labs	#Tutorials
1A	109	42	2	65
1B	102	45	2	55
2A	81	34	10	37
2B	65	20	8	37

Table 4.1.: Overview of each block's events

Block	z_1	Room	17-19	Wasted Space	Time (min)
1A	714.8	22	4	688.8	0.493
1B	533	24	4	505	1.188
2A	424.4	26	0	398.4	0.397
2B	302.4	26	0	276.4	0.339

Table 4.2.: Results Model 1 ($z = f_1 + f_2 + f_3$)

Block	z_2	Room	17-19	Wasted Space	Time (min)
1A	90.48	18	0	724.8	55.528
1B	74.5	19	1	545	171.577
2A	64.84	22	1	418.4	52.479
2B	51.24	22	0	292.4	23.846

Table 4.3.: Results Model 2 ($z = f_1 + 0.1 \cdot f_2 + f_3$)

Block	Model 1	Model 2
1A	6.3	6.7
1B	4.9	5.3
2A	4.9	5.2
2B	4.3	4.5

Table 4.4.: Average wasted seating spaces per event for each block

As it is possible to see from the two tables above 4.3 and 4.2 the two models give us very different results [2]. In Model 1 (table 4.2) the average number of wasted spaces per event (see table 4.4) goes from a minimum of 4.3 to a maximum of 6.3. Moreover, there is a total of 8 events scheduled from 17 to 19 and the number of rooms used goes from a minimum of 22 to a maximum of 26.

On the other side, in Model 2, the number of wasted spaces increases as scaling the function f_2 is equivalent to giving it more slack. However, the number of events scheduled in the fifth timeslot goes to 2 and the number of rooms decreases to a minimum of 18, reaching a maximum of 22. Another evident difference between the two models is the average running time: Model 2 takes significantly more time than the other.

While the two models provide us different results they have two similar characteristics. The first one is obvious: as the size of the problem increases the average running time increases. The second, and more interesting one, is that even if blocks 1A and 1B have a similar number of total events and type of events the latter takes much longer than the former. The reason for this "odd" behavior resides in the fact that our university uses a curriculum course-based timetabling system. In block 1B, third-year students choose their electives which means that several courses should be available for all the different tracks. Therefore the computations needed to find a conflict-free timetable lead to an increase in average running time.

Figure 4.1 shows a visual output of our model. On the horizontal axis, we have the five timeslots while on the vertical axis, we have the rooms that are used within that block (in this case 2A). The orange, light blue, and purple represent lectures, tutorials, and computer labs respectively. To see all the visual outputs see Appendix A.3

[2] In order to compute the time we run each instance multiple times and take the average. This is because the running time depends on the initial vertex choice. Therefore the average allows us to give a more realistic running time.

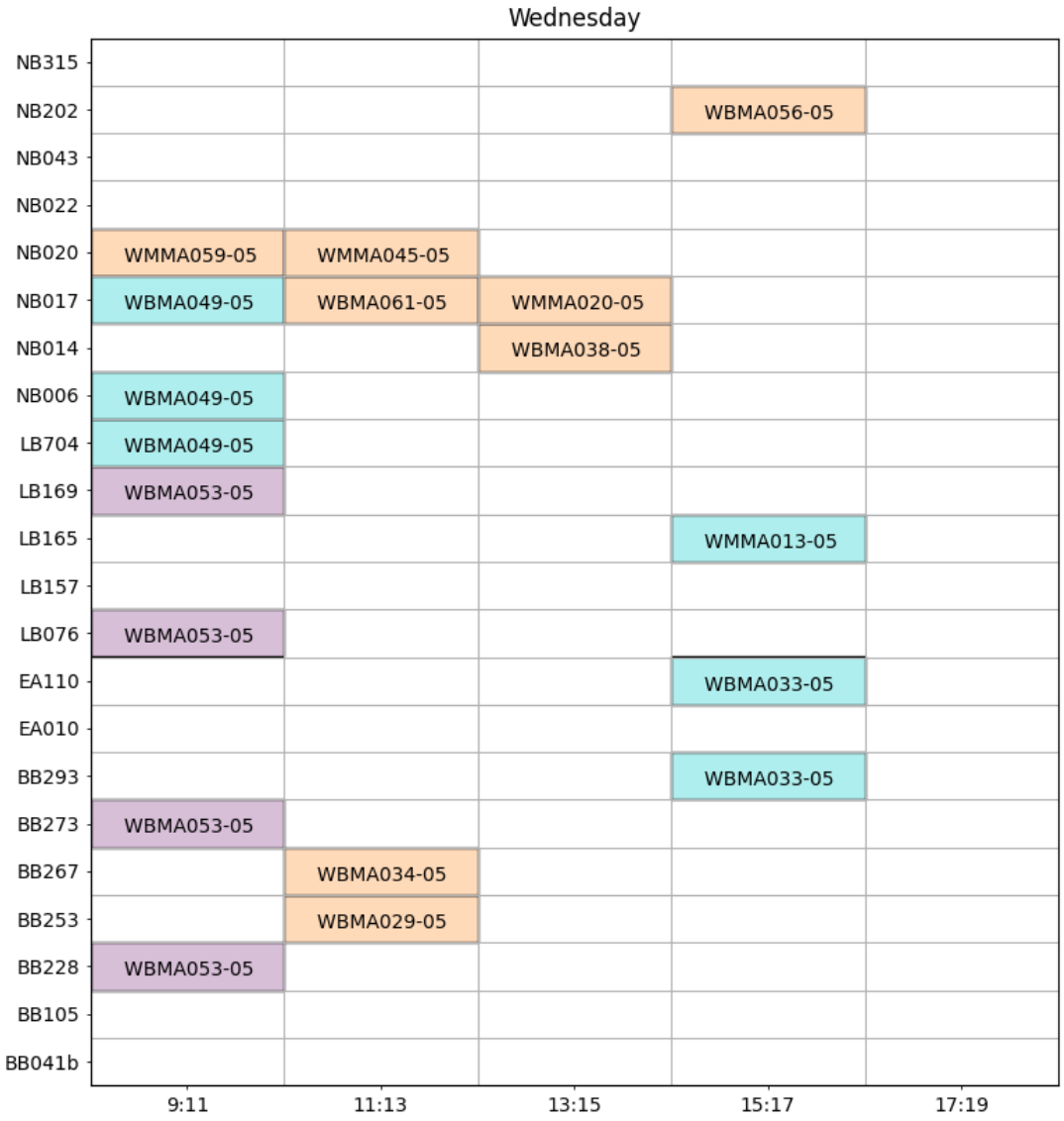

Figure 4.1.: Sample output for Wednesday Block 2A

5. Conclusion

In this thesis, we developed an integer programming model for timetabling at the University of Groningen (RUG). To do this we introduced (integer) linear programming, proving some fundamental results which later allowed us to discuss the algorithms and methods used to solve these special optimization problems.

Furthermore, we introduced our model which provides a higher degree of control over the timetable (in particular a stronger parent-child relationship between the events) than (Havås et al., 2013). Our model was implemented in AMPL and Python and later was tested (using the HiGHS solver) with 4 datasets obtained via the mathematics department of the RUG. All the tests were successful as the constraints were satisfied and the results showed us it is possible to improve how the university resources are used.

Even though our model provides us solid results it still possesses some limitations. In this section, we briefly discuss them and their possible solutions.

Extensions to the entire faculty

As of now the model was only aimed at and tested on the mathematics department of the RUG. Ideally, we would want the model to be extended to the entire Faculty of Science and Engineering (and possibly to the entire university). A possible way to do this would be the following. Use our model (or a slight variant of it) to determine the minimum number of rooms needed by similar departments (for example physics and mathematics or computer science and artificial intelligence). Then we could partition the university and assign a set of rooms to those departments only. Then our running time would decrease as the total number of variables would be smaller.

Objectives scaling

In the thesis, we considered two models: one in which we sum the 3 sub-objective functions and one in which, before summing, we scaled the wasted space function (i.e. f_2) by a factor 0.1. The rationale behind this choice was to match the magnitude of

f_2 with the magnitude of the other two sub-objectives. However, this is likely not the optimal choice. A possible extension should, therefore, research appropriate scaling factors for three different functions and in particular f_2. However, this scaling may negatively affect the running time of the model. To diminish this effect we suggest using more powerful machines and using a different solver, for example, CPLEX (Cplex, 2009).

Student sectioning

As we have seen in both models it takes a longer time to find the optimal solutions for block 1B, as in that trimester third-year students can choose among several electives that are available to all the tracks. This creates a problem as now many courses must not be scheduled at the same time. To overcome this we could employ student sectioning. This is itself a problem which consists of partitioning the student sets to avoid any scheduling conflicts.

6. Bibliography

Albers, Donald J. and Constance Reid (1986). "An Interview with George B. Dantzig: The Father of Linear Programming". In: *The College Mathematics Journal* 17.4, pp. 292–314. DOI: 10.1080/07468342.1986.11972971. eprint: https://doi.org/10.1080/07468342.1986.11972971. URL: https://doi.org/10.1080/07468342.1986.11972971.

Andreasson, Niclas, Anton Evgrafov and Michael Patriksson (2005). *An introduction to continuous optimization : foundations and fundamental algorithms*. English. Lund, Sweden: Studentlitteratur. ISBN: 9144044550; 9789144044552.

Bertsimas, Dimitris and John Tsitsiklis (1997). *Introduction to Linear Optimization*. 1st. Athena Scientific. ISBN: 1886529191.

Bland, Robert G. (1977). "New Finite Pivoting Rules for the Simplex Method". In: *Mathematics of Operations Research* 2.2, pp. 103–107. ISSN: 0364765X, 15265471. URL: http://www.jstor.org/stable/3689647 (visited on 26/06/2024).

Ceschia, Sara, Luca Di Gaspero and Andrea Schaerf (2023). "Educational timetabling: Problems, benchmarks, and state-of-the-art results". In: *European Journal of Operational Research* 308.1, pp. 1–18. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2022.07.011. URL: https://www.sciencedirect.com/science/article/pii/S0377221722005641.

Chakraborty, Atlanta, Vijay Chandru and M. R. Rao (2020). "A linear programming primer: from Fourier to Karmarkar". In: *Annals of Operations Research* 287.2, pp. 593–616. ISSN: 1572-9338. DOI: 10.1007/s10479-019-03186-2. URL: https://doi.org/10.1007/s10479-019-03186-2.

Chen, Mei Ching et al. (2021). "A Survey of University Course Timetabling Problem: Perspectives, Trends and Opportunities". In: *IEEE Access* 9, pp. 106515–106529. DOI: 10.1109/ACCESS.2021.3100613.

Cornuéjols, Gérard (2008). "Valid inequalities for mixed integer linear programs". In: *Mathematical Programming* 112.1, pp. 3–44. ISSN: 1436-4646. DOI: 10.1007/s10107-006-0086-0. URL: https://doi.org/10.1007/s10107-006-0086-0.

Cplex, IBM ILOG (2009). "V12. 1: User's Manual for CPLEX". In: *International Business Machines Corporation* 46.53, p. 157.

Dantzig, G. B. (1951). "Maximization of a linear function of variables subject to linear inequalities". In: *Activity Analysis of Production and Allocation*. Ed. by Tjalling C. Koopmans. Proceedings of a Conference (Proceedings Conference on Linear Programming, Chicago, Illinois, 1949). New York: Wiley, pp. 339–347.

Dias Rasteiro, Deolinda M.L. (2020). "9 - Shortest path problem and computer algorithms". In: *Calculus for Engineering Students*. Ed. by Jesús Martín-Vaquero et al. Mathematics in Science and Engineering. Academic Press, pp. 179–195. DOI: https://doi.org/10.1016/B978-0-12-817210-0.00016-3. URL: https://www.sciencedirect.com/science/article/pii/B9780128172100000163.

Fourer, R., D.M. Gay and B.W. Kernighan (2003). *AMPL: A Modeling Language for Mathematical Programming*. Scientific Press series. Thomson/Brooks/Cole. ISBN: 9780534388096. URL: https://books.google.nl/books?id=Ij8ZAQAAIAAJ.

Groningen, University of (2023). *Deadlines timetabling Faculty of Science and Engineering*. URL: https://www.rug.nl/fse/timetabling/deadlines-timetabling (visited on 27/11/2023).

Havås, Johan et al. (2013). "Modeling and optimization of university timetabling - A case study in integer programming". In: URL: https://api.semanticscholar.org/CorpusID:59868847.

Huangfu, Q. and J. A. J. Hall (2018). "Parallelizing the dual revised simplex method". In: *Mathematical Programming Computation* 10.1, pp. 119–142. DOI: 10.1007/s12532-017-0130-5.

Huiberts, Sophie (2022). "Geometric aspects of linear programming : shadow paths, central paths, and a cutting plane method". PhD thesis.

Kutateladze, Semen (Mar. 2012). "MATHEMATICS AND ECONOMICS IN THE LEGACY OF LEONID KANTOROVICH". In: *Владикавказский математический журнал*. DOI: 10.23671/VNC.2012.14.10950.

Lessard, Laurent (2017-18). *Cutting planes, branch and bound*.

Mahlous, Ahmed Redha and Houssam Mahlous (Feb. 2023). "Student timetabling genetic algorithm accounting for student preferences". en. In: *PeerJ Comput. Sci.* 9, e1200.

Mccollum, Barry (Aug. 2006). "A Perspective on Bridging the Gap Between Theory and Practice in University Timetabling". In: vol. 3867, pp. 3–23. ISBN: 978-3-540-77344-3. DOI: 10.1007/978-3-540-77345-0_1.

Morrison, David R. et al. (2016). "Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning". In: *Discrete Optimization* 19, pp. 79–102. ISSN: 1572-5286. DOI: https://doi.org/10.1016/j.disopt.2016.01.005. URL: https://www.sciencedirect.com/science/article/pii/S1572528616000062.

Müller, Tomáš, Hana Rudová and Zuzana Müllerová (2024). "Real-world university course timetabling at the International Timetabling Competition 2019". In: *Journal of Scheduling*. ISSN: 1099-1425. DOI: 10.1007/s10951-023-00801-w. URL: https://doi.org/10.1007/s10951-023-00801-w.

Ploskas, Nikolaos and Nikolaos Samaras (Jan. 2014). "Pivoting rules for the revised simplex algorithm". In: *Yugoslav Journal of Operations Research* 24, pp. 321–332. DOI: 10.2298/YJOR140228016P.

Rader, David J. (2010). *Deterministic operations research : models and methods in linear optimization*. English. Hoboken, N.J.: John Wiley & Sons, Inc. ISBN: 9780470484517; 0470484519.

Robbins, Henry W. et al. (2023). "GILP: An Interactive Tool for Visualizing the Simplex Algorithm". In: *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*. SIGCSE 2023. Association for Computing Machinery, pp. 108–114. DOI: 10.1145/3545945.3569815.

Roma, Massimo (2019). *Appunti dalle lezioni di Ricerca Operativa*.

Rudová, Hana, Tomáš Müller and Keith Murray (2011). "Complex university course timetabling". In: *Journal of Scheduling* 14.2, pp. 187–207. ISSN: 1099-1425. DOI: 10.1007/s10951-010-0171-3. URL: https://doi.org/10.1007/s10951-010-0171-3.

Schmidt, G. and T. Ströhlein (Jan. 1980). "Timetable construction – an annotated bibliography". In: *The Computer Journal* 23.4, pp. 307–316. ISSN: 0010-4620. DOI: 10.1093/comjnl/23.4.307. eprint: https://academic.oup.com/comjnl/article-pdf/23/4/307/993940/230307.pdf. URL: https://doi.org/10.1093/comjnl/23.4.307.

Soyemi, Jumoke, Akinode John Lekan and Abiodun Oloruntoba (Aug. 2017). "Automated Lecture Time-tabling System for Tertiary Institutions". In: *International Journal of Applied Information Systems (IJAIS). Foundation of Computer Science FCS, New York, USA* 12, pp. 21–27. DOI: 10.5120/ijais2017451700.

Tehranchi, Michael (2017a). *IB Optimisation: Lecture 5*. URL: https://www.statslab.cam.ac.uk/~mike/optimisation/lecture5.pdf.

— (2017b). *The fundamental theorem of linear programming*. URL: https://www.statslab.cam.ac.uk/~mike/optimisation/linearprogram.pdf.

Vanderbei, Robert J. (2020). *Linear programming : foundations and extensions*. English. International Series in Operations Research & Management Science. Boston: Springer Cham. ISBN: 978-3-030-39415-8; 978-3-030-39414-1.

A. Appendix

A.1. Duality

To solve our timetabling problem we use a solver known as HiGHS (Huangfu & Hall, 2018). It solves either the primal or the dual problem. Therefore below we provide a brief recap of duality. For more information regarding the Dual Simplex Method see section 4.5 of Bertsimas and Tsitsiklis, 1997.

Consider the following LP problem

$$\begin{aligned} \text{maximize} \quad & c^T x \\ \text{subject to} \quad & Ax \leq b, \\ & x \geq 0 \end{aligned} \tag{P}$$

We call this the primal problem. Its corresponding dual problem is the following (Tehranchi, 2017a):

$$\begin{aligned} \text{minimize} \quad & b^T y \\ \text{subject to} \quad & A^T y \geq c, \\ & t \geq 0 \end{aligned} \tag{D}$$

Define the two following quantities:

$$p = \sup\{c^T x : x \in \mathbb{R}^n, Ax \leq b, x \geq 0\}$$

and

$$d = \inf\{b^T y : y \in \mathbb{R}^m, A^T y \geq c, y \geq 0\}$$

with the convention that $\sup \emptyset = -\infty$ and $\inf \emptyset = \infty$. Then we have two important results (Tehranchi, 2017b).

Theorem 9. *(Weak duality) For p and d as above we have $p \leq d$.*

Theorem 10. *(Strong duality) For p and d as above if either $p > -\infty$ or $d < \infty$ then $p = d$.*

A.2. Data

Room	Capacity	Type
EA010	40	LT
EA029	203	LT
EA058	28	LT
EA062	44	LT
EA110	38	LT
EA114	34	LT
BB041b	36	LT
BB105	120	LT
BB151	294	LT
BB222	36	LT
BB253	130	LT
BB267	89	LT
BB289	40	LT
BB293	40	LT
LB704	22	LT
LB045	24	LT
LB050	30	LT
LB055	60	LT
LB141	34	LT
LB149	36	LT
LB151	30	LT
LB157	34	LT
LB165	32	LT
LB176	24	LT
LB217	24	LT
LB034	24	LT
NB006	22	LT
NB022	200	LT
NB080	145	LT
NB012	28	LT
NB104	28	LT
NB201	30	LT
NB202	40	LT
NB004	40	LT
NB043	56	LT
NB008	18	LT
NB013	30	LT
NB014	16	LT

NB017	24	LT
NB020	14	LT
NB317	40	LT
NB-152	56	LT
NB-156	58	LT
NB161	28	LT
BB204	12	LAB
BB207	14	LAB
BB208	17	LAB
BB216	18	LAB
BB228	30	LAB
BB273	32	LAB
BB283	24	LAB
LB076	32	LAB
LB169	45	LAB
NB057	48	LAB
NB071	48	LAB
NB315	22	LAB

Table A.1.: Avaliable Rooms

Course Code	Track 1	Track	#Students	#Groups	#Lec 1	# Lec 2	# Tut 1	# Tut 2	# Lab 1	# Lab 2	Lecturer
WBMA003-05	B1	B1	150	5	1	1	1	1	0	0	L1
WBMA020-05	B1	B1	168	5	1	1	1	1	0	0	L2
WBMA051-05	B1	B1	167	5	1	1	1	1	0	0	L3
WBMA005-05	B2	BA3	106	4	1	1	1	1	0	0	L4
WBMA036-05	B2	B2	112	3	1	1	1	1	0	0	L5
WBMA009-05	B2	B2	80	3	1	1	1	1	0	0	L6
WBMA054-05	B3	BA2	45	1	1	1	1	1	0	0	L7
WBMA058-05	BG3	BG3	23	1	1	1	1	1	0	0	L8
WBMA004-05	BA3	BA3	14	1	1	1	0	1	0	1	L9
WBMA057-05	BG3	BG3	29	1	1	1	1	1	0	0	L4
WBMA059-05	BG3	BG3	14	1	1	1	1	1	0	0	L10
WMMA012-05	MA	MA	9	1	1	1	0	0	0	0	L11
WMMA054-05	MA	MA	6	1	1	1	1	0	0	0	L12
WMMA021-05	MA	MA	15	1	1	1	0	0	0	0	L13
WMMA015-05	MA	MA	22	1	1	1	0	0	1	0	L6
WMMA039-05	MA	MA	17	1	1	1	0	0	0	0	L14
WMMA049-05	MG	MG	10	1	1	1	1	1	0	0	L15
WMMA019-05	MG	MG	10	1	1	1	0	0	0	0	L16
WMMA043-05	MG	MG	8	1	1	1	1	0	0	0	L17
WMMA033-05	MG	MG	9	1	1	1	1	1	0	0	L18
WMMA042-05	MG	MG	5	1	1	1	0	0	0	0	L19

Table A.2.: Courses for Block 1A and their requirements

Course Code	Track 1	Track	#Students	#Groups	#Lec 1	# Lec 2	# Tut 1	# Tut 2	# Lab 1	# Lab 2	Lecturer
WBMA012-05	B1	B1	173	4	1	1	1	1	0	0	L1
WBMA052-05	B1	B1	145	4	1	1	1	1	0	0	L2
WBMA060-05	B1	B1	28	1	1	1	1	1	0	0	L3
WBMA018-05	B2	B2	102	3	1	1	1	1	0	0	L4
WBMA019-05	BG3	BP2	43	1	1	1	1	1	0	0	L2
WBMA022-05	B2	B2	100	3	1	1	1	1	0	0	L5
WBMA031-05	BA2	BG2	91	3	1	1	1	1	0	0	L6
WBMA011-05	B3	B3	26	1	1	1	1	1	0	0	L7
WBMA013-05	B3	B3	25	1	1	1	1	1	0	0	L8
WBMA048-05	B3	B3	33	1	1	1	1	1	0	0	L9
WBMA028-05	BA2	BP2	23	1	1	0	1	0	0	1	L10
WBMA023-05	BA3	BA3	13	1	1	1	1	0	0	0	L11
WBMA001-05	BA3	BA3	11	1	1	1	1	0	0	0	L12
WMMA051-05	MA	MA	11	1	1	1	0	0	0	1	L13
WMMA057-05	MA	MA	14	1	1	1	1	0	0	0	L14
WMMA058-05	MA	MA	12	1	1	1	0	0	0	0	L15
WMMA056-05	MA	MA	8	1	1	1	1	1	0	0	L16
WMMA061-05	MA	MA	18	1	1	1	1	1	0	0	L17
WMMA037-05	MG	MG	1	1	1	1	0	0	0	0	L18
WMMA047-05	MG	MG	6	1	1	1	0	0	0	0	L8
WMMA018-05	MG	MG	15	1	1	1	1	1	0	0	L19
WMMA040-05	MG	MG	6	1	1	1	1	0	0	0	L20
WMMA048-05	MG	MG	15	1	1	1	0	0	0	0	L21

Table A.3.: Courses for Block 1B and their requirements

Course Code	Track 1	Track 2	#Students	#Groups	#Lec 1	#Lec 2	#Tut 1	#Tut 2	#Lab 1	#Lab 2	Lecturer
WBMA029-05	B1	B1	159	4	1	1	1	1	0	0	L1
WBMA035-05	B1	B1	186	4	1	1	1	1	0	0	L2
WBMA053-05	B1	B1	144	4	1	1	0	0	1	1	L3
WBMA033-05	B2	B2	91	2	1	1	1	1	0	0	L4
WBMA034-05	BG2	BP2	85	2	1	1	1	1	0	0	L5
WBMA038-05	BP2	BP2	20	1	1	1	1	0	0	1	L6
WBMA026-05	BG2	BP2	27	1	1	1	0	1	0	0	L7
WBMA061-05	BA2	BA2	29	1	1	1	1	1	0	0	L8
WBMA027-05	BA2	BA2	17	1	1	1	0	0	0	0	L9
WBMA056-05	B3	B3	48	1	1	0	0	0	0	0	L10
WMMA013-05	M	M	38	1	1	0	1	0	0	0	L11
WMMA055-05	MA	MA	4	1	1	1	0	1	0	0	L12
WMMA059-05	MA	MA	13	1	1	1	0	0	0	0	L13
WMMA020-05	MA	MA	28	1	1	1	0	0	0	0	L14
WMMA008-05	MA	MA	23	1	1	1	0	0	1	0	L15
WMMA045-05	MG	MG	16	1	1	1	0	1	0	0	L16
WMMA035-05	MG	MG	8	1	1	1	0	0	0	0	L17
WBMA049-05	B2	B3	70	3	1	1	1	1	0	0	L18

Table A.4.: Courses for Block 2A and their requirements

Course Code	Track 1	Track 2	#Students	#Groups	#Lec 1	#Lec 2	#Tut 1	#Tut 2	#Lab 1	#Lab 2	Lecturer
WBMA040-05	B1	B1	135	1	1	1	0	0	0	0	L1
WBMA043-05	B1	B1	171	4	1	1	1	1	0	0	L2
WBMA046-05	B1	B1	187	4	1	1	1	1	0	0	L3
WBMA007-05	BA2	BA2	37	1	1	1	0	0	0	0	L4
WBMA045-05	B2	B2	160	4	1	1	1	1	1	1	L5
WBMA008-05	B2	B2	110	3	1	1	1	1	0	0	L6
WBMA039-05	BG2	BG2	74	1	1	1	1	1	0	0	L7
WBMA024-05	BP2	BP2	46	1	1	1	1	1	0	0	L3
WMMA029-05	M	M	30	1	1	1	1	0	0	0	L9
WMMA046-05	MG	MG	14	1	1	1	1	1	0	0	L10

Table A.5.: Courses for Block 2B and their requirements

A.3. Figures

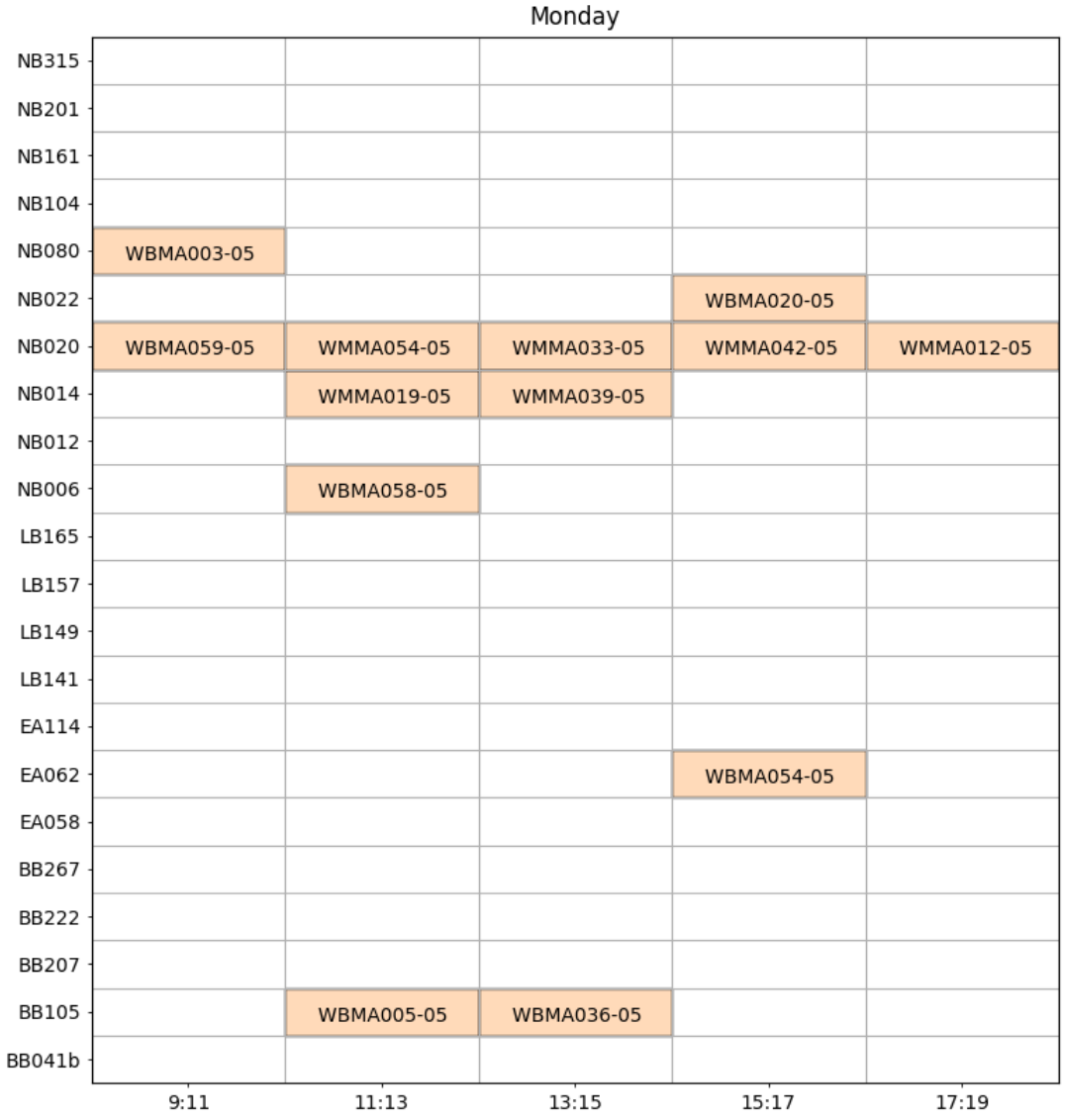

Figure A.1.: Block 1A - Model 1

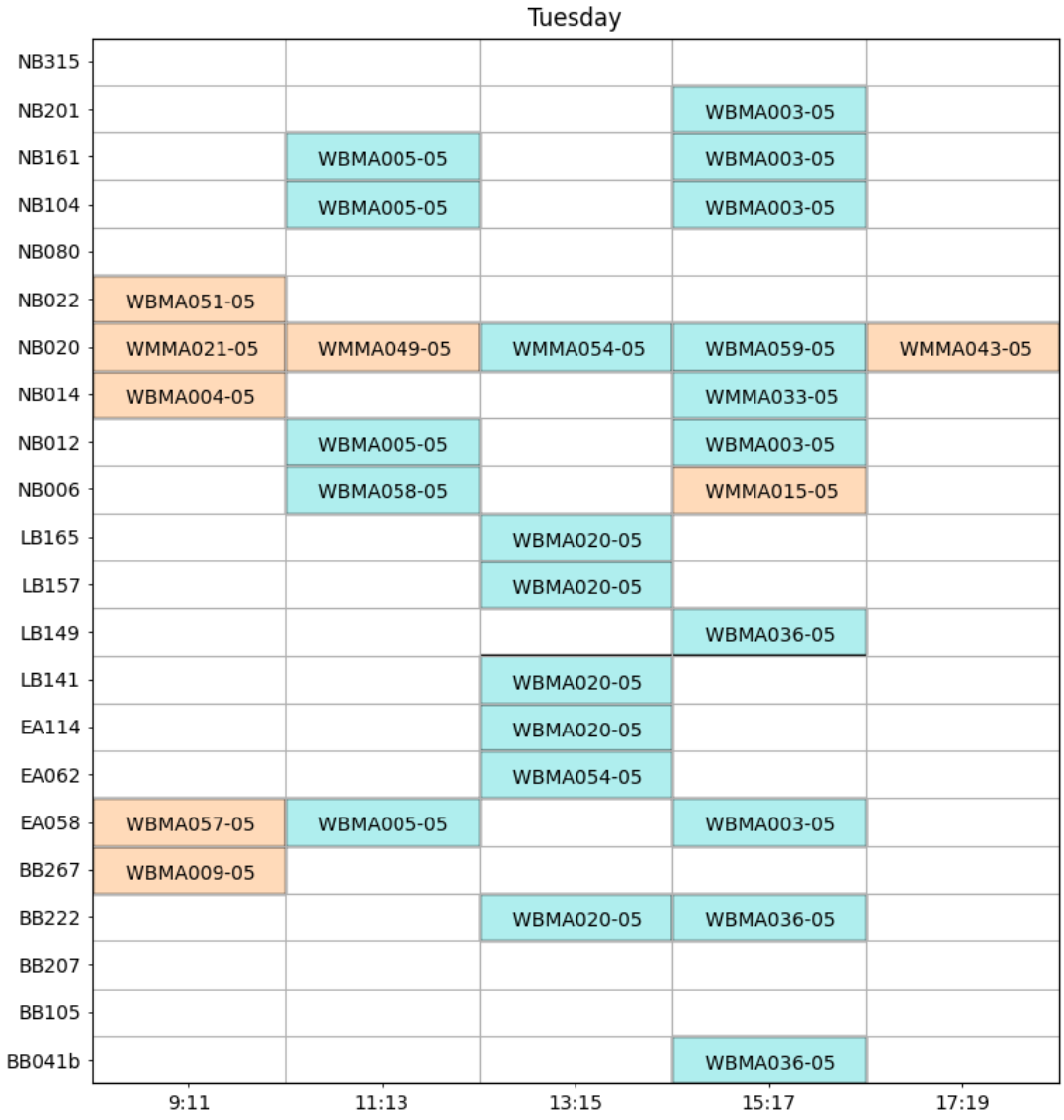

Figure A.2.: Block 1A - Model 1

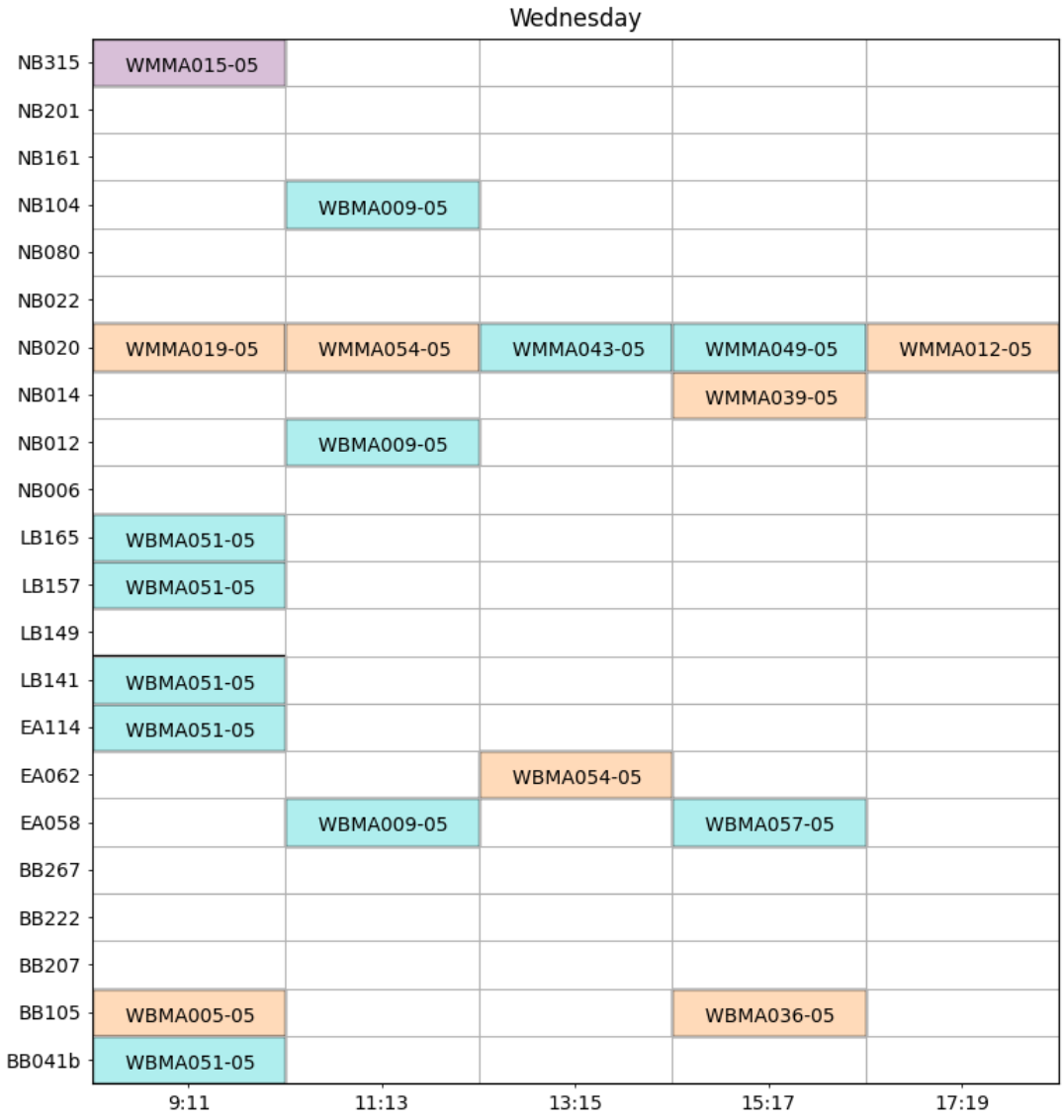

Figure A.3.: Block 1A - Model 1

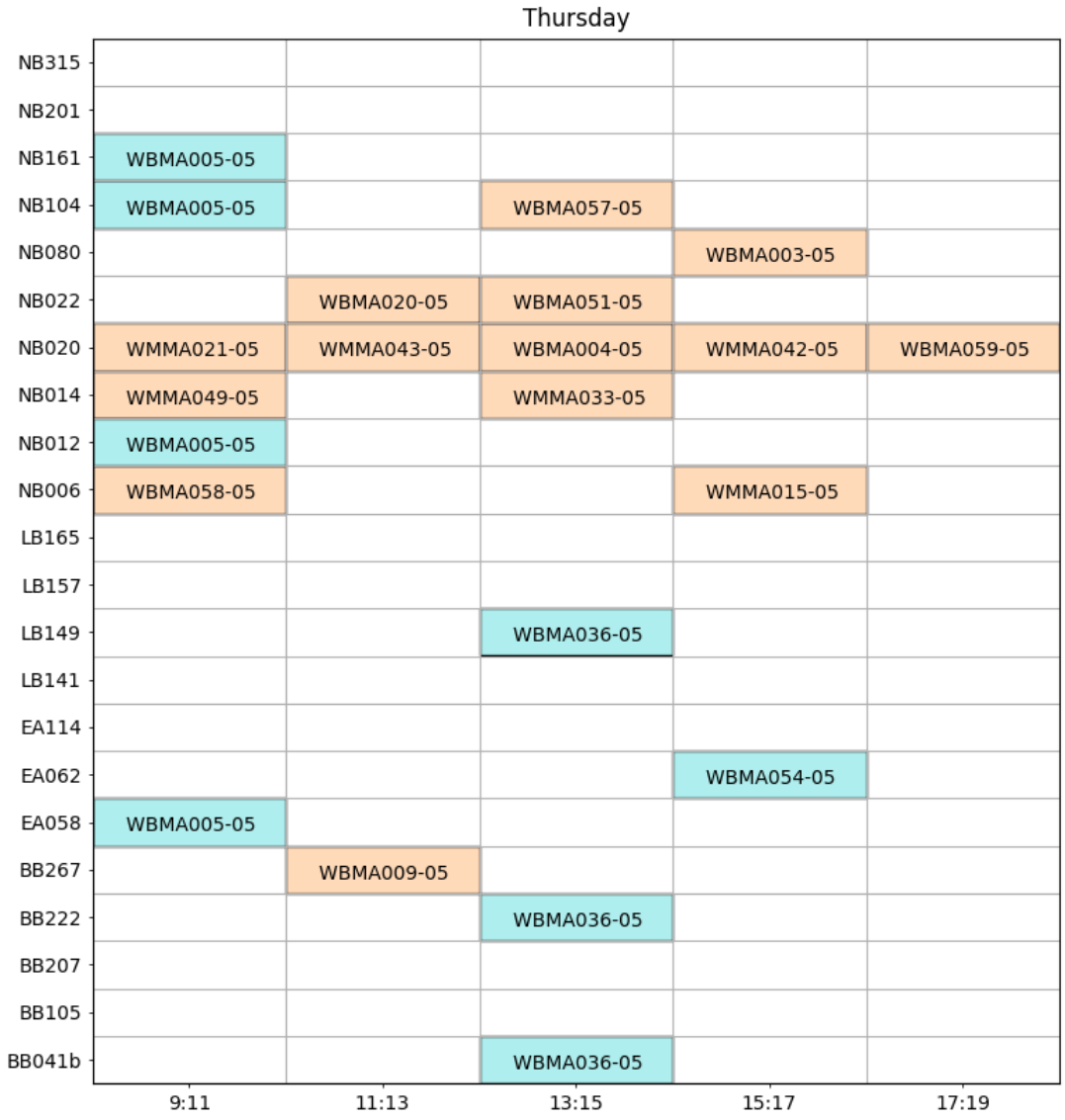

Figure A.4.: Block 1A - Model 1

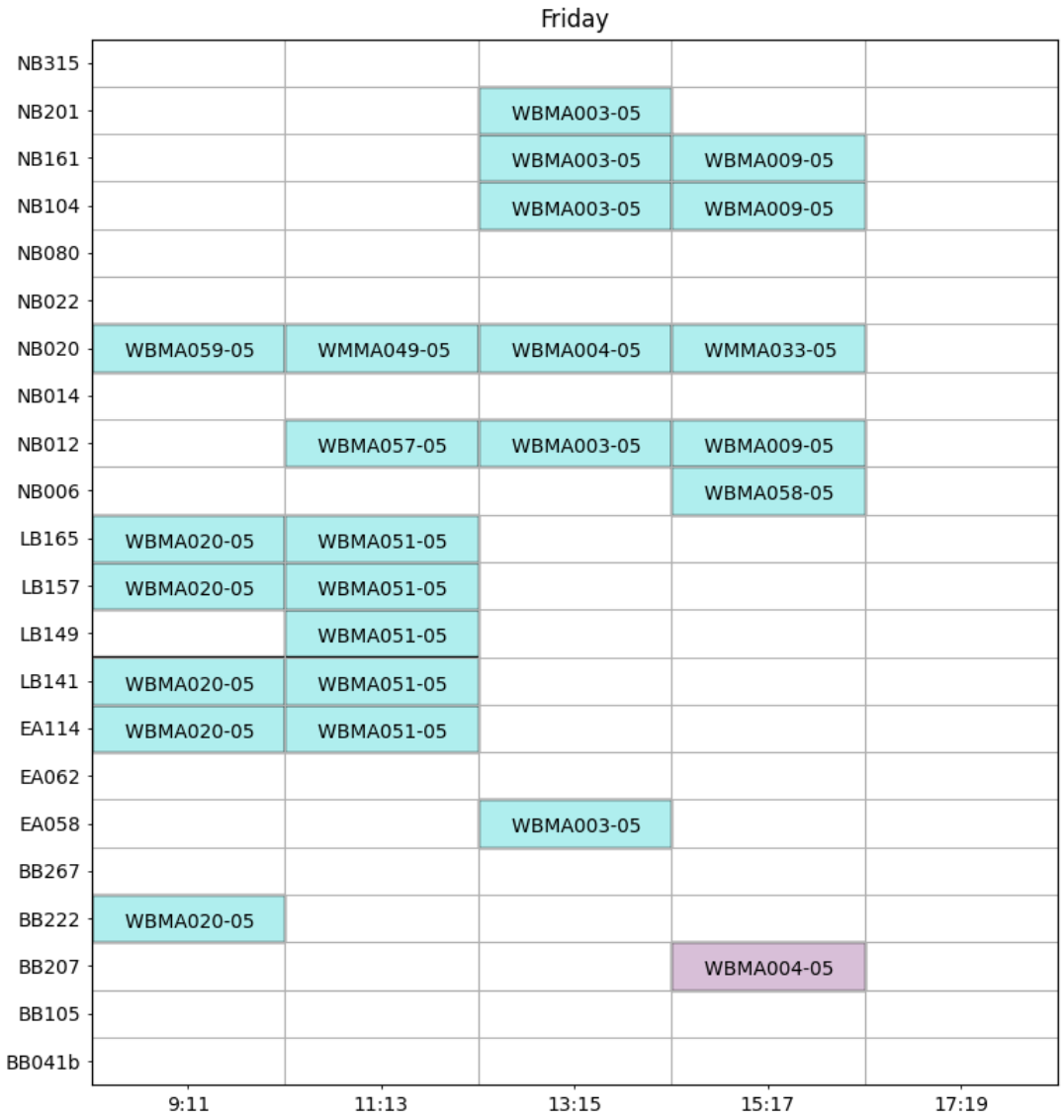

Figure A.5.: Block 1A - Model 1

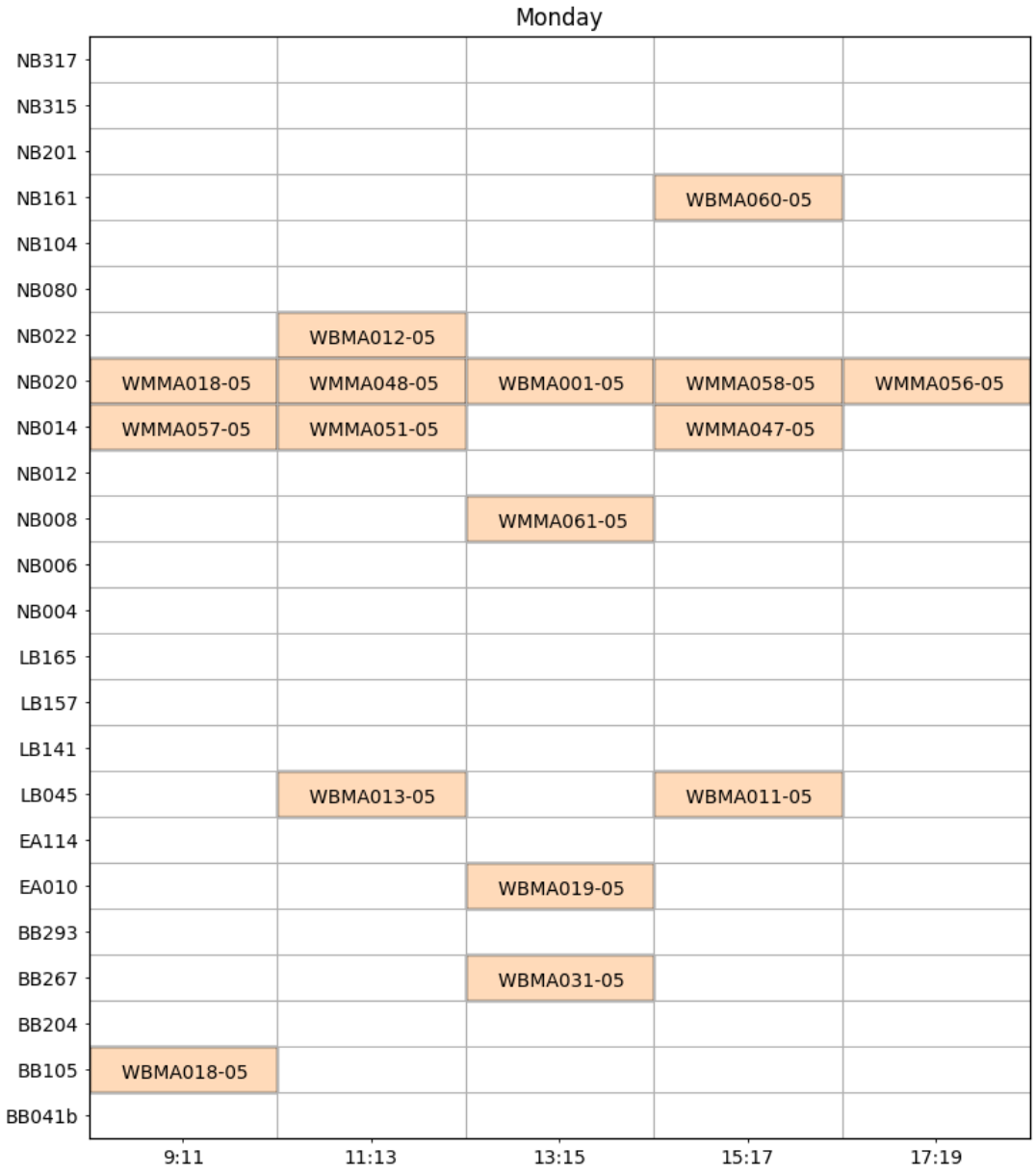

Figure A.6.: Block 1B - Model 1

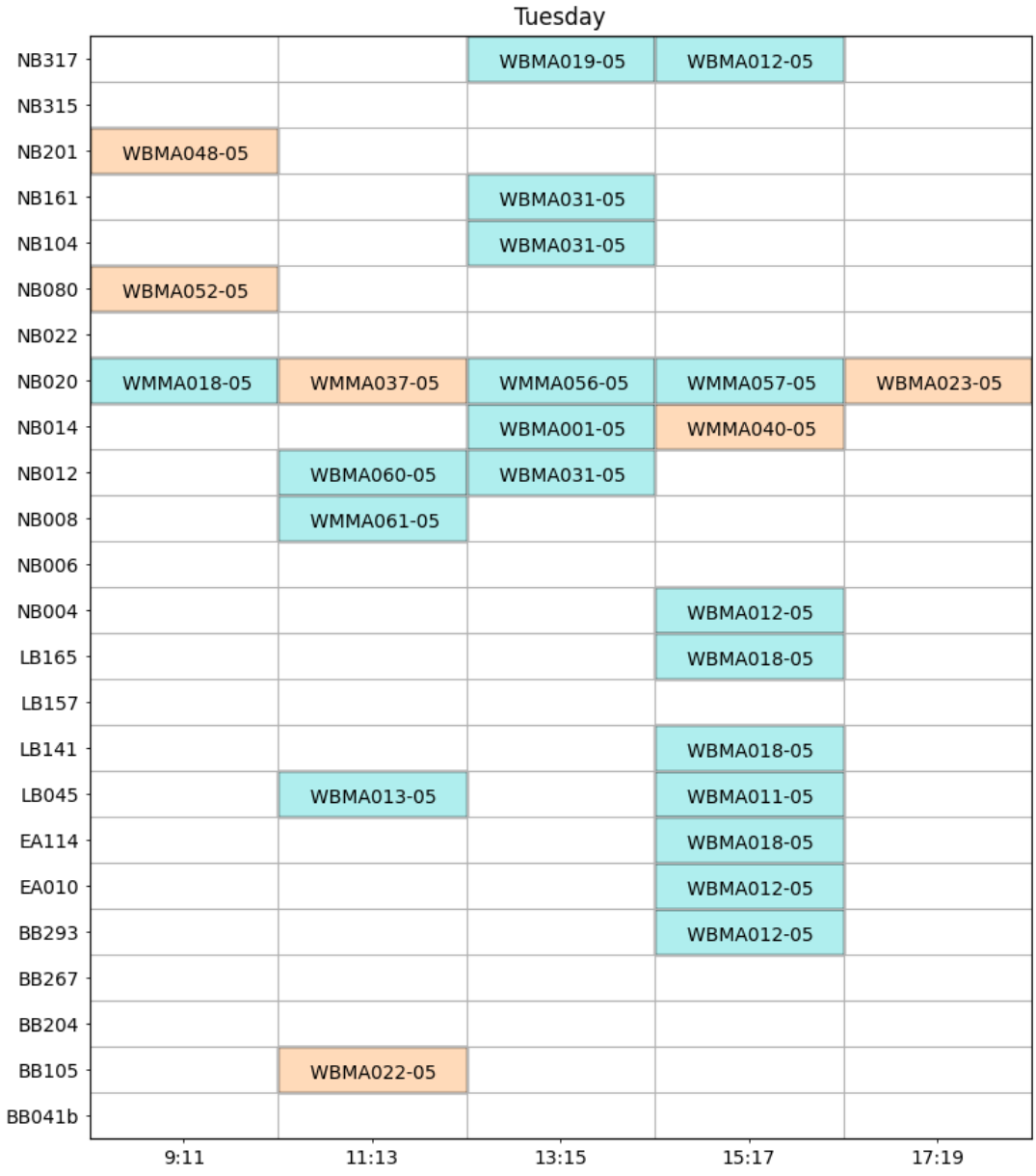

Figure A.7.: Block 1B - Model 1

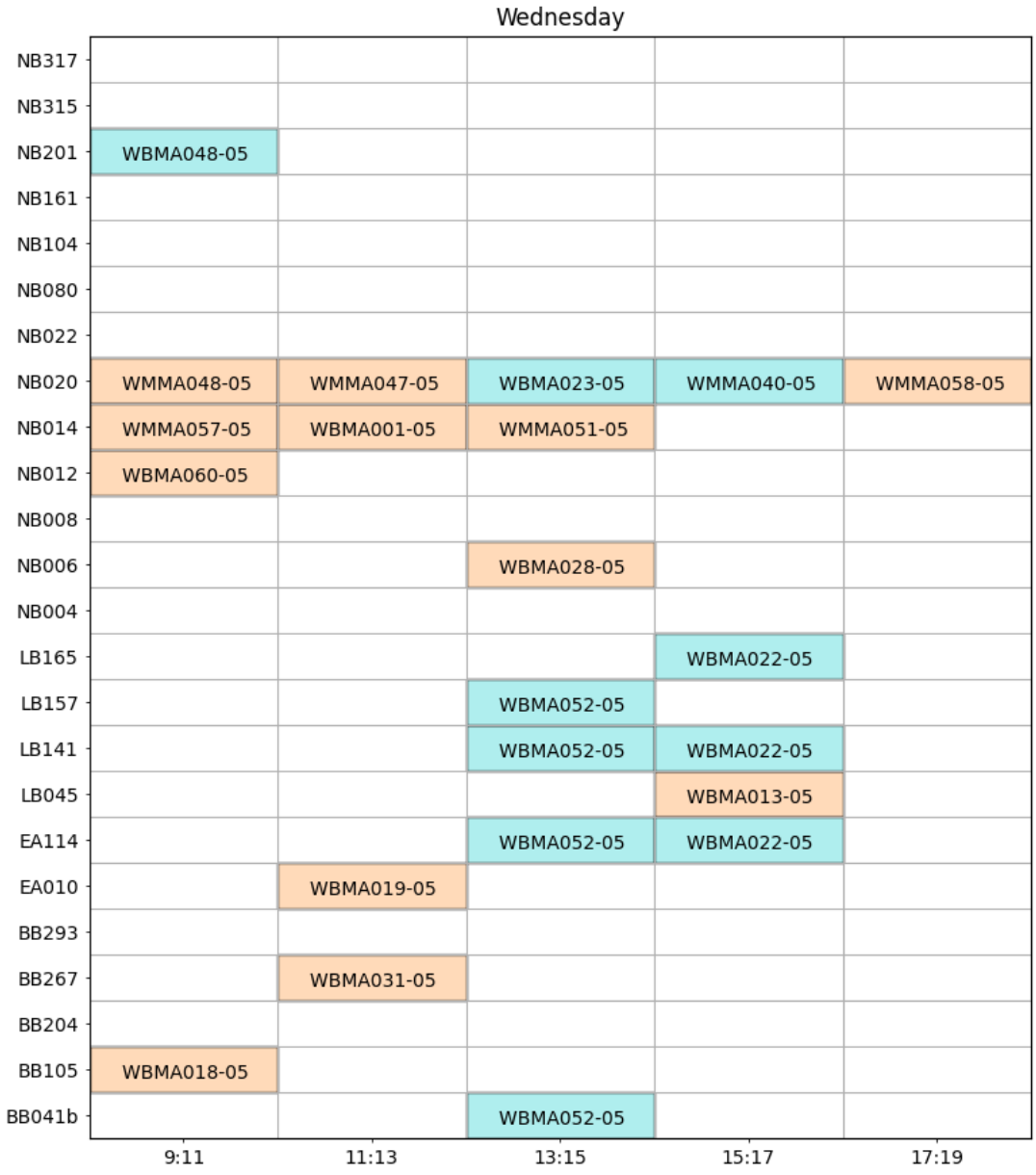

Figure A.8.: Block 1B - Model 1

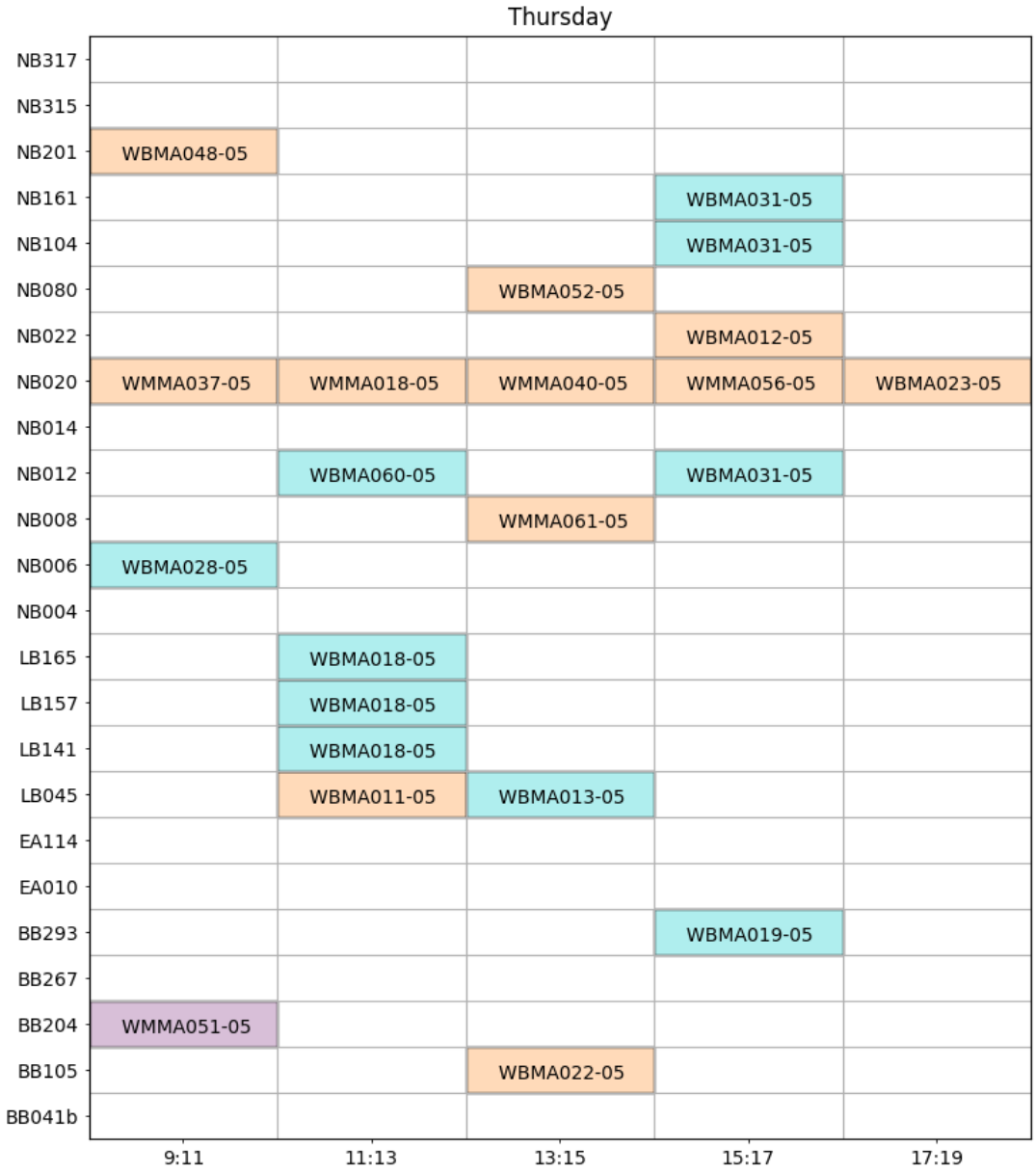

Figure A.9.: Block 1B - Model 1

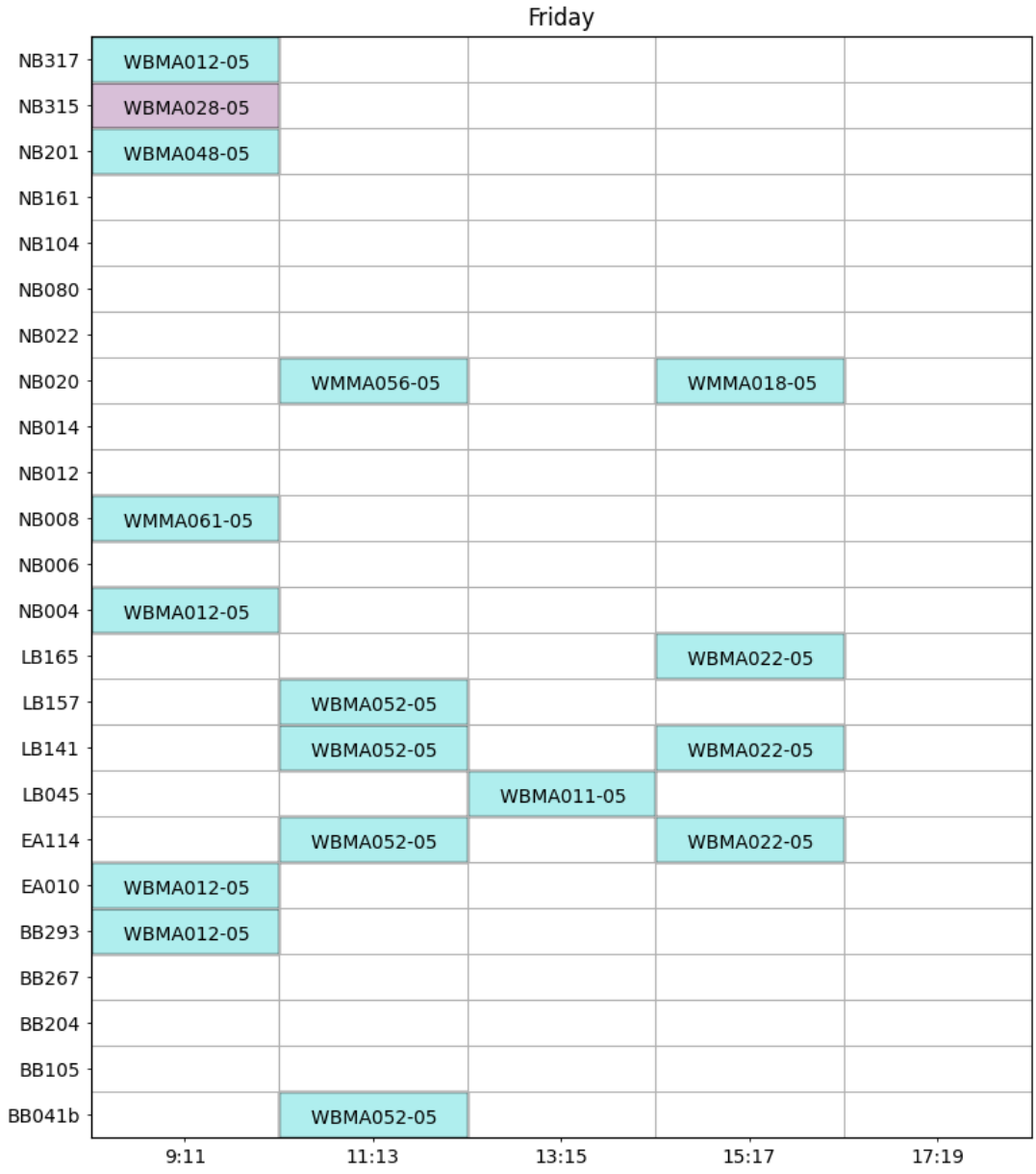

Figure A.10.: Block 1B - Model 1

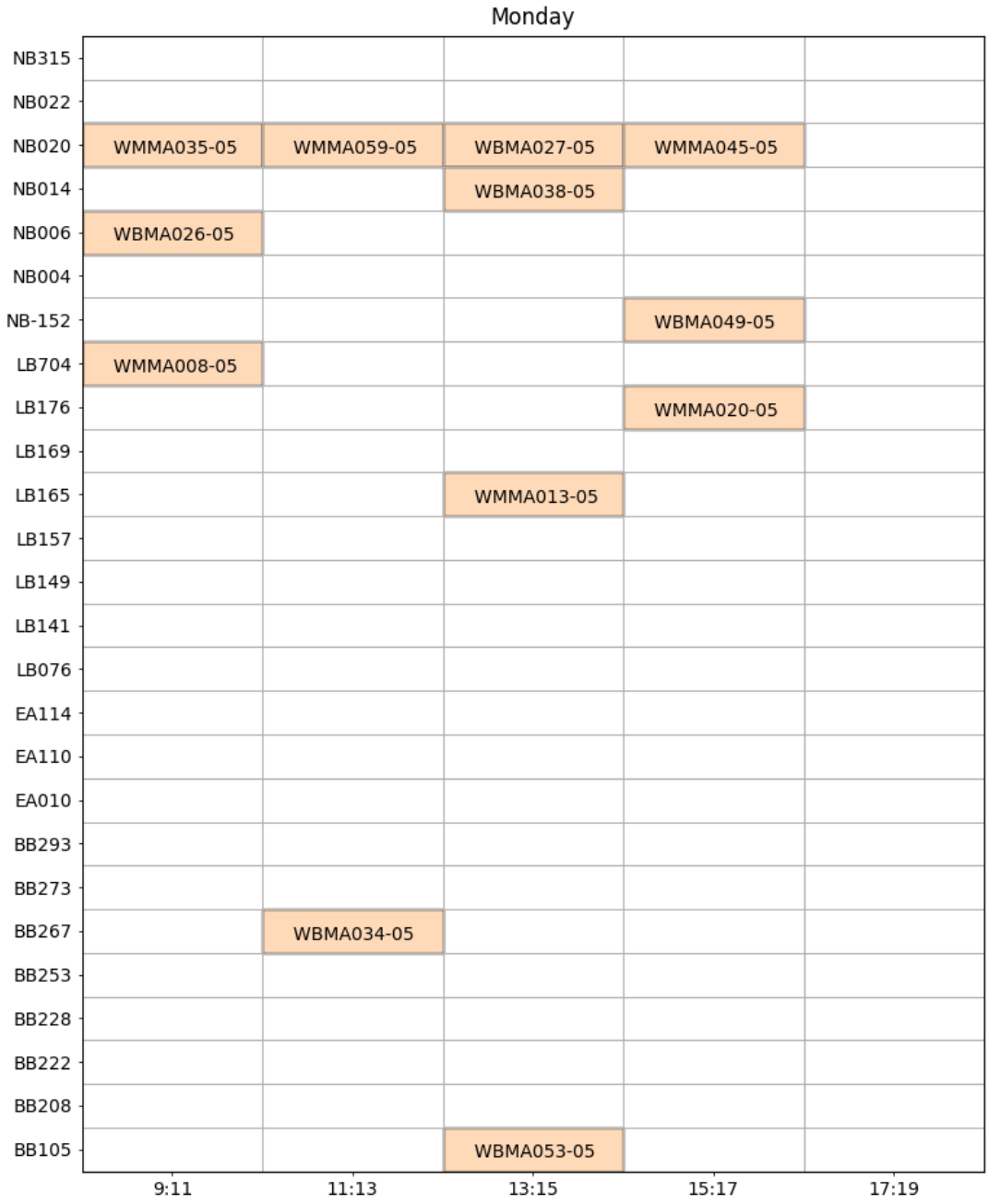

Figure A.11.: Block 2A - Model 1

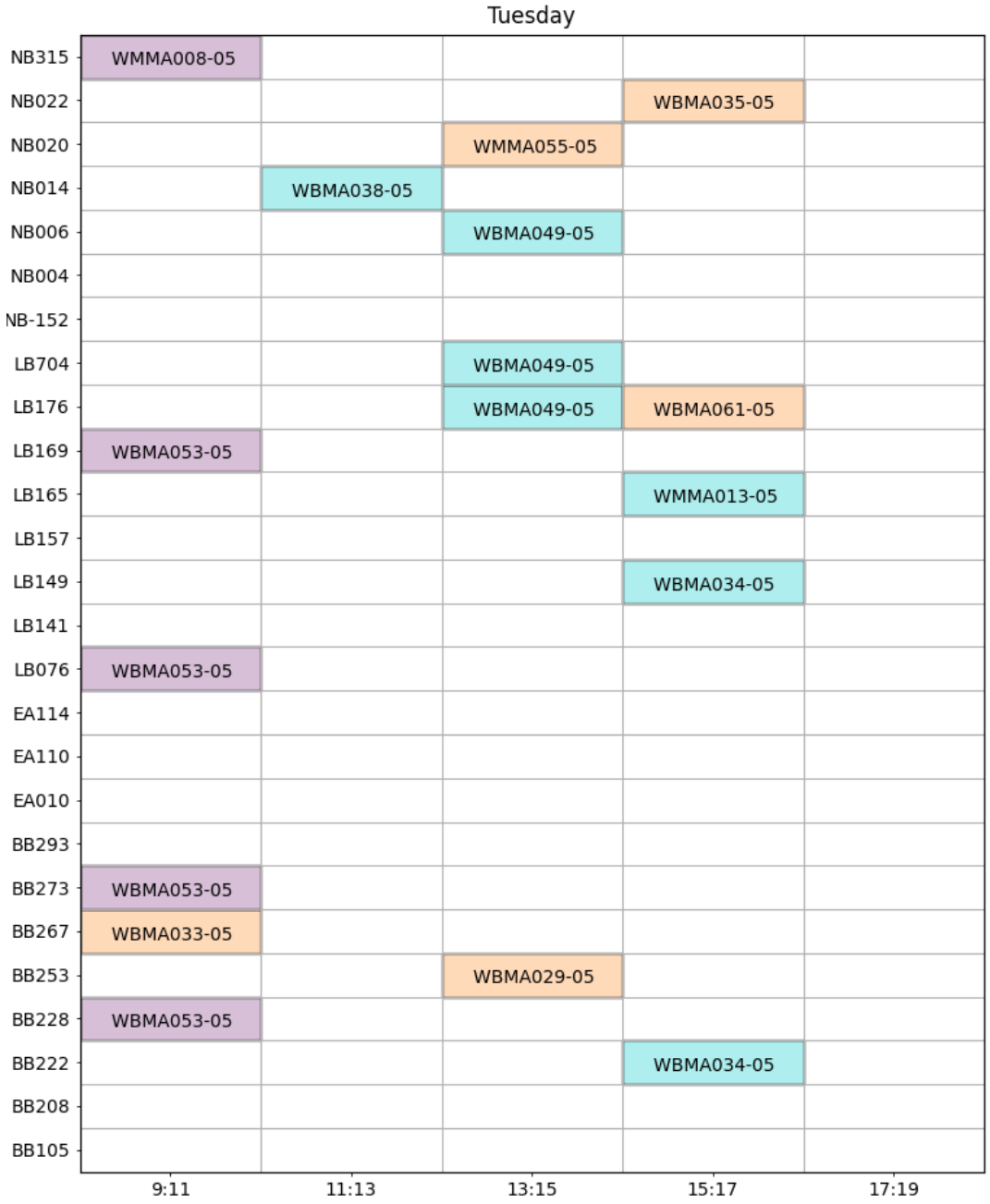

Figure A.12.: Block 2A - Model 1

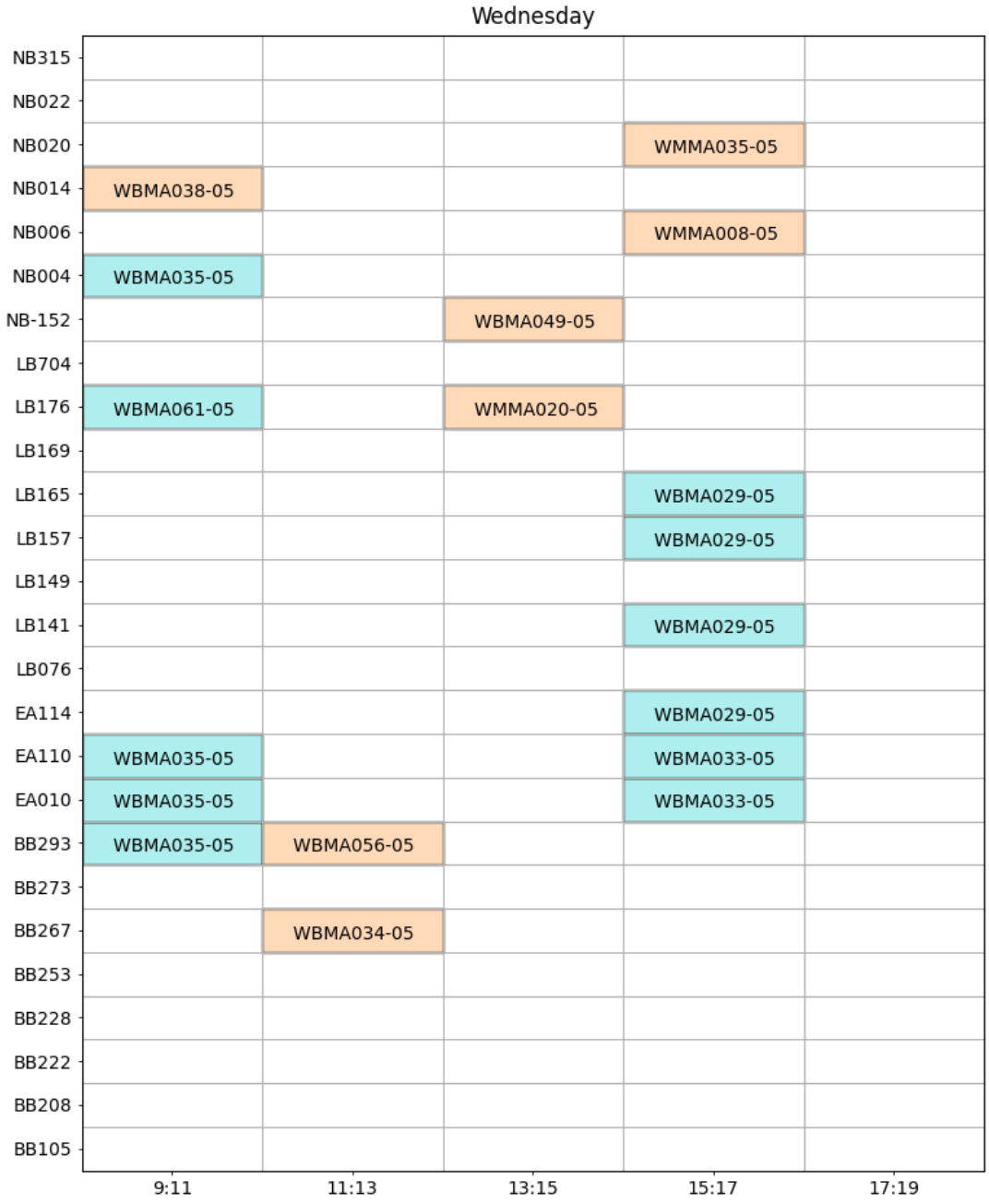

Figure A.13.: Block 2A - Model 1

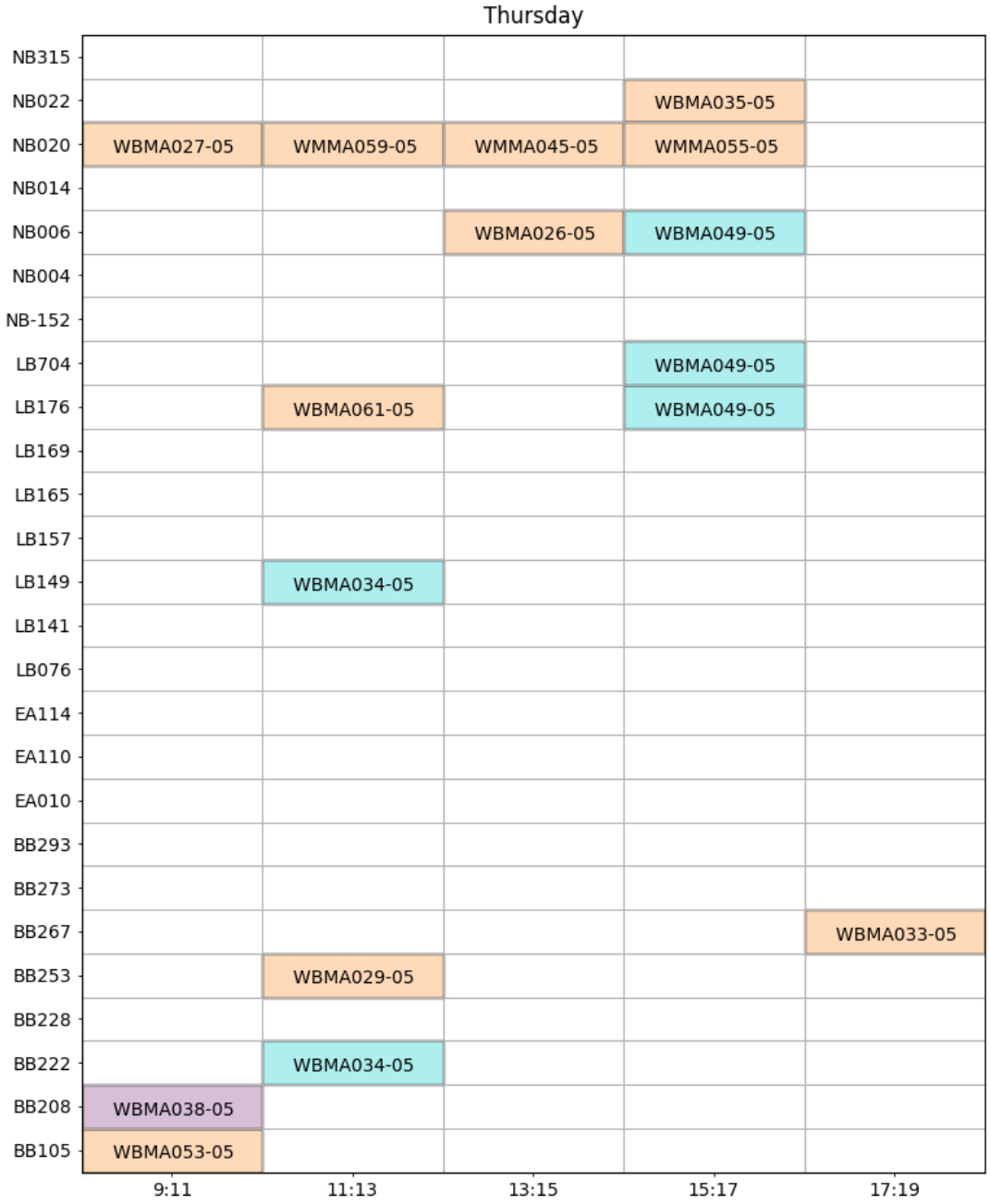

Figure A.14.: Block 2A - Model 1

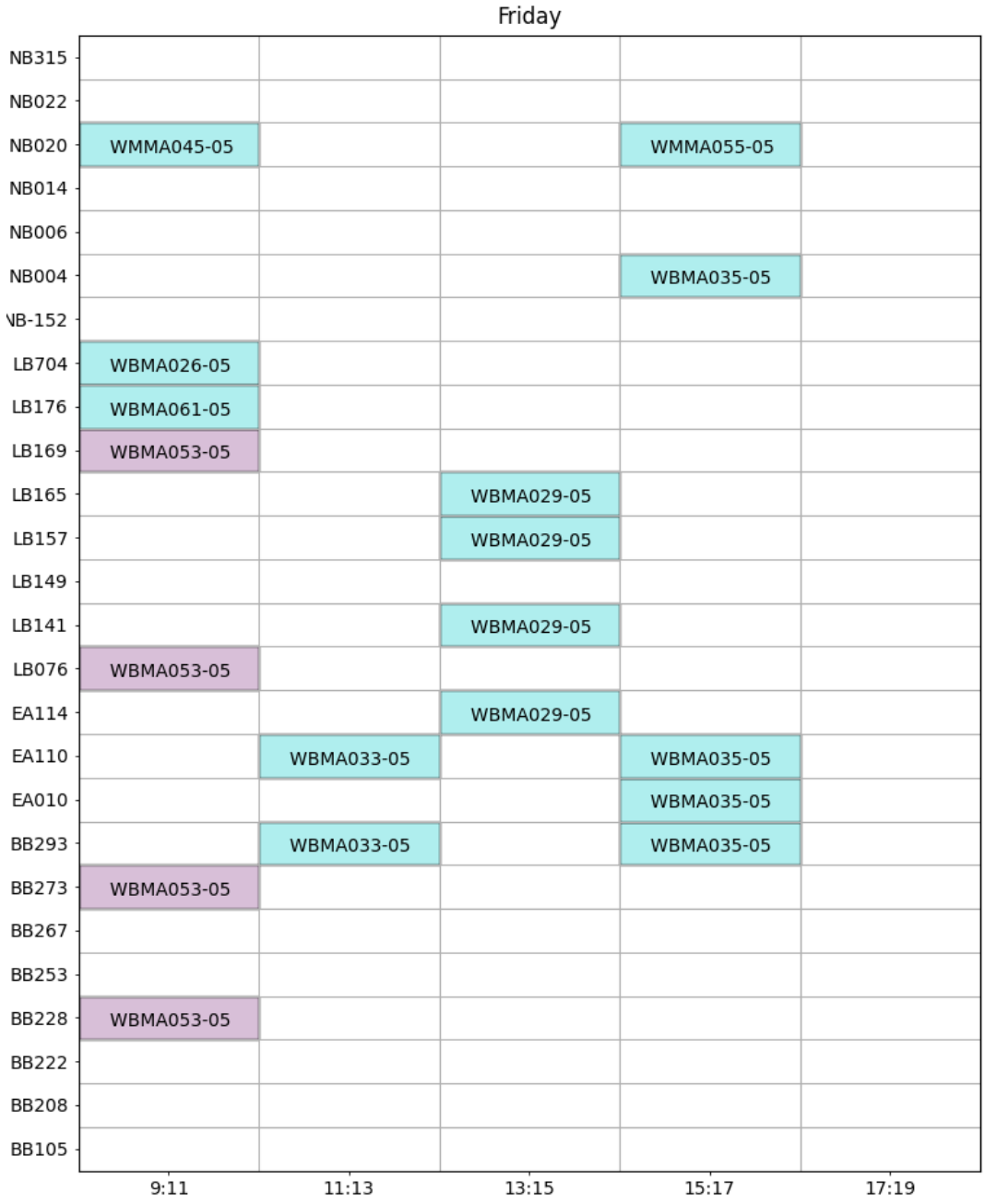

Figure A.15.: Block 2A - Model 1

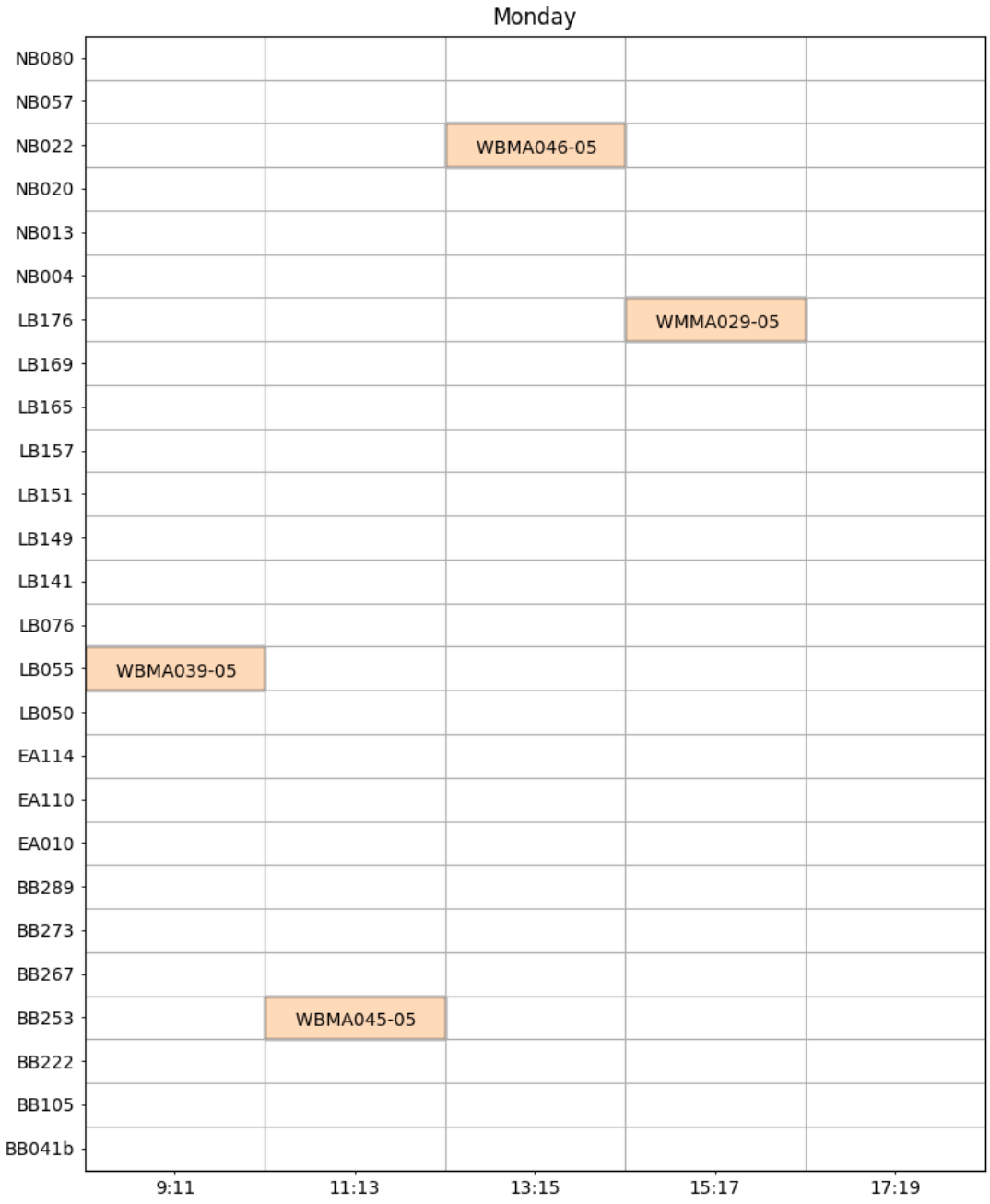

Figure A.16.: Block 2B - Model 1

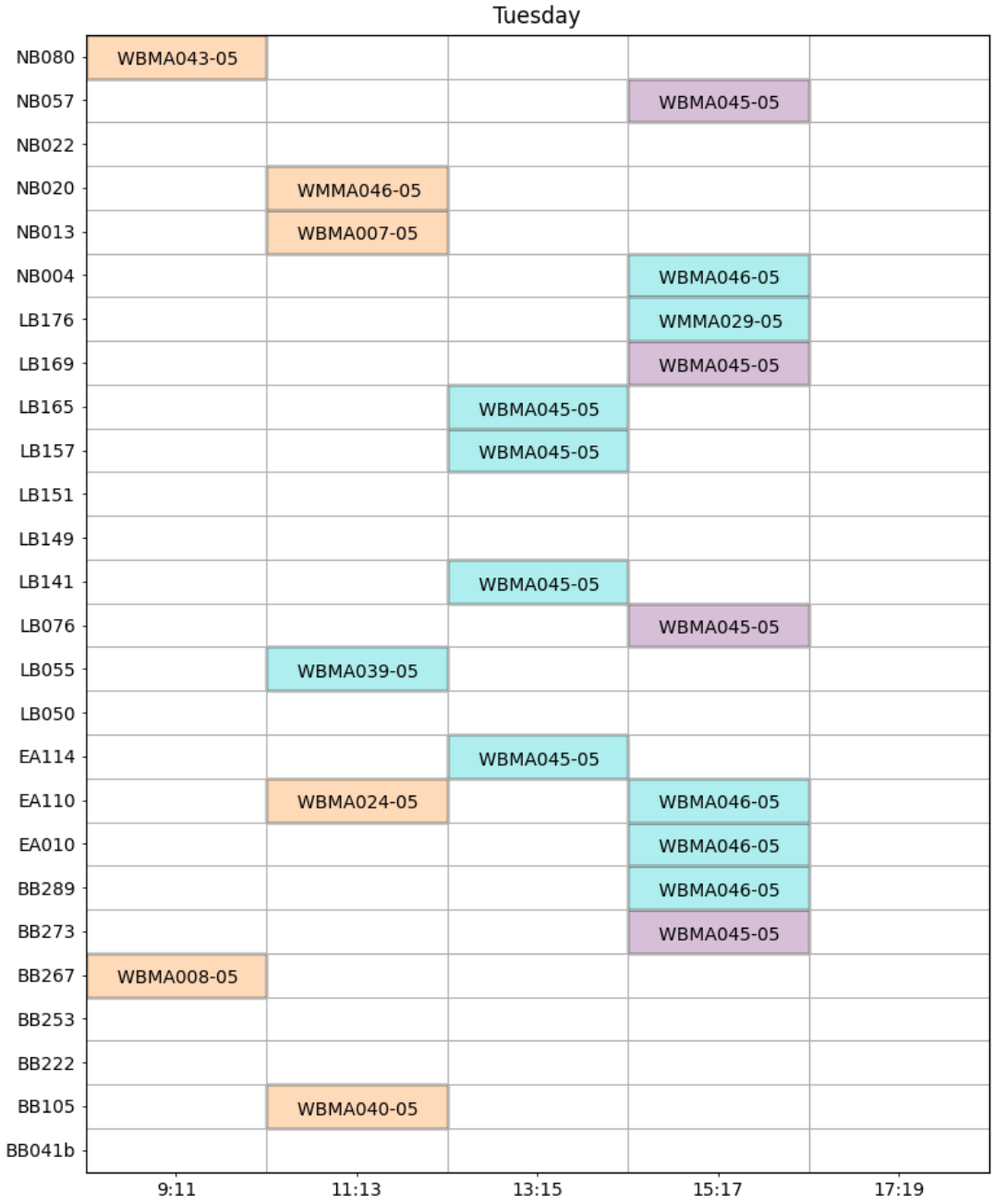

Figure A.17.: Block 2B - Model 1

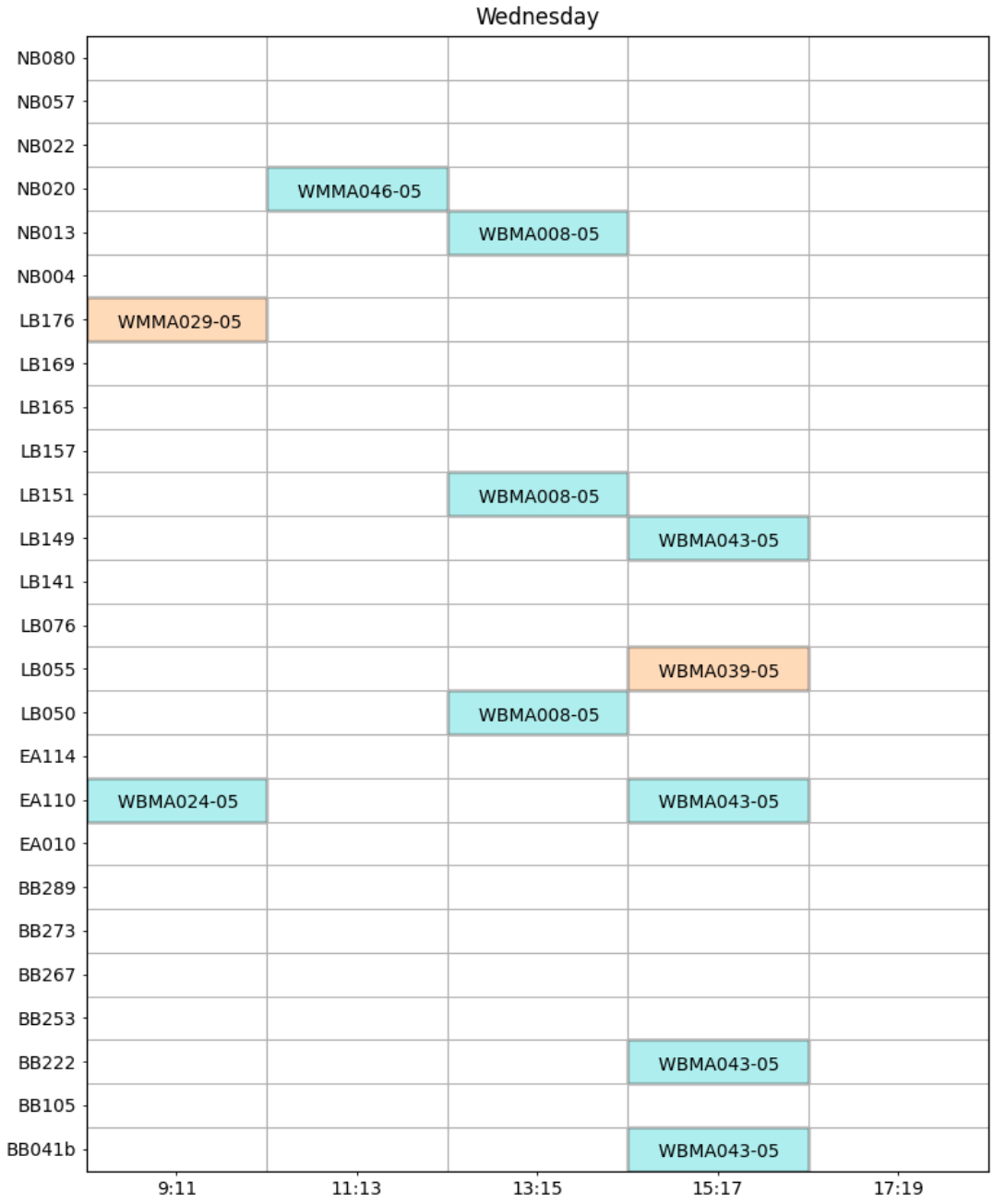

Figure A.18.: Block 2B - Model 1

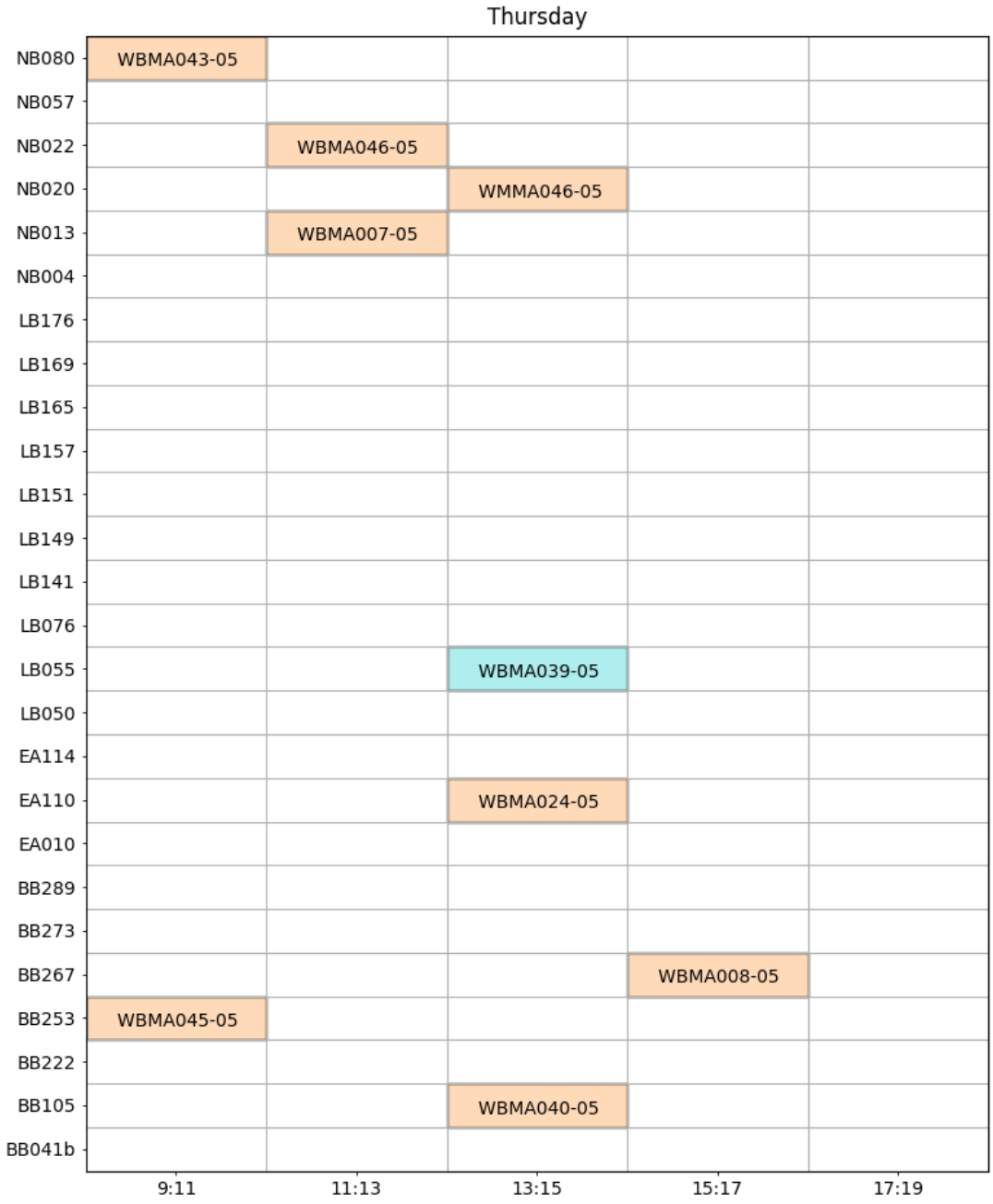

Figure A.19.: Block 2B - Model 1

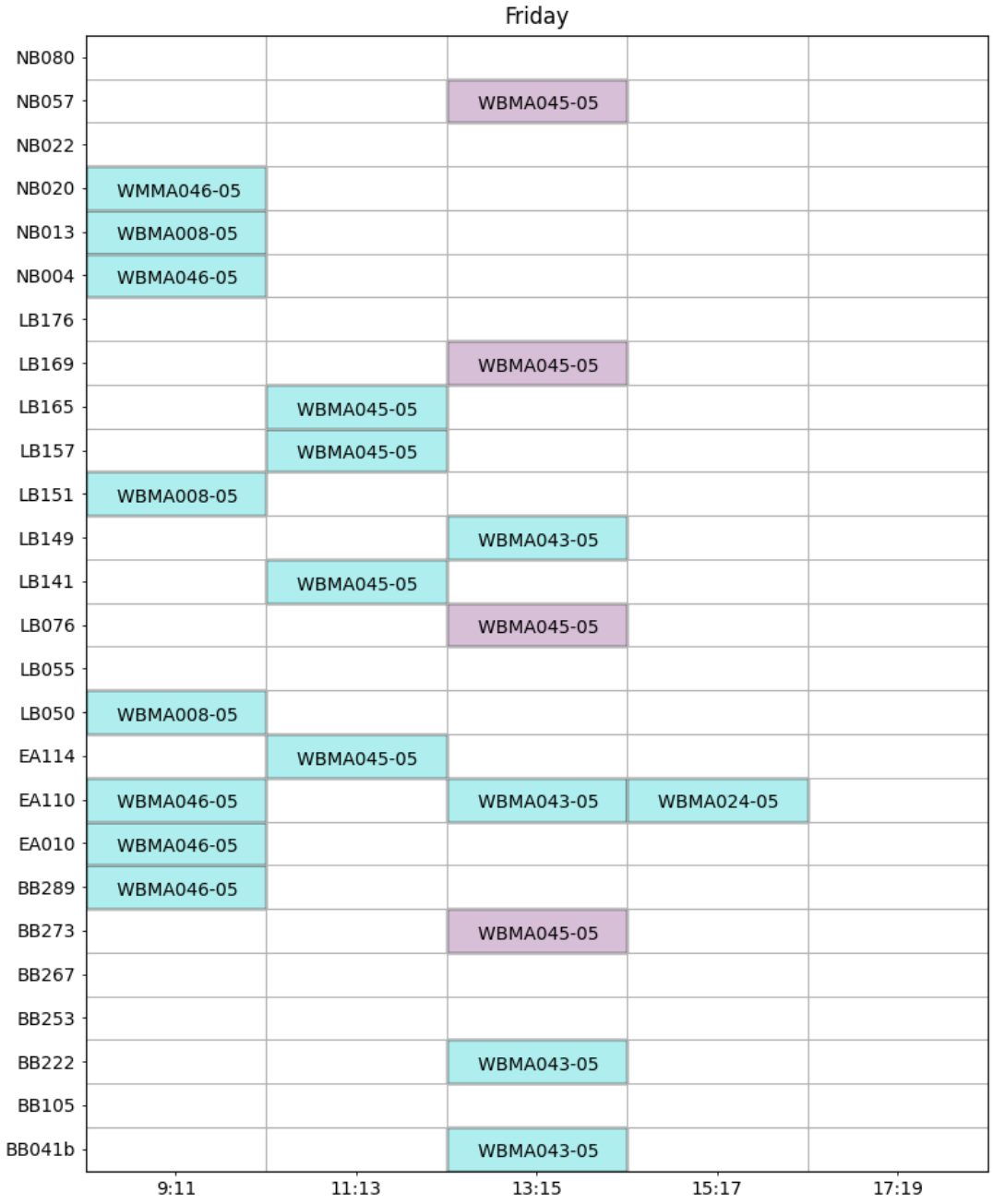

Figure A.20.: Block 2B - Model 1

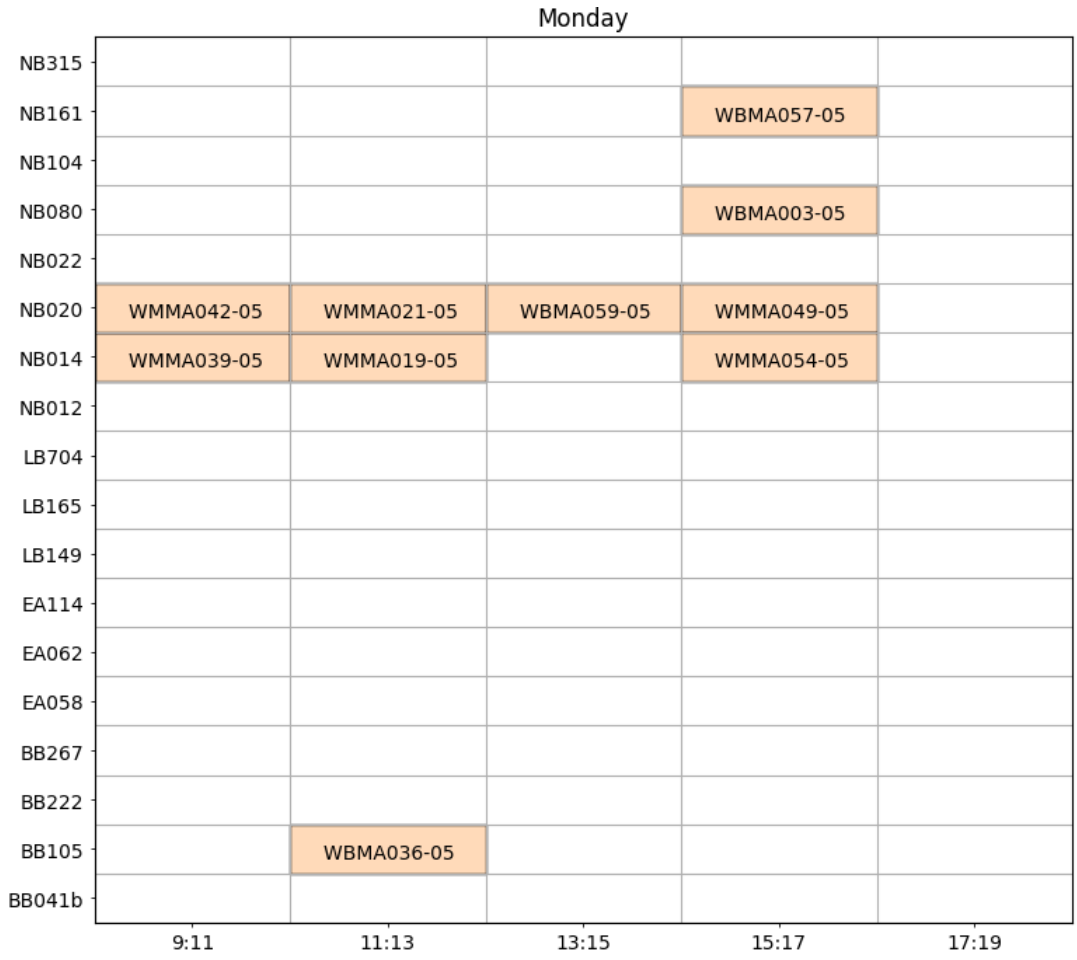

Figure A.21.: Block 1A - Model 2

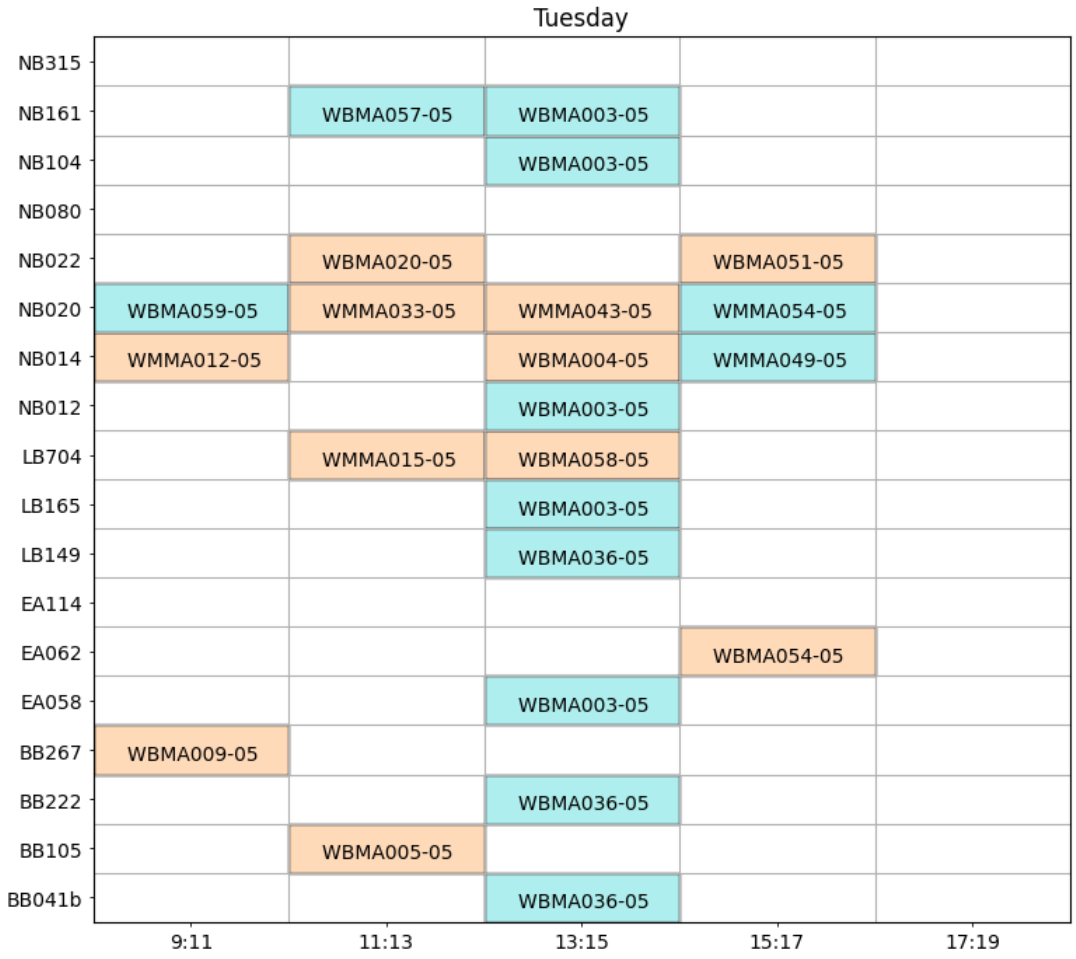

Figure A.22.: Block 1A - Model 2

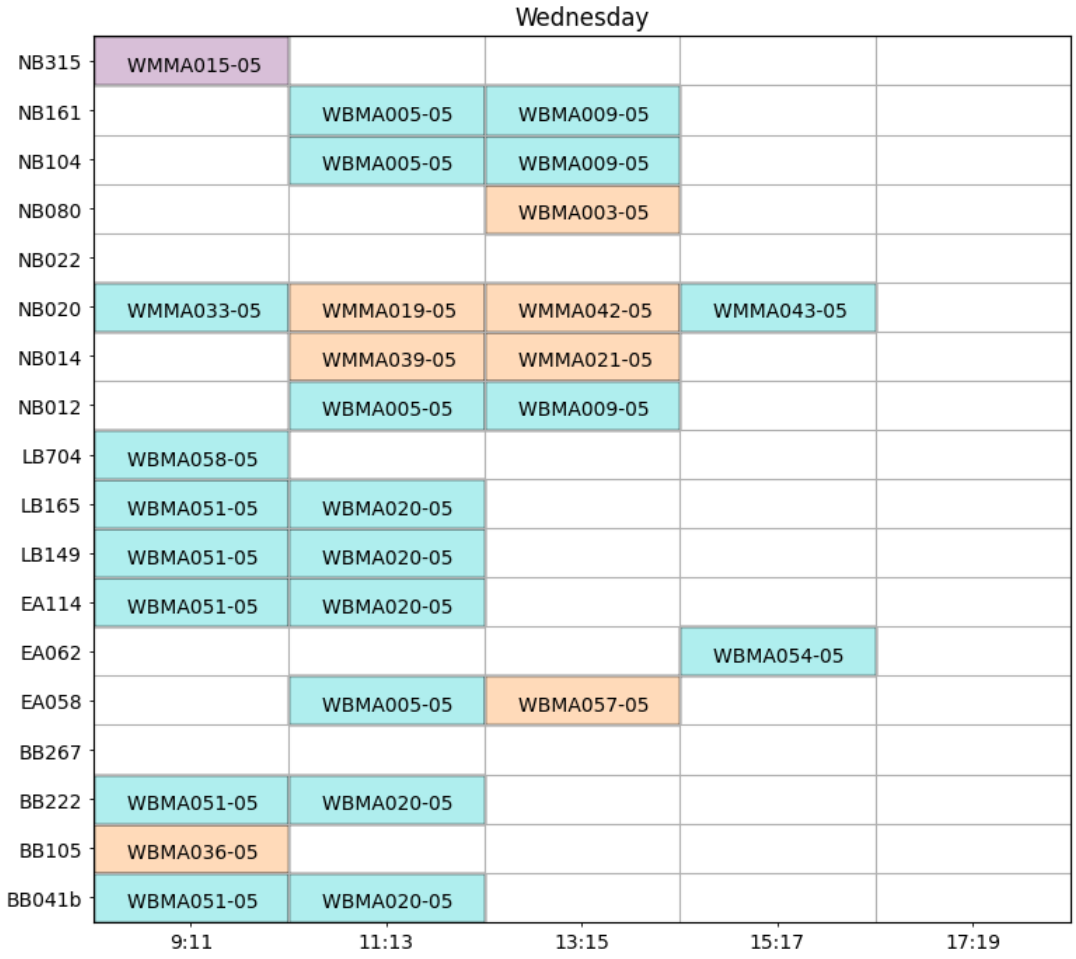

Figure A.23.: Block 1A - Model 2

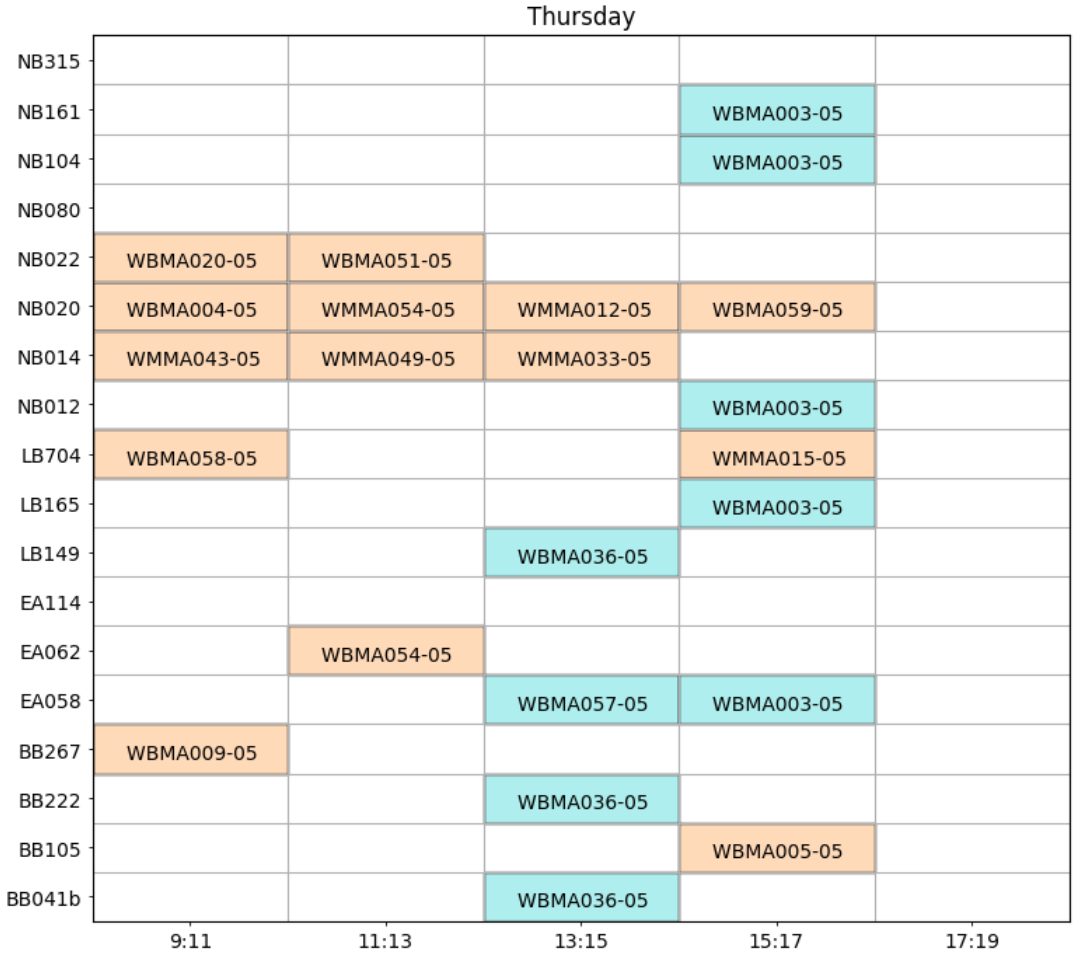

Figure A.24.: Block 1A - Model 2

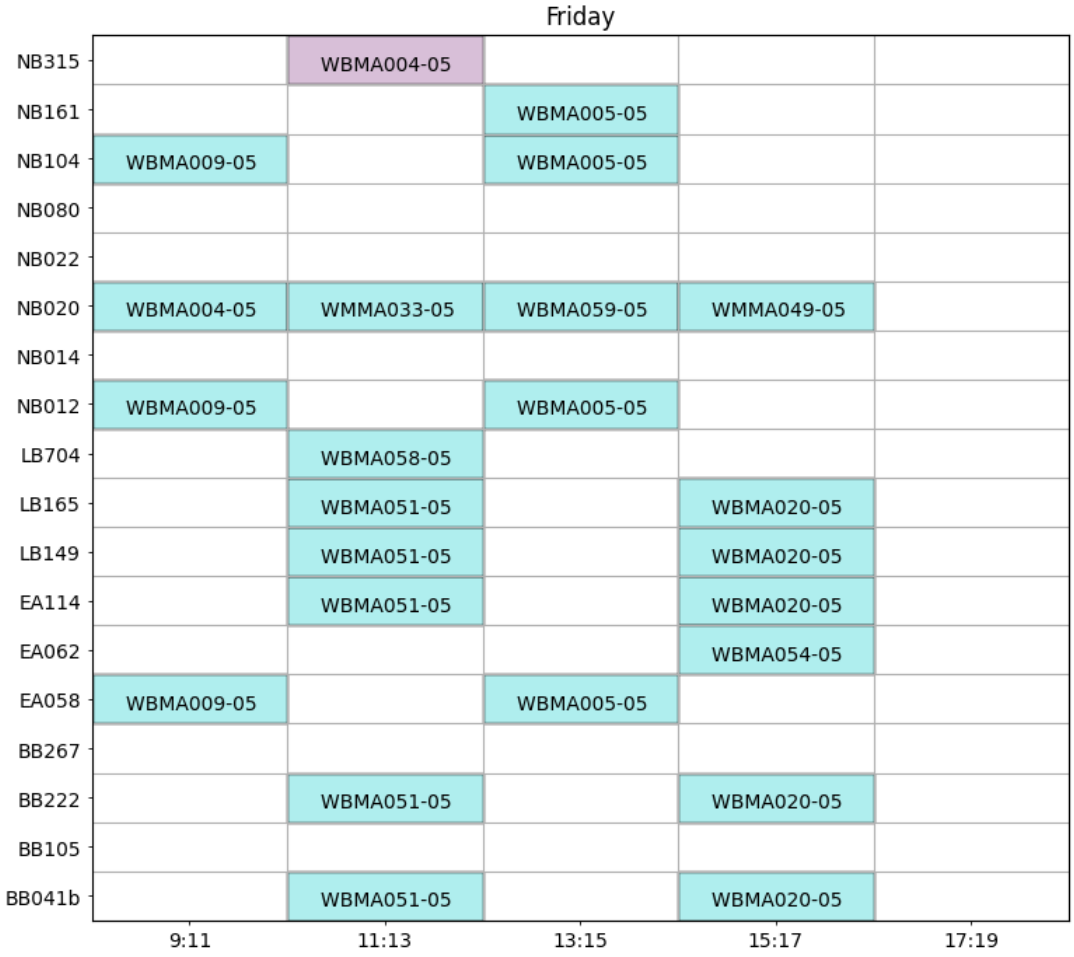

Figure A.25.: Block 1A - Model 2

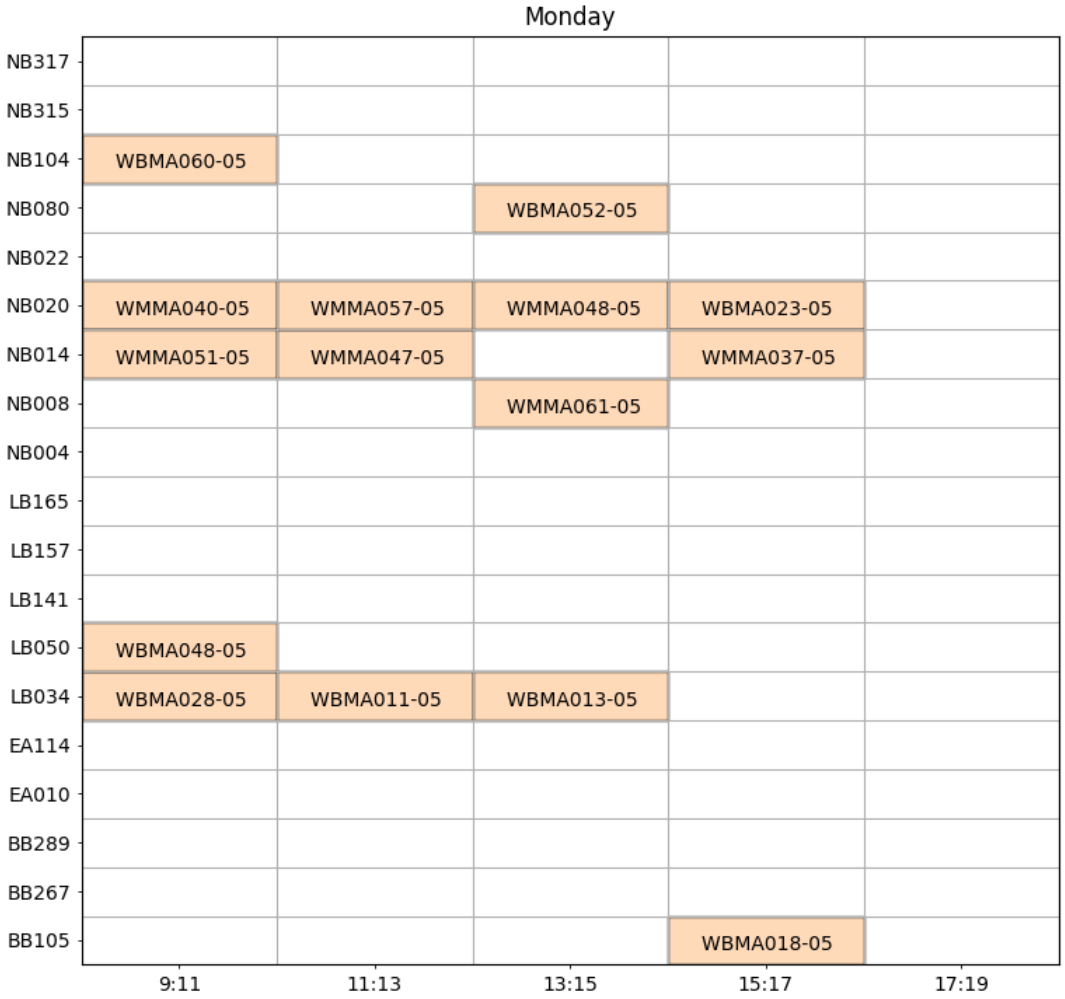

Figure A.26.: Block 1B - Model 2

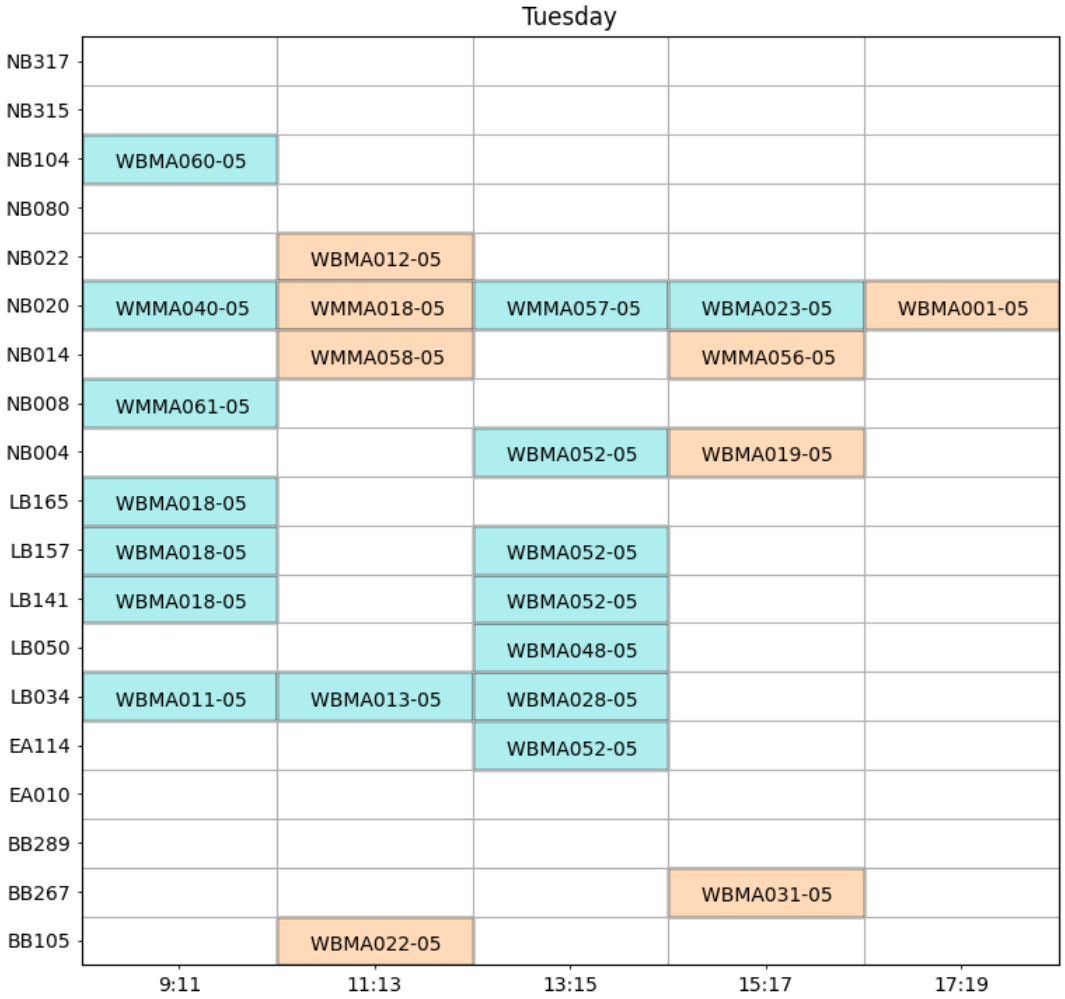

Figure A.27.: Block 1B - Model 2

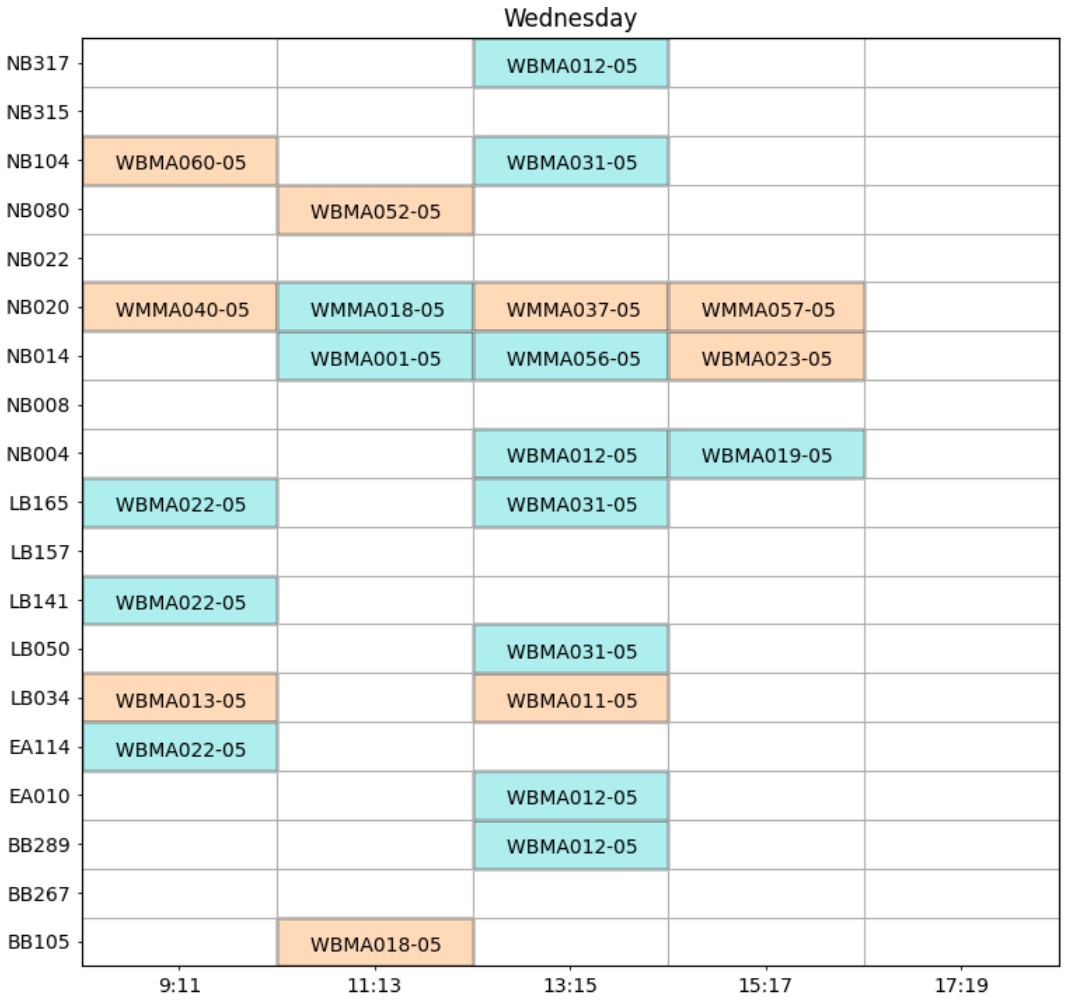

Figure A.28.: Block 1B - Model 2

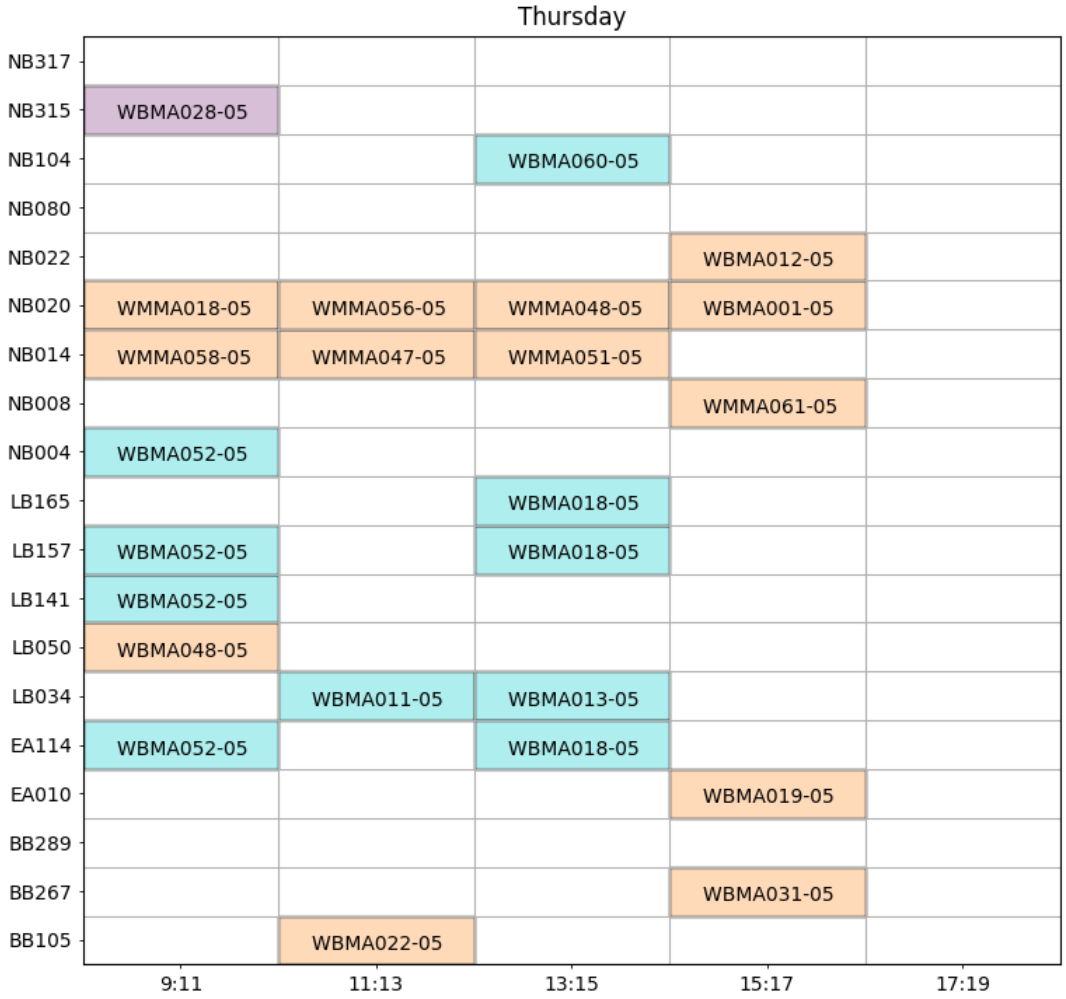

Figure A.29.: Block 1B - Model 2

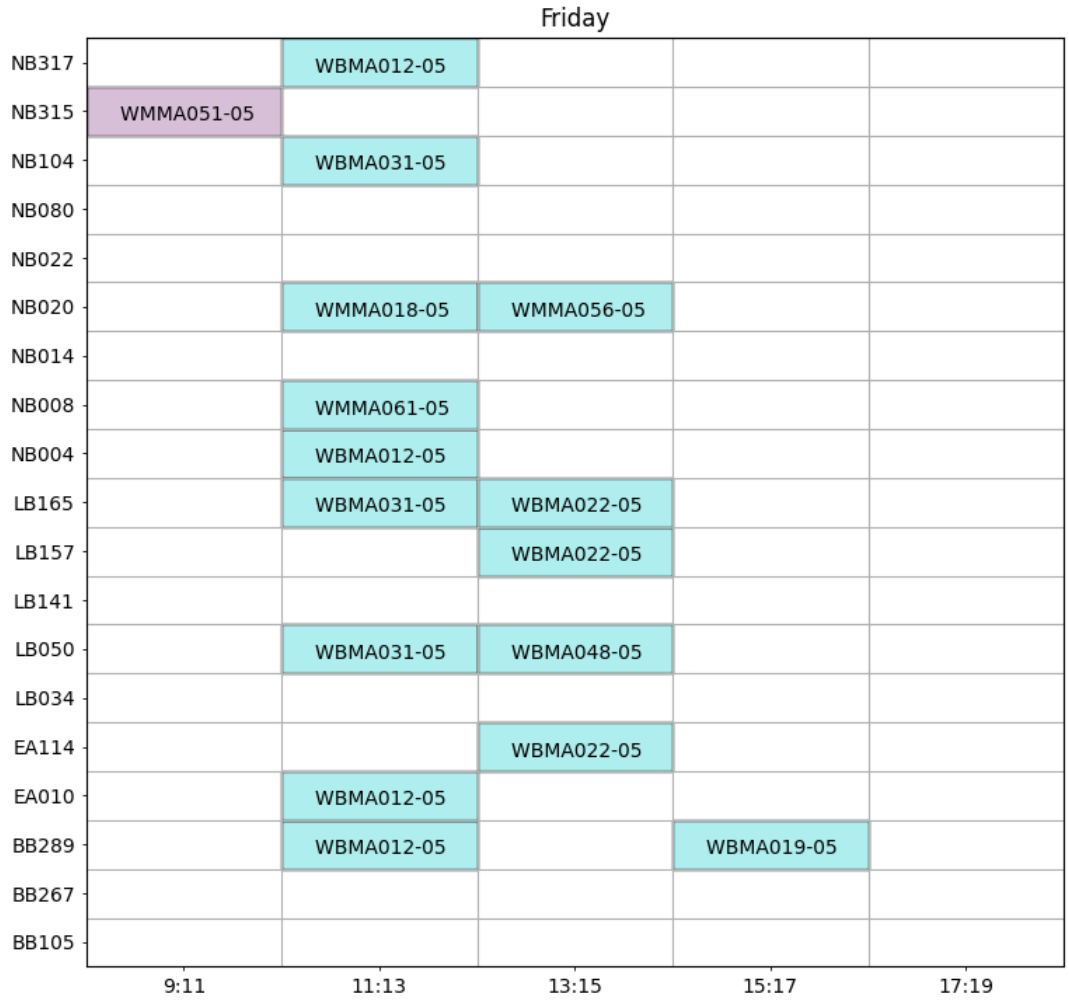

Figure A.30.: Block 1B - Model 2

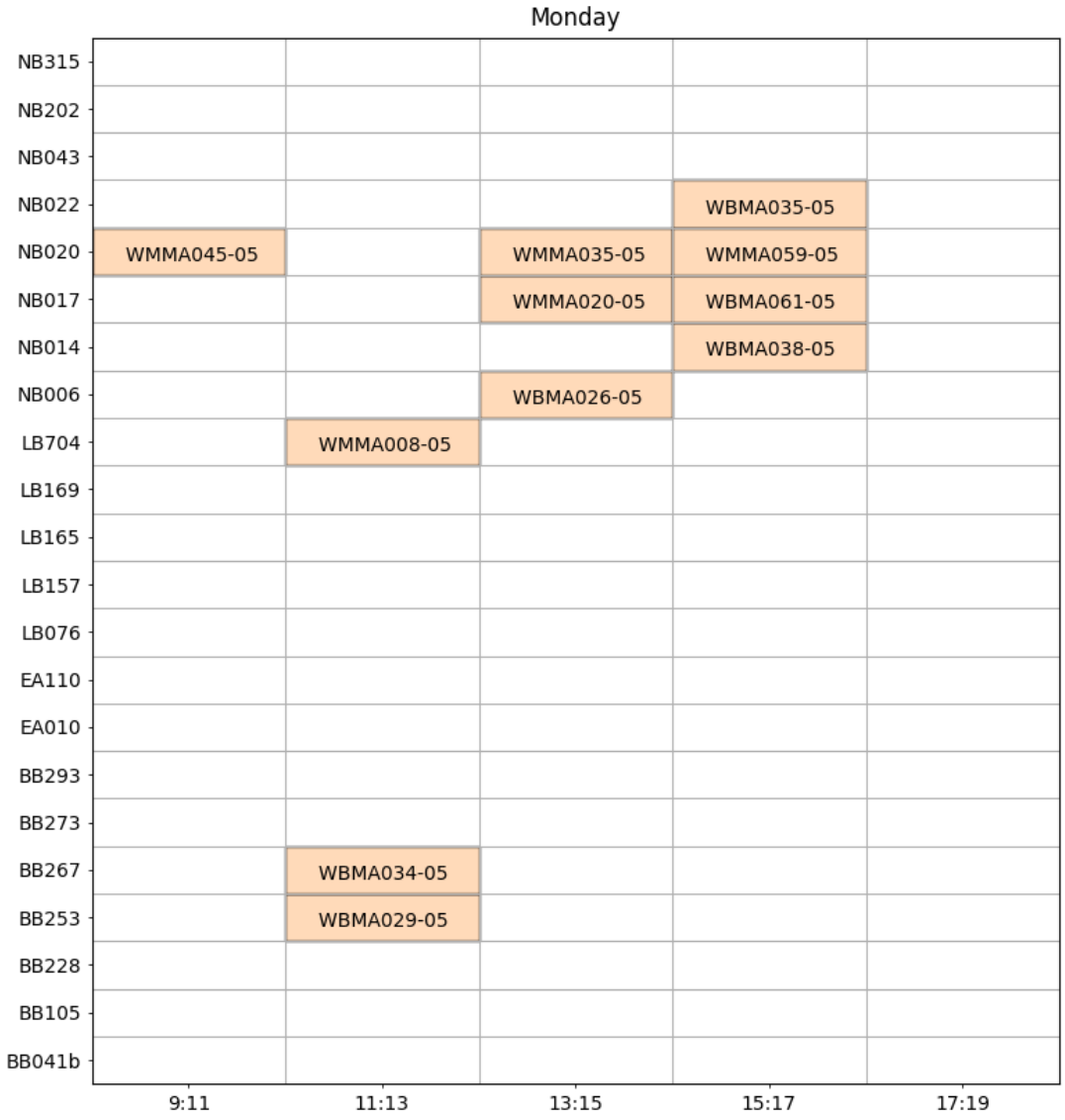

Figure A.31.: Block 2A - Model 2

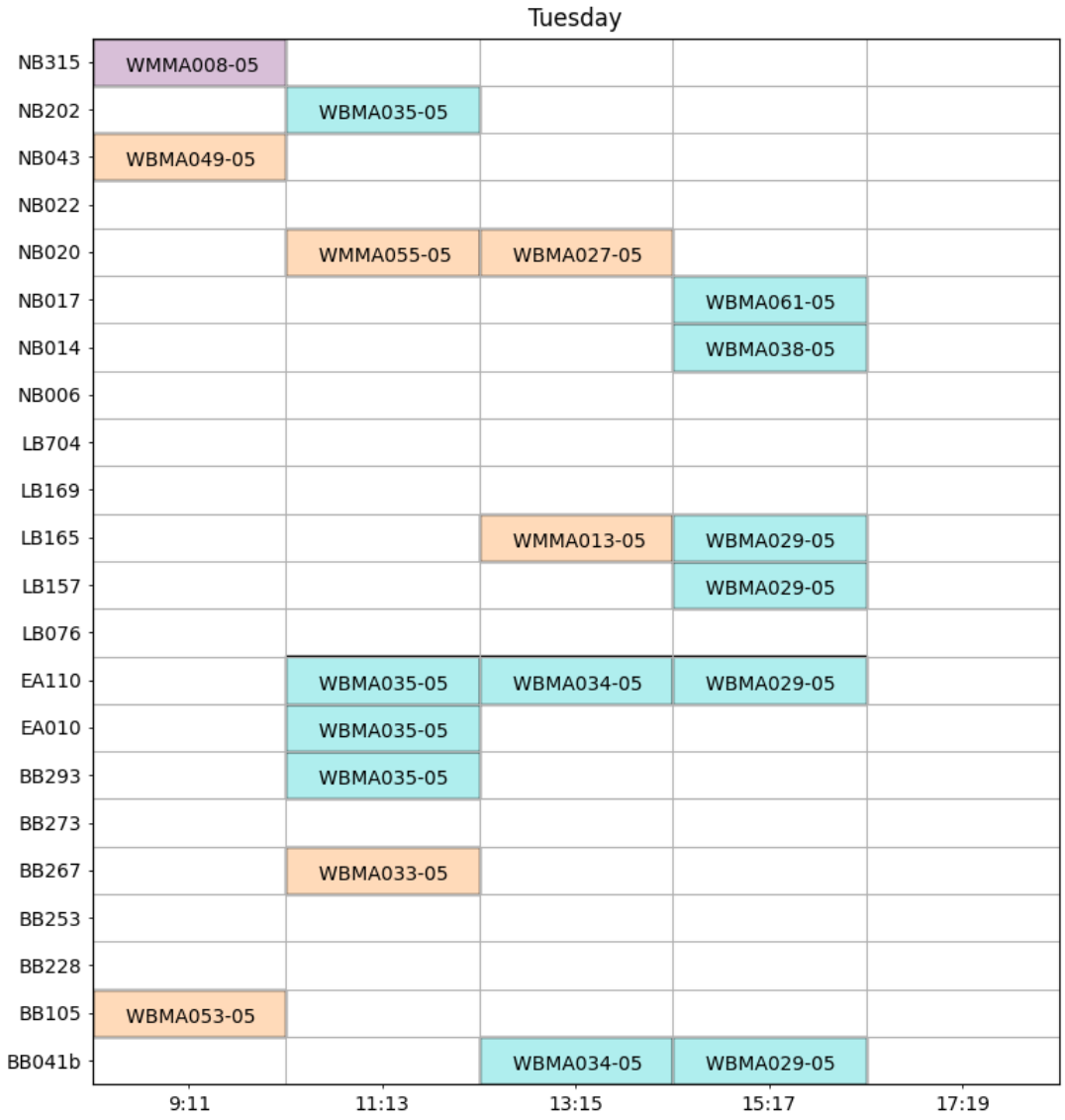

Figure A.32.: Block 2A - Model 2

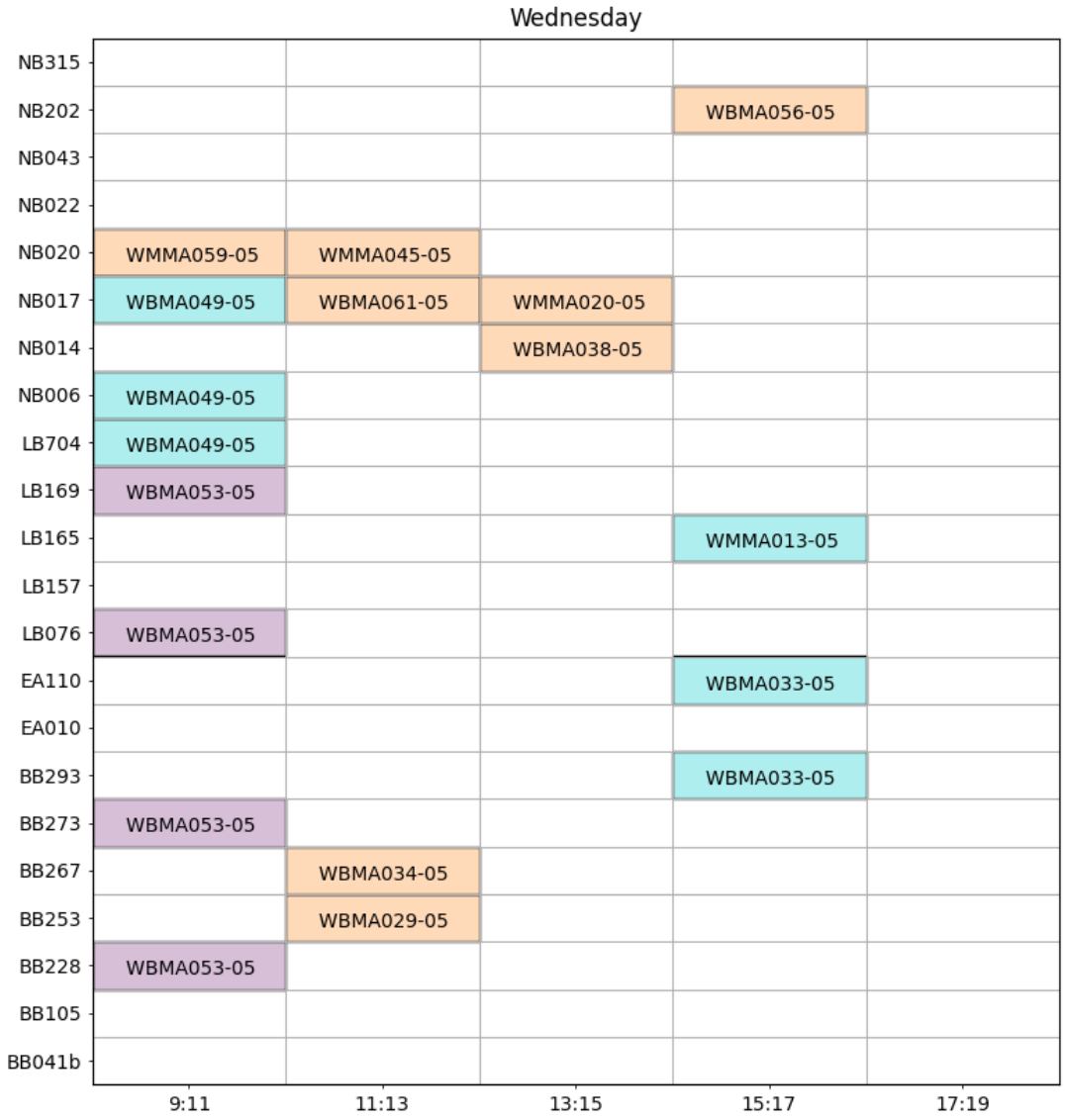

Figure A.33.: Block 2A - Model 2

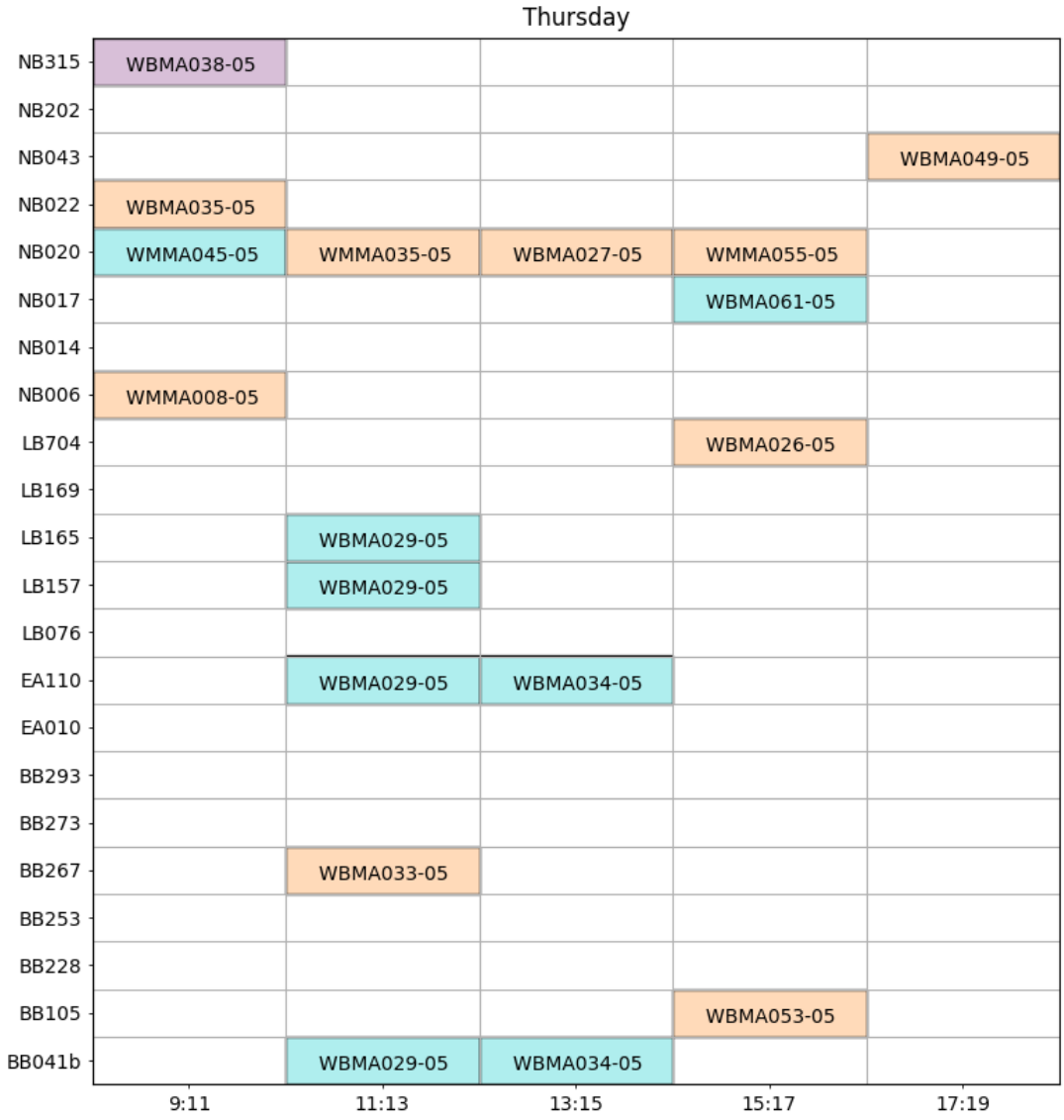

Figure A.34.: Block 2A - Model 2

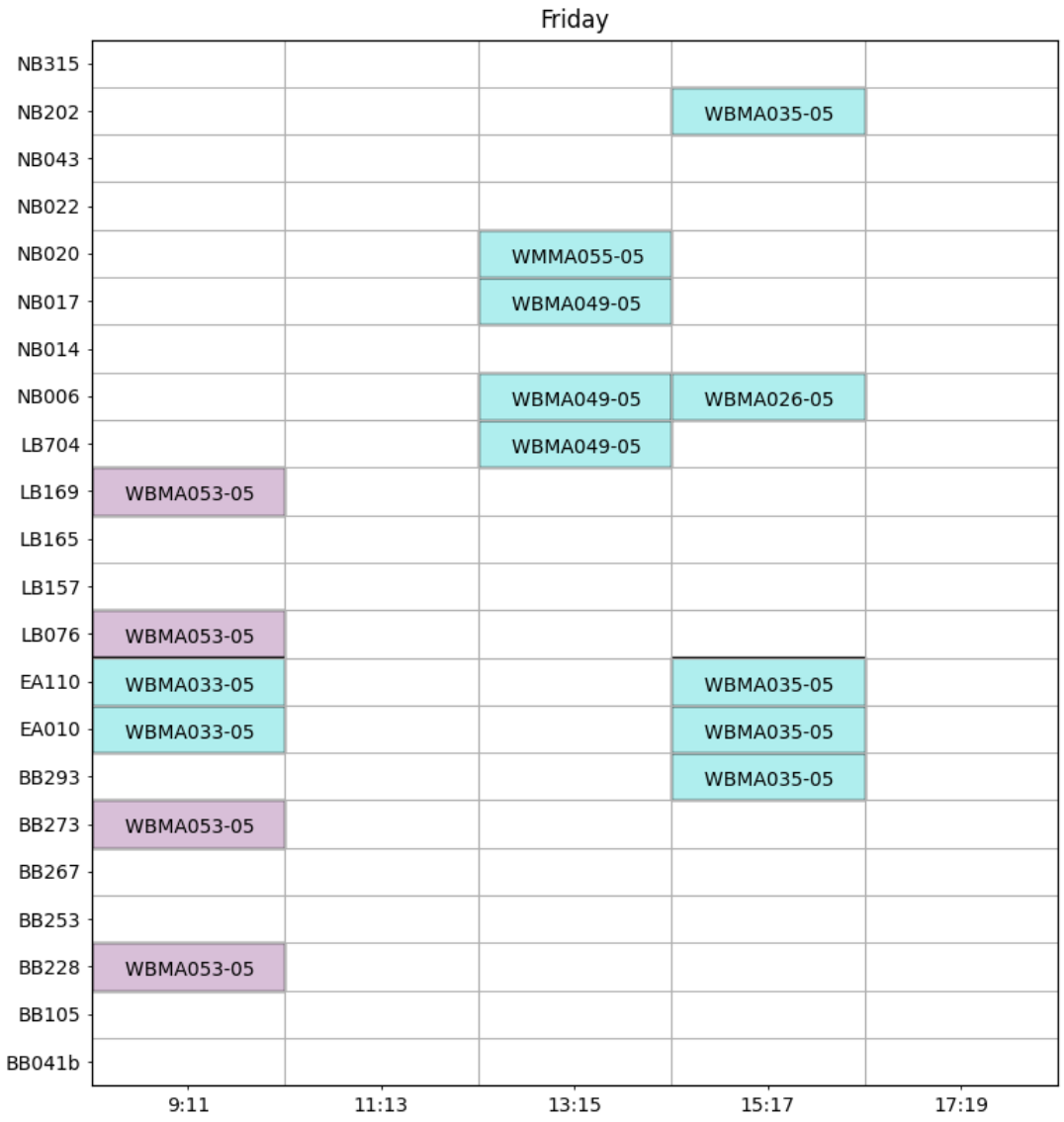

Figure A.35.: Block 2A - Model 2

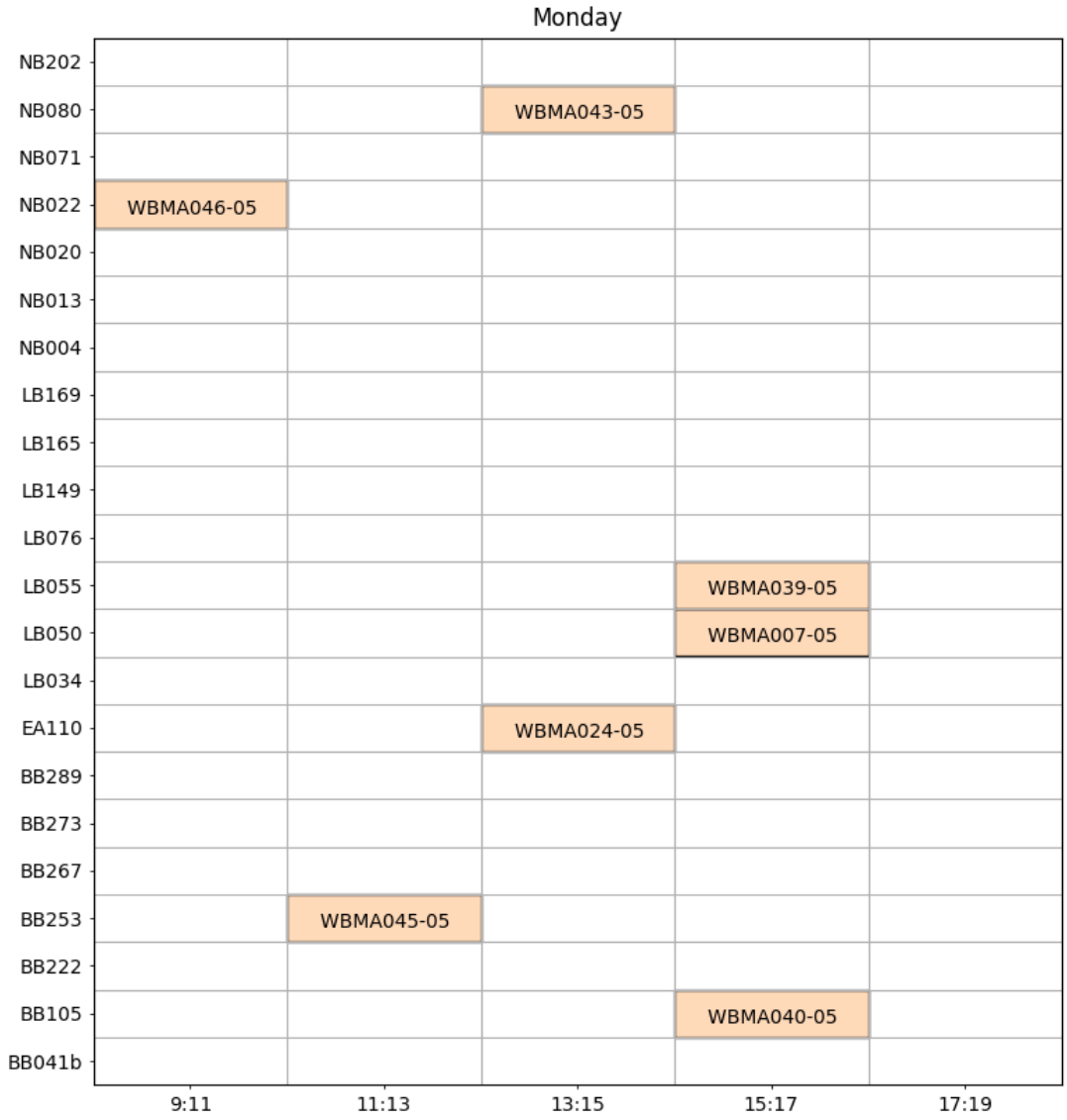

Figure A.36.: Block 2B - Model 2

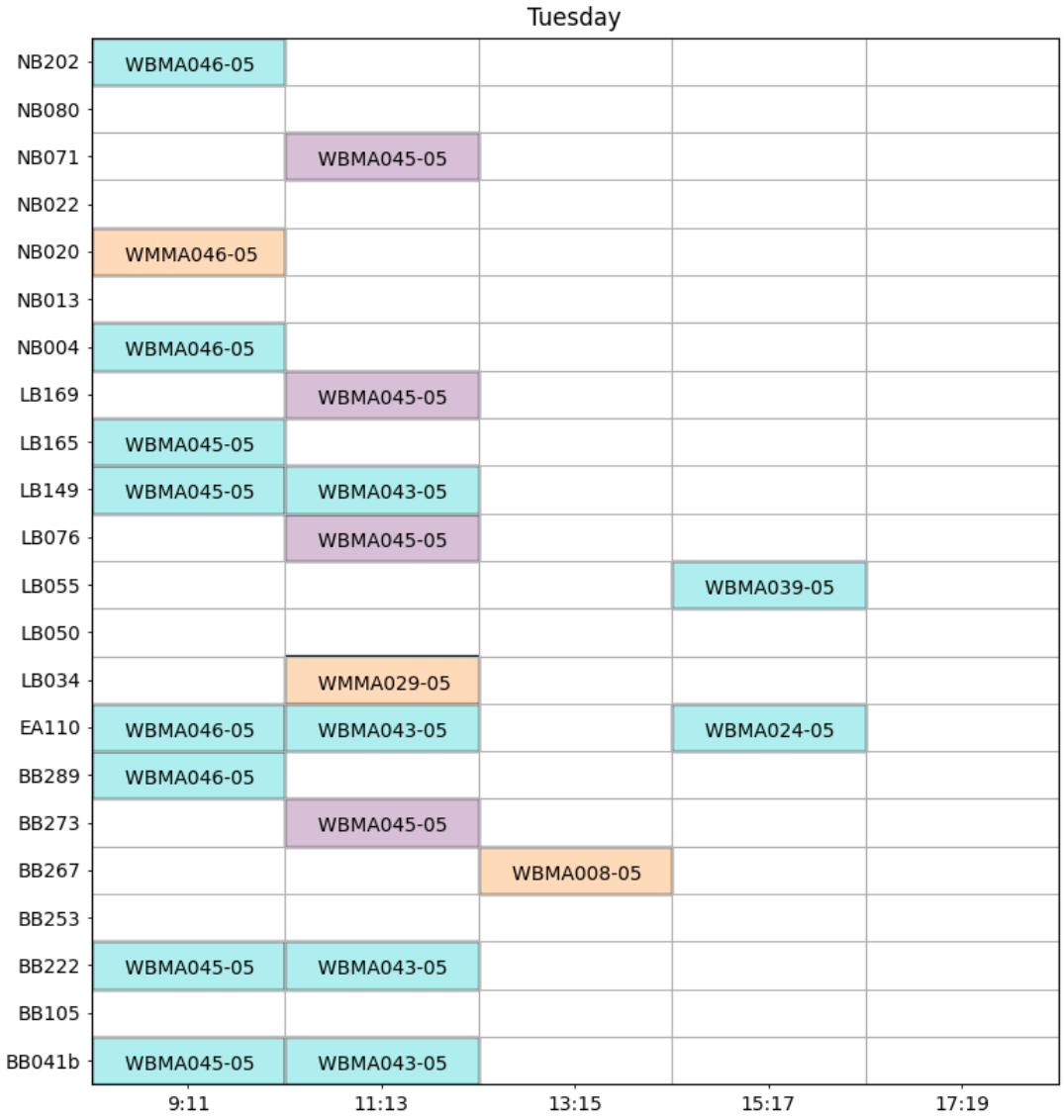

Figure A.37.: Block 2B - Model 2

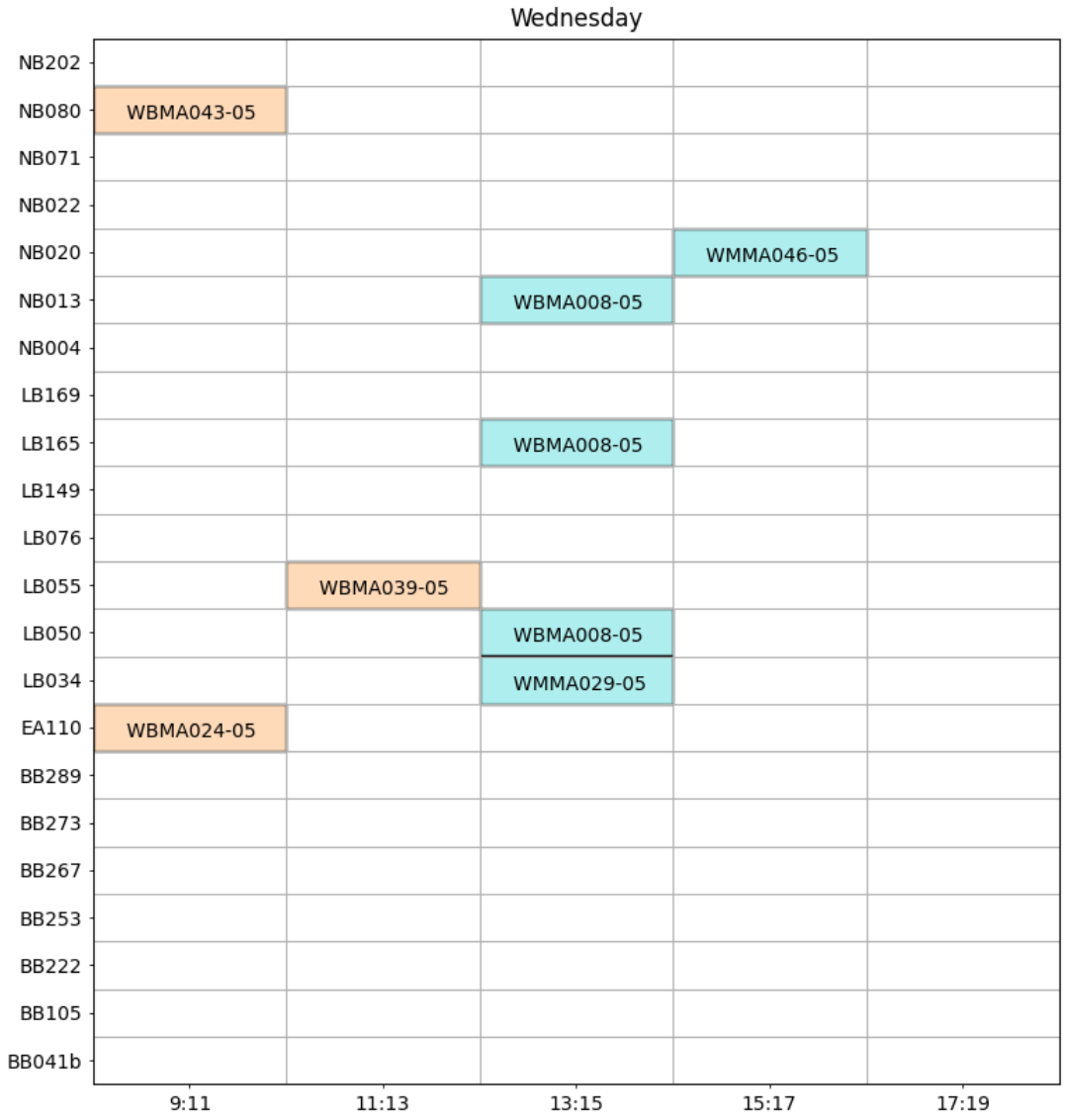

Figure A.38.: Block 2B - Model 2

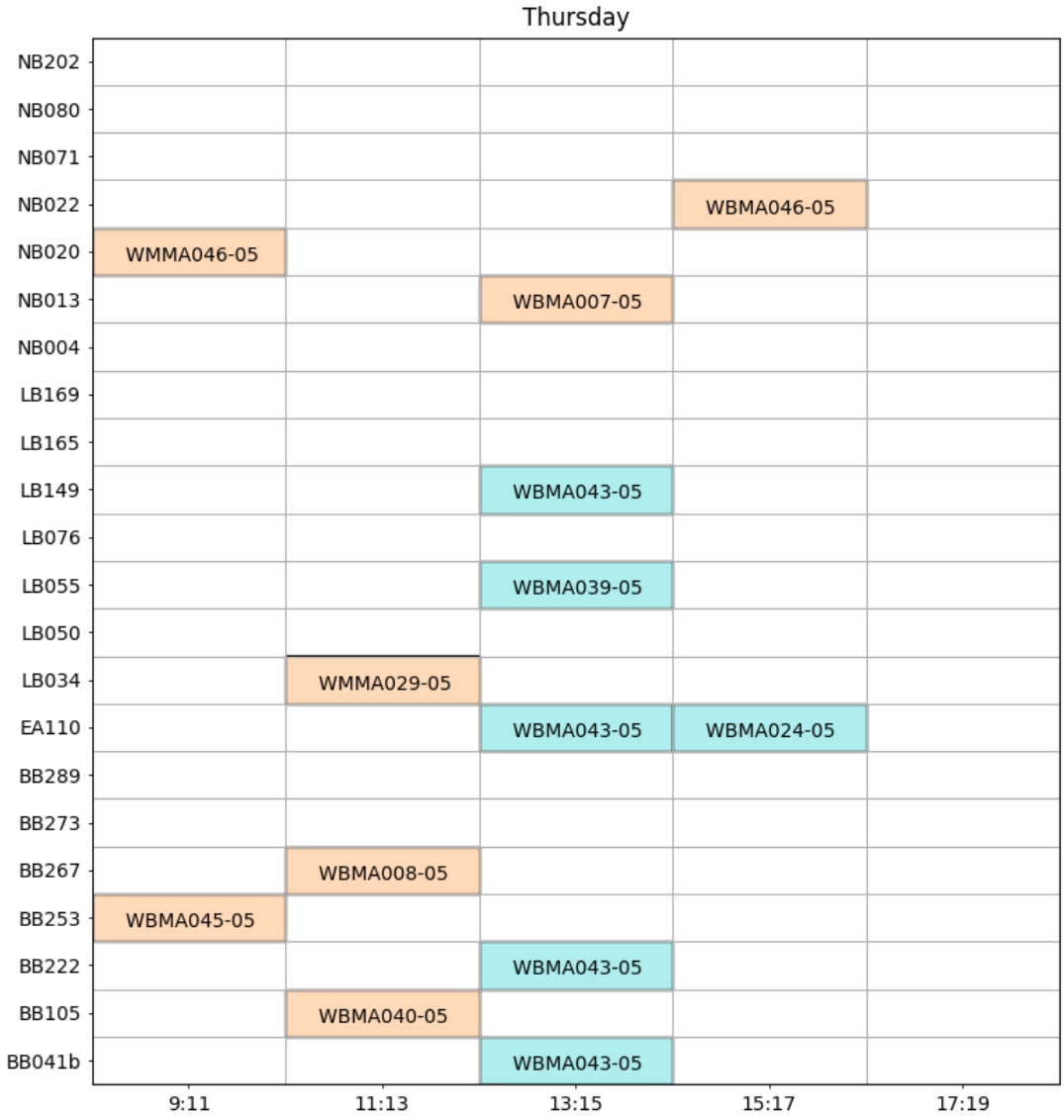

Figure A.39.: Block 2B - Model 2

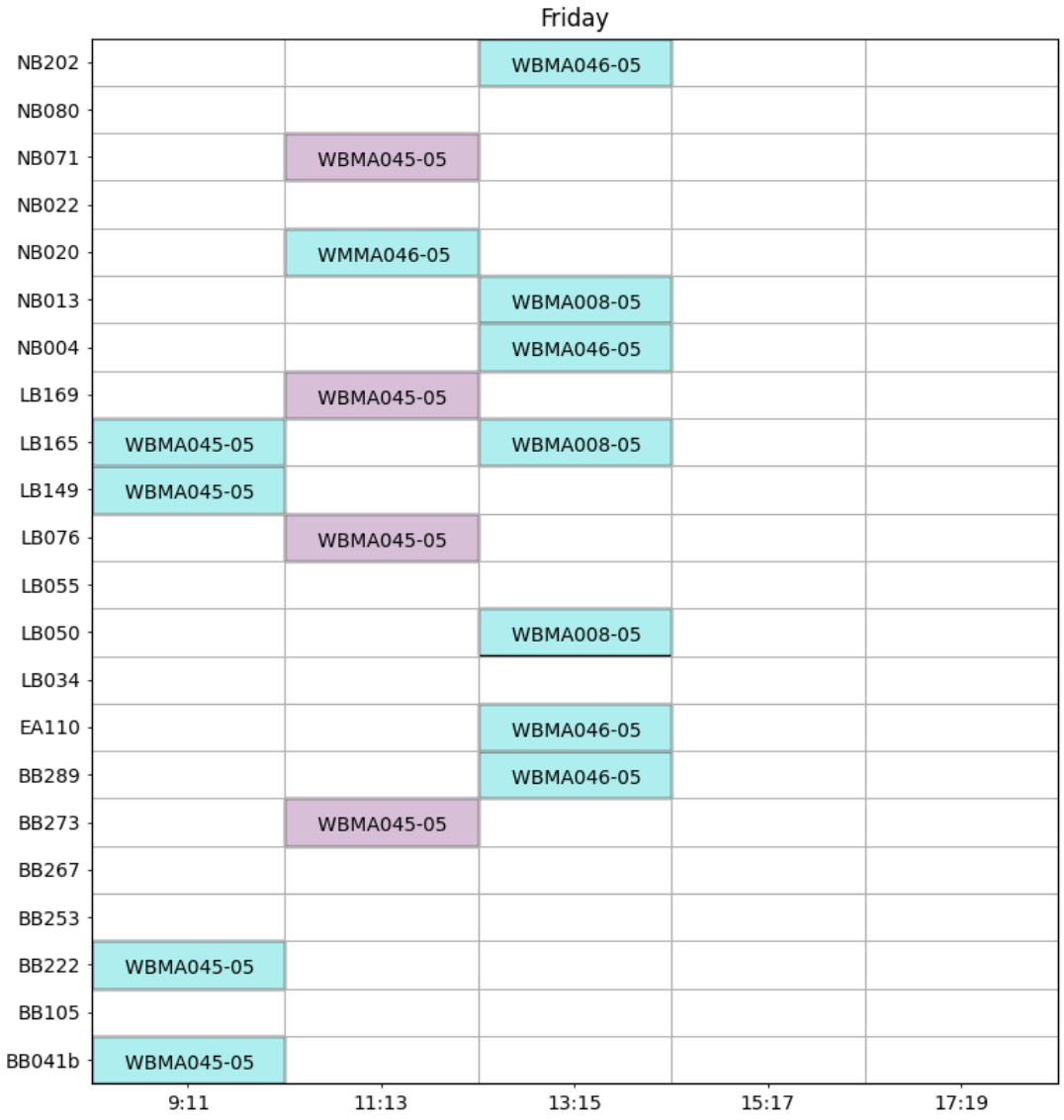

Figure A.40.: Block 2B - Model 2