Universität Rostock

Traditio et Innovatio

Optimization of Pairing Lists

in Sailing Leagues

Name:	Cedric Rönnfeld
Matrikelnummer:	221200104
Abgabedatum:	20.08.2025
Betreuer und Gutachter:	Prof. Dr. Achill Schürmann Universität Rostock Institut für Mathematik
Gutachter:	Prof. Dr. Thomas Kalinowski Universität Rostock Institut für Mathematik

Bachelorarbeit am Institut für Mathematik

Mathematisch-Naturwissenschaftliche Fakultät

Contents

1 Introduction

In the proposed thesis we will study the construction of pairing lists in the application of sailing leagues through the ideas of combinatorial designs and integer optimization, continuing previous work of Robert Schüler and Achill Schürmann [1] on the topic from a more theoretical point.

We define a fairness term based on the distribution of co-occurrences between two teams and aim to find the minimal span f of these distributions for fixed configurations of teams and flights.

To analyze possible values of f we will find necessary divisibility constraints for the cases of $f = 0$ and $f = 1$, providing alternative proofs to known theorems for specific types of designs and Hadamard-Matrices.

One of our main theoretical result is a complete characterization of f in the cases of $n = 6, k = 3$ and $n = 8, k = 4$ which both showing periodic behavior. Using the developed method of proving we design an algorithmic way of constructing or contradicting the existence of fair and almost fair schedules.

On the algorithmic side we compare six MILP and MIQP formulations for finding optimal schedules in the case of $n = 2k$ and evaluate them for different amounts of teams and flights.

Lastly we will apply the work on the polish sailing league providing a alternative schedule that increases our understanding of fairness by a significant amount as well as providing proof for a lower bound of the specific case, because even the showcased alternative may not be optimal. In the scenario of real world application we also discuss the possibilities to further enhance the schedules to form more robust designs, viable for cutting the amount of planned flights while still keeping fairness.

2 Theoretical Background

In practice a sailing league competition consists of a fixed number of teams that compete against each other. The competitions often features multiple flights that each consists of smaller races, called heats.

In our case, a flight always splits the teams into equally sized heats, this way not all teams compete at the same time. This allows for more control over the races but makes it difficult to create a fair ranking for all teams.

Since the finishing times of each team may vary by outside condition like weather, which may change from one race to another, the common way of ranking is by adding the group rankings of each heat. Doing so creates a potential problem, if the teams are not fairly grouped.

For example it could occur, that an average team always faces below average teams and therefore wins their group often, while a originally better team always faces way better teams and loses. The resulting ranking would then place the average team higher.

To avoid such problems and ensure maximal fairness, we hope for a pair of two teams to face each other as often as every other pair.

2.1 Mathematical Model

To start, we need a mathematical representation of the pairing lists, that decide which teams are grouped into heats:

Definition 1. *Let N be a n-set, i.e. $|N| = n < \infty$ and let $k, r \in \mathbb{N} \geq 2$ with, such that $n = t \cdot k$ for $t \in \mathbb{N}_{>1}$.*
We define a (N, k, r)-schedule as a family

$$\mathcal{S} = (A_{ls} \,|\, l \in [r], s \in [t])$$

with $|A_{ls}| = k$ for all $l \in [r], s \in [t]$ and $(A_{ls})_{s=1}^{t}$ being a t-partition of N, that is a division into t disjoint subsets.
We will further call the tuple $(A_{ls})_{s=1}^{t}$ the l-th arrangement of the schedule and therefore A_{ls} the s-th block in the l-th arrangement.

Here N is the set of all teams, k the size of each heats, and r the amount of flights that are competed. A Block A_{ls} is therefore a set of k teams, forming the groups that competes in the s-th heat of the l-th flight.

Definition 2. *An (N, k, r)-schedule $\mathcal{S} = (A_{ls})$ and an (M, k, r)-schedule $\mathcal{T} = (B_{ls})$ are called similar, if there exists a bijections $\pi : N \to M$ and permutations $\phi : [r] \to [r], \psi : [t] \to [t]$, such that:*

$$\pi(A_{\phi(l),\psi(s)}) = B_{ls}$$

If $\phi = id_r$ and $\psi = id_t$ they are called equivalent.

In practice, this definition allows us to call schedules equivalent when the teams have different labels/orders and similar if they only differ in the order of heats or flights.

Next, we need a way to describe how often two teams compete against each other:

Definition 3. *For a given (N, k, r)-schedule $\mathcal{S}$ and $i, j \in N$ with $i \neq j$, we define*

$$\lambda_{ij}(\mathcal{S}) := \left|\left\{l \in [r] \,|\, \exists\, s \in [t] \,:\, \{i, j\} \subset A_{ls}\right\}\right|$$

as the number of co-occurrences of i and j.
We further define

$$\lambda^+(\mathcal{S}) = \max_{\substack{i,j \in N \\ i \neq j}} \lambda_{ij}(\mathcal{S}), \quad \textit{and} \quad \lambda^-(\mathcal{S}) = \min_{\substack{i,j \in N \\ i \neq j}} \lambda_{ij}(\mathcal{S})$$

as the highest and lowest co-occurrences in $\mathcal{S}$.
The difference

$$\Delta(\mathcal{S}) = \lambda^+(\mathcal{S}) - \lambda^-(\mathcal{S})$$

is called fairness deviation of $\mathcal{S}$. If the context is clear, we will just write $\Delta, \lambda^+, \lambda^-$ or λ_{ij}.
For better readability we will write $N^2_* := \left\{(i, j) \in N^2 \,|\, i \neq j\right\}$ from now on.

Corollary 1. *For a given (N, k, r)-schedule $\mathcal{S}$, the co-occurrence numbers are symmetrical, i.e.*

$$\lambda_{ij}(\mathcal{S}) = \lambda_{ji}(\mathcal{S}) \qquad \textit{for all } (i, j) \in N^2_*$$

As the name implies, a high fairness deviation implies an unfair pairing list, while a fairness deviation close to 0 intuitively leads to a more fair grouping of the teams. Two important special cases are 0 and 1:

Definition 4. *A schedule $\mathcal{S}$ is called fair, if its fairness deviation is 0, i.e. $\Delta(\mathcal{S}) = 0$, and almost fair, if it's 1.*

Remark 1. A fair (N, k, r)-schedule $\mathcal{S}$, implying $\lambda_{ij} = \lambda \in \mathbb{N}$ for all $i, j \in N_*^2$, is equivalent to a (n, k, λ)-resolvable balanced incomplete Block Design (RBIBD). For such Designs, many results are already established [2], we therefore aim to abstract the idea of RBIBDs to non constant values for λ_{ij}.

For later representation of pairing lists in examples and solutions we need a way to describe $\mathcal{S}$ more readable:

Definition 5. *Let $\mathcal{S} = (A_{ls})$ be a (N, k, r)-schedule. The schedule-tableau is given by*

$\mathcal{S}$	$-\ N\ -$
1	
$\vdots$	$(S_{li})_{l\in[r],i\in N}$
r	

where S is the assignment matrix of the schedule $\mathcal{S}$ with $S_{li} = s$ for $i \in A_{ls}$. Note, that S_{li} is well defined, because $(A_{ls})_{s=1}^{t}$ is a partition of N for each $l \in [r]$.

For a more readable Table we will often use a renaming of F1,F2,... for the flight numbers to distinguish between flights and teams.

Similar we have another way to represent the co-occurrence numbers using graphs:

Definition 6. *Let $\mathcal{S}$ be a $([n], k, r)$-schedule. The co-occurrence graphs are defined as a family of graphs by $(G_\theta)_{\theta\in\Lambda}$ where $\Lambda = \{\theta \mid \exists (i, j) \in N_*^2 : \lambda_{ij}(\mathcal{S}) = \theta\}$ and*

$$G_\theta = (N, E_\theta) \quad \textit{with} \quad \{i, j\} \in E_\theta \iff \lambda_{ij}(\mathcal{S}) = \theta$$

G_θ is called the θ-co-occurrence graph of $\mathcal{S}$.

Lemma 1. *Let $\mathcal{S}$ be a (N, k, r)-schedule with assignment matrix S, then the following formulations are equivalent definitions for λ_{ij}:*

$$\begin{aligned}\lambda_{ij}(\mathcal{S}) &= \left|\left\{(l, s) \in [r] \times [t] \mid \{i, j\} \in A_{ls}\right\}\right| \\ &= \sum_{l\in[r]} \sum_{s\in[t]} 1_{A_{ls}}(i) \cdot 1_{A_{ls}}(j) \\ &= \sum_{l\in[r]} 1_{\{S_{li}\}}(S_{lj}) = \sum_{l\in[r]} 1_{\{S_{lj}\}}(S_{li})\end{aligned}$$

Proof. Since for each $l \in [r]$ there can only exist at most one $s \in [t]$ with $\{i,j\} \in A_{ls}$ the first equation follows directly from Definition 3 of λ_{ij}:

$$\lambda_{ij}(\mathcal{S}) = \left|\left\{l \in [r] \,|\, \exists\, s \in [t] \,:\, \{i,j\} \subset A_{ls}\right\}\right| = \left|\left\{(l,s) \in [r] \times [t] \,|\, \{i,j\} \in A_{ls}\right\}\right|$$

Using the definition of the indicator function, the second equation follows using

$$1_A(x) := \begin{cases} 1, & x \in A \\ 0, & x \notin A \end{cases} \implies 1_A(x) \cdot 1_A(y) = \begin{cases} 1, & \{x,y\} \subset A \\ 0, & \{x,y\} \not\subset A \end{cases}$$

and therefore

$$\begin{aligned} \left|\left\{(l,s) \in [r] \times [t] \,|\, \{i,j\} \in A_{ls}\right\}\right| &= \sum_{(l,s) \in [r] \times [t]} 1_{A_{ls}}(i) \cdot 1_{A_{ls}}(j) \\ &= \sum_{l \in [r]} \sum_{s \in [t]} 1_{A_{ls}}(i) \cdot 1_{A_{ls}}(j) \end{aligned}$$

The last two equations are based on the initial definition again:

$$\exists s \in [t] : \{i,j\} \in A_{ls} \overset{s=S_{li}}{\iff} j \in A_{l,S_{li}} \iff S_{li} = S_{lj}$$

□

To revisit these Definition, we can examine an example:

Example 1. Given 6 teams, labeled 1 through 6, we aim to create a (not necessary optimal) pairing list for 4 flights, where each race consists of exactly 3 teams.
This is equivalent of finding a $([6], 3, 4)$-schedule. One suitable way of organizing the teams could be as followed

$$\begin{aligned} A_{11} &= \{1,2,3\}, \quad A_{12} = \{4,5,6\}, \\ A_{21} &= \{1,4,6\}, \quad A_{22} = \{2,3,5\}, \\ A_{31} &= \{1,2,4\}, \quad A_{32} = \{3,5,6\}, \\ A_{41} &= \{1,3,6\}, \quad A_{42} = \{2,4,5\}. \end{aligned}$$

The corresponding then schedule-tableau is given by

$\mathcal{S}$	T1	T2	T3	T4	T5	T6
F1	1	1	1	2	2	2
F2	1	2	2	1	2	1
F3	1	1	2	1	2	2
F4	1	2	1	2	2	1

This schedule has two co-occurrence graphs G_0 and G_2:

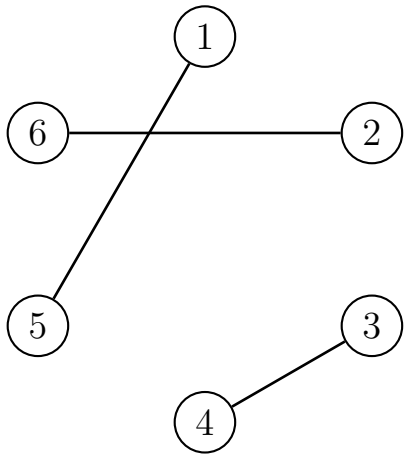


Figure 1: 0-co-occurrence graph

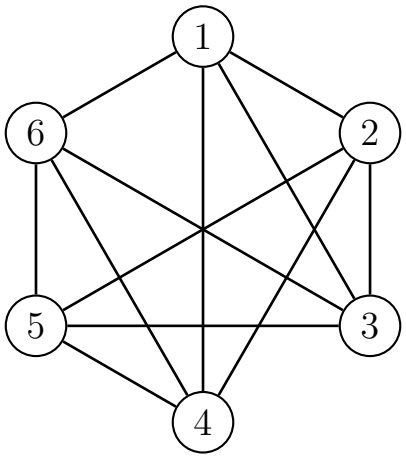


Figure 2: 2-co-occurrence graph

As we can see, this schedule has an fairness deviation of $\Delta(\mathcal{S}) = 2$.
We will later prove, that this is in fact a most fair schedule possible, in a sense of $\Delta(\mathcal{S})$ being minimal under all $([6], 3, 4)$-schedules.

Theorem 1. *Let $\mathcal{S}$ and $\mathcal{T}$ be similar (N, k, r) -and (M, k, r)-schedules with relabelling function $\pi : N \to M$, then*

$$\lambda_{ij}(\mathcal{S}) = \lambda_{\pi(i),\pi(j)}(\mathcal{T}), \qquad \text{for all } (i, j) \in N_*^2$$

and

$$\lambda^+(\mathcal{S}) = \lambda^+(\mathcal{T}), \quad \lambda^-(\mathcal{S}) = \lambda^-(\mathcal{T}), \quad \Delta(\mathcal{S}) = \Delta(\mathcal{T})$$

Proof. Since $\mathcal{S}$ and $\mathcal{T}$ are similar there exist permutations $\phi : [r] \to [r]$ and $\psi : [t] \to [t]$ such that $\pi(A_{\phi(l),\psi(s)}) = B_{ls}$ for all l, s.

From Lemma 1 we use

$$\begin{aligned}
\lambda_{ij}(\mathcal{S}) &= \sum_{l\in[r]}\sum_{s\in[t]} 1_{A_{ls}}(i)\cdot 1_{A_{ls}}(j) \\
&= \sum_{l\in[r]}\sum_{s\in[t]} 1_{\pi^{-1}(B_{\phi^{-1}(l),\psi^{-1}(s)})}(i)\cdot 1_{\pi^{-1}(B_{\phi^{-1}(l),\psi^{-1}(s)})}(j) \\
&= \sum_{l\in[r]}\sum_{s\in[t]} 1_{\pi^{-1}(B_{ls})}(i)\cdot 1_{\pi^{-1}(B_{ls})}(j) \\
&= \sum_{l\in[r]}\sum_{s\in[t]} 1_{B_{ls}}(\pi(i))\cdot 1_{B_{ls}}(\pi(j)) \\
&= \lambda_{\pi(i),\pi(j)}(\mathcal{T})
\end{aligned}$$

□

2.2 Necessary conditions for fair and almost fair schedules

After understanding our problem, we can start by formulating the first restrictions on the existence of fair and almost fair schedules.

Lemma 2. *Let $\mathcal{S}$ be a (N,k,r)-schedule. For every $i\in N$, the following equation holds:*

$$\sum_{j\in N\setminus\{i\}} \lambda_{ij} = r\cdot(k-1)$$

Proof. Let $i\in N$ be fixes. We define a bipartite graph[1] $G=(V,E)$ with

$$V = [r]\dot\cup N\setminus\{i\} \quad\text{and}\quad \{l,j\}\in E \iff \exists\, s\in[t] : \{i,j\}\subset A_{ls}.$$

Using the principle of double counting we can conclude:

$$\sum_{j\in N\setminus\{i\}} \deg j = \sum_{l=1}^{r} \deg l$$

By the definition of co-occurrence numbers $\deg j = \lambda_{ij}$.

[1] Note that although the sets $[r]$ and $N\setminus\{i\}$ may contain the same elements numerically, they represent disjoint parts in the bipartite graph: $[r]$ corresponds to the left vertex class and $N\setminus\{i\}$ to the right.

For $l \in [r]$, $\deg l$ counts the amount of $j \in N\backslash\{i\}$ that are in the same block as i in the l-th arrangement, which is just $|A_{ls}\backslash\{i\}| = k-1$, where $s = S_{li}$.

$$\sum_{j\in N\backslash\{i\}} \lambda_{ij} = \sum_{j\in N\backslash\{i\}} \deg j = \sum_{l=1}^{r} \deg l = \sum_{l=1}^{r} (k-1) = r\cdot(k-1)$$

□

This Lemma leads to the first necessary condition on fair schedules:

Corollary 2. *Let $\mathcal{S}$ be a fair (N,k,r) schedule, then $r\cdot(k-1) \equiv 0 \mod (n-1)$. The quotient $\lambda = \frac{r(k-1)}{n-1}$ will be the constant value of λ_{ij}.*

Proof. Since $\Delta(\mathcal{S}) = 0$ we can write $\lambda_{ij} = \lambda \in \mathbb{N}$ for all $(i,j) \in N_*^2$. Using Lemma 2 an arbitrary $i \in N$ gives

$$(n-1)\cdot\lambda = \sum_{j\in N\backslash\{i\}} \lambda_{ij} = r\cdot(k-1) \implies r\cdot(k-1) \equiv 0 \mod (n-1)$$

□

Revisiting our previous example, we can now say, that for there can not exists a fair $([6],3,4)$-schedule, since

$$\frac{r\cdot(k-1)}{n-1} = \frac{4\cdot 2}{5} = \frac{8}{5} \notin \mathbb{Z}.$$

Another important results of above lemma yields information about the distribution of co-occurrence numbers in almost fair schedules:

Corollary 3. *Let $\mathcal{S}$ be an almost fair (N,k,r)-schedule. For a fixed $i \in N$, the distribution of its co-occurrence numbers λ_{ij} is given by*

$$\left|\{j \in N\backslash\{i\} : \lambda_{ij} = q+1\}\right| = p \quad \textit{and} \quad \left|\{j \in N\backslash\{i\} : \lambda_{ij} = q\}\right| = n-1-p,$$

where

$$r\cdot(k-1) = q\cdot(n-1)+p \quad \textit{for} \quad 1 \le p \le n-2.$$

This also grants the necessary condition $r\cdot(k-1) \not\equiv 0 \mod (n-1)$ for $\Delta(S) = 1$.

Proof. From $\Delta(\mathcal{S}) = \lambda^+ - \lambda^- = 1$, we can define $q := \lambda^-$ and conclude from its definition, that $\lambda_{ij} \in \{q, q+1\}$ for $(i,j) \in N_*^2$.

Let $i \in N$ be fixed and define

$$\Lambda_{\tilde{q}} := \left\{ j \in N \backslash \{i\} : \lambda_{ij} = \tilde{q} \right\} \quad \text{for } \tilde{q} \in \{q, q+1\}$$

Since only q or $q+1$ are viable co-occurrence numbers, we know that Λ_q and Λ_{q+1} form a disjoint partition of $N \backslash \{i\}$. Defining $p := |\Lambda_{q+1}|$ allows us to write $|\Lambda_q| = n - 1 - p$. Using Lemma 2 we now get

$$r \cdot (k-1) = \sum_{j \in N \backslash \{i\}} \lambda_{ij} = \sum_{j \in \Lambda_{q+1}} \lambda_{ij} + \sum_{j \in \Lambda_q} \lambda_{ij} = p(q+1) + (n-p-1)q = q \cdot (n-1) + p$$

From $\Lambda_q, \Lambda_{q+1} \neq \emptyset$ we have $1 \leq p \leq n-2$. □

We will later mainly focus on the case of $n = 2k$ and will now present a stronger version of Corollary 2 as necessary condition for fair schedules:

Theorem 2. *Let $\mathcal{S}$ be a fair (N, k, r)-schedule with $n = 2k$ and k odd, then $r \cdot (k-1) \equiv 0 \mod 2(n-1)$.*

Proof. Let $N = \{x_1, \ldots, x_n\}$. We define a helper function

$$g_s^i(a, b) := \left| \{ l \mid a \leq l \leq b, x_i \in A_{ls} \} \right| = \sum_{l=a}^{b} 1_{\{s\}}(S_{lx_i})$$

for $i \in [n], s \in [t]$ and $1 \leq a \leq b \leq r$, as the amount of times, x_i appears in the s-th block of the a-th to b-th arrangement.

Since $\mathcal{S}$ is fair, we know $\lambda_{x_i x_j} = \lambda \in \mathbb{N}$ for $(x_i, x_j) \in N_*^2$. This especially implies $\lambda_{x_1 x_2} = \lambda_{x_1 x_3} = \lambda_{x_2 x_3} = \lambda$.
From Theorem 1 we can assume

$$x_1 \in A_{l1} \quad \text{and} \quad x_2 \in \begin{cases} A_{l1}, & 1 \leq l \leq \lambda \\ A_{l2}, & \lambda + 1 \leq l \leq r \end{cases}, \qquad \text{for } 1 \leq l \leq r \tag{1}$$

Let $\mu = g_1^3(1, \lambda)$ with $1 \leq \mu \leq \lambda$ the amount of times x_3 appears in the first block of the first λ arrangements.

x_1 being always in the first block, allows us to write

$$\begin{aligned}\lambda = \lambda_{x_1x_3} &= g_1^3(1,r)\\ &= \sum_{l=1}^{r} 1_{\{1\}}(S_{lx_i})\\ &= \sum_{l=1}^{\lambda} 1_{\{1\}}(S_{lx_i}) + \sum_{l=\lambda+1}^{r} 1_{\{1\}}(S_{lx_i})\\ &= g_1^3(1,\lambda) + g_1^3(\lambda+1,r) = \mu + g_1^3(\lambda+1,r)\end{aligned}$$

And therefore $g_1^3(\lambda+1,r) = \lambda - \mu$ is the amount of times x_3 appears in the last block of the remaining arrangements.

Since we only have 2 blocks per arrangement ($n = 2k$), the blocks A_{l1} and A_{l2} form a partition N, allowing

$$\begin{aligned}g_1^3(a,b) + g_2^3(a,b) &= \big|\{l \,|\, a \le l \le b, x_i \in A_{l1}\}\big| + \big|\{l \,|\, a \le l \le b, x_i \in A_{l2}\}\big|\\ &= \big|\{l \,|\, a \le l \le b, x_i \in A_{l1} \text{ or } x_i \in A_{l2}\}\big|\\ &= \big|\{l \,|\, a \le l \le b, x_i \in A_{l1} \cup A_{l2}\}\big|\\ &= \big|\{l \,|\, a \le l \le b, x_i \in N\}\big|\\ &= \big|\{l \,|\, a \le l \le b\}\big|\\ &= b - a + 1\end{aligned}$$

This leads to

$$g_2^3(\lambda+1,r) = r - \lambda - g_1^3(\lambda+1,r) = r - \lambda - (\lambda - \mu) = r + \mu - 2\lambda$$

From the placement of x_2 in (1) we get

$$\lambda = \lambda_{x_2,x_3} = g_1^3(1,\lambda) + g_2^3(\lambda+1,r) = \mu + r + \mu - 2\lambda = 2\mu + r - 2\lambda$$

Rearranging this equation and using Corollary 2 we have

$$\mu = \frac{1}{2}(3\lambda - r) = \frac{1}{2}\left(3\frac{r\cdot(k-1)}{n-1} - r\right) = \frac{r(k-2)}{2(2k-1)}$$

Because $\mu \in \mathbb{N}$ it follows, that $2(2k-1)$ has to be a divisor of $r(k-2)$. Since $2(2k-1)$ is even and $k-2$ is odd, we need r to be even and therefore $2 \,|\, r(k-1)$. Corollary 2 gives

$2k-1 \mid r(k-1)$ and since $gcd(2, 2k-1)$ this allows for $2(2k-1) \mid r(k-1)$. Granting the expected result of $r \cdot (k-1) \equiv 0 \mod 2 \cdot (2k-1)$. □

Note, that this condition only works for k odd. In a later section we will show that, for example, there always exists a fair $([8], 4, 7p)$-schedule for each $p \in \mathbb{N}$, which would form a contradiction to expanding the theorem to even k. A still open conjecture in coding theory, is if there always exists a fair $([2k], k, 2k-1)$-schedule when k is even. Many cases are already shown, the smallest unproven case being $k = 334$ (cf. [3],[4]).

Remark 2. A fair $(n, n/2, n-1)$-schedule (for n even) is equivalent to a Hadamard Matrix, that is a matrix $H \in \{1, -1\}^{n \times n}$ with $HH^T = nI_n$. Theorem 2 therefore provides an alternative proof to the well known fact that such Hadamard Matrices can only exists for n being a multiple of 4 (or $n \in \{1, 2\}$) ([5]).

3 Optimization Analysis

Revisiting Example 1, we now aim to get information about optimal schedules for a given parameter set (N, k, r)

Definition 7. *For $n, k, r \in \mathbb{N}_{>1}$ with $n = tk$ for $t \in \mathbb{N}_{>1}$ (2) we define the minimal fairness deviation as*

$$f(n, k, r) = \min\left\{\Delta(\mathcal{S}) \,|\, \mathcal{S} \text{ is a } ([n], k, r)\text{-schedule}\right\}$$

We further extend this definition by

$$f(n, k, 0) = 0, \quad f(n, k, 1) = 1, \quad f(0, 0, r) = 0, \quad f(t, 1, r) = 0$$

to allow $n, k \in \mathbb{N}_0$, even though this was not covered in above section, as it's analysis is trivial.

Definition 8. *A (N, k, r)-schedule $\mathcal{S}^*$ is called optimal schedule, if*

$$\Delta(\mathcal{S}^*) = \min\{\Delta(\mathcal{S}) \,|\, \mathcal{S} \text{ is } (M, k, r)\text{-schedule}, |M| = n\}$$

Using Theorem 1 this is equivalent to the condition

$$\Delta(\mathcal{S}^*) = \min\{\Delta(\mathcal{S}) \,|\, \mathcal{S} \text{ is } ([n], k, r)\text{-schedule}\}.$$

Allowing us to only consider the set $[n]$ for optimization analysis.

3.1 Properties of the minimal fairness deviation

Our first important result is that the existence of a fair schedule for $([n], k, r)$ for $k, r \geq 2$ always implies the existence of an almost fair schedule for $([n], k, r \pm 1)$.

Lemma 3. *For parameters n, k, r as described in Definition 7, the following inequality holds:*

$$|f(n, k, r) - f(n, k, r + 1)| \leq 1$$

Proof. For $k = 0, 1$ this is trivial, since $f(n, k, r)$ is constant.
Considering the special cases for r we see that $|f(n, k, 0) - f(n, k, 1)| = |0 - 1| = 1$ which satisfies the inequality. Since $0 \leq f(n, k, r) \leq r$ for obvious reasons, it also follows that $|f(n, k, 1) - f(n, k, 2)| = |1 - f(n, k, 2)| \leq 1$.

We assume that $k, r \geq 2$ and split the absolute value inequality into two parts:

(i) $f(n, k, r + 1) - f(n, k, r) \leq 1$

(ii) $f(n, k, r) - f(n, k, r + 1) \leq 1$

Allowing us to directly conclude $|f(n, k, r) - f(n, k, r + 1)| \leq 1$ from their validity.

Starting with (i), let $\mathcal{S}^{(op)} = (A_{ls})$ be an optimal $([n], k, r)$-schedule, i.e. $\Delta(\mathcal{S}^{(op)}) = f(n, k, r)$. We construct a $([n], k, r+1)$-schedule $\mathcal{S}' = (B_{ls})$ from $\mathcal{S}^{(op)}$ by simply adding a arbitrary arrangement $(B_{r+1,s})$. Examining a pair $(i, j) \in N_*^2$ gives us two possible outcomes for $\lambda_{ij}(\mathcal{S}')$:

$$\lambda_{ij}(\mathcal{S}') = \sum_{l=1}^{r+1} 1_{S'_{li}}(S'_{lj}) = \sum_{l=1}^{r} 1_{S_{li}^{(op)}}(S_{lj}^{(op)}) + 1_{S'_{r+1,i}}(S'_{r+1,j}) = \lambda(ij)(\mathcal{S}^{(op)}) + 1_{S'_{r+1,i}}(S'_{r+1,j})$$

Since $1_{S'_{r+1,i}}(S'_{r+1,j}) \in \{0, 1\}$ this results in

$$\lambda(ij)(\mathcal{S}^{(op)}) \leq \lambda(ij)(\mathcal{S}') \leq \lambda(ij)(\mathcal{S}^{(op)}) + 1, \qquad \text{for all } (i, j) \in N_*^2$$

and therefore

$$\lambda^+(\mathcal{S}') \leq \lambda^+(\mathcal{S}^{(op)}) + 1 \quad \text{and} \quad \lambda^-(\mathcal{S}') \geq \lambda^-(\mathcal{S}^{(op)}).$$

For the minimal fairness deviation we then get

$$\begin{aligned} f(n, k, r + 1) &= \min\left\{\Delta(\mathcal{S}) \,|\, \mathcal{S} \text{ is a } ([n], k, r + 1)\text{-schedule}\right\} \\ &\leq \Delta(\mathcal{S}') \\ &= \lambda^+(\mathcal{S}') - \lambda^-(\mathcal{S}') \\ &\leq \lambda^+(\mathcal{S}^{(op)}) + 1 - \lambda^-(\mathcal{S}^{(op)}) \\ &= f(n, k, r) + 1 \end{aligned}$$

which finishes (i).

The second part (ii) works analogously by simply removing the r-th arrangement from an optimal $([n], k, r+1)$-schedule $\mathcal{S}^{(op)}$ to construct a $([n], k, r)$-schedule $\mathcal{S}$ giving

$$\lambda(ij)(\mathcal{S}^{(op)}) - 1 \leq \lambda(ij)(\mathcal{S}') \leq \lambda(ij)(\mathcal{S}^{(op)}), \qquad \text{for all } (i,j) \in N_*^2$$

This again results in the wanted inequality by estimating $f(n,k,r)$:

$$\begin{aligned} f(n,k,r) &= \min\left\{\Delta(\mathcal{S}) \,|\, \mathcal{S} \text{ is a } ([n],k,r)\text{-schedule}\right\} \\ &\leq \Delta(\mathcal{S}') \\ &= \lambda^+(\mathcal{S}') - \lambda^-(\mathcal{S}') \\ &\leq \lambda^+(\mathcal{S}^{(op)}) - (\lambda^-(\mathcal{S}^{(op)}) - 1) \\ &= f(n,k,r+1) + 1 \end{aligned}$$

□

As a direct results we get two corollaries:

Corollary 4. *The minimal fairness deviation is a contraction in r, i.e.*

$$|f(n,k,r) - f(n,k,\tilde{r})| \leq |r - \tilde{r}| \qquad \textit{for all } r, \tilde{r} \in \mathbb{N}_{>2}.$$

Proof. W.l.o.g. $\tilde{r} > r$. Using Lemma 3 we get

$$\begin{aligned} |f(n,k,r) - f(n,k,\tilde{r})| &= \left|\sum_{j=r}^{\tilde{r}-1} f(n,k,j) - f(n,k,j+1)\right| \\ &\leq \sum_{j=r}^{\tilde{r}-1} |f(n,k,j) - f(n,k,j+1)| \\ &\leq \sum_{j=r}^{\tilde{r}-1} 1 = (\tilde{r} - 1 - r + 1) \cdot 1 \\ &= \tilde{r} - r \end{aligned}$$

□

Corollary 5. *The existence of a fair schedule for $([n], k, r)$ implies the existence of a almost fair schedule for $([n], k, r \pm 1)$ and prohibits the existence of fair $([n], k, r \pm 1)$-*

schedules, i.e.

$$f(n,k,r) = 0 \implies f(n,k,r-1) = f(n,k,r+1) = 0$$

Proof. Applying Corollary 4 to $\tilde{r} = \pm 1$ we get, that $|f(n,k,r \pm 1)| \leq 1$ and since f is non-negative this implies that $f(n,k,r \pm 1)$ is either 0 or 1.
Assuming $f(n,k,r \pm 1) = 0$ while also $f(n,k,r) = 0$ would give

$$r \cdot (k-1) \equiv 0 \mod (n-1) \quad \text{and} \quad (r \pm 1) \cdot (k-1) \equiv 0 \mod (n-1)$$

because of Corollary 2. Adding these congruences yield $k - 1 \equiv 0 \mod (n-1)$. But for this congruence to hold we would have $|n-1| \leq |k-1|$ or $k - 1 = 0$, both being a contradiction to $n > k > 2$. □

We will later prove that, for the cases $n = 6, k = 3$ and $n = 8, k = 4$, these are also the only times $f(n,k,r) = 1$ occur. Even though numerical analysis leads to the conjecture that $f(n,k,r) = 1$ always results in $f(n,k,r+1) = 0$ or $f(n,k,r+1) = 0$, this still remains unproven.

To finish the properties of f for now, we will include the subadditivity of f, which is needed in the next sections.

Lemma 4. *The minimal fairness deviation is subadditive in r, i.e.*

$$f(n,k,r_1 + r_2) \leq f(n,k,r_1) + f(n,k,r_2) \quad \textit{for all } r_1, r_2 \in \mathbb{N}.$$

Proof. The case of $n = 0, 1$ or $r = 0$ is trivial and the case $r = 1$ follows directly from Lemma 3.
Let $\mathcal{S}^{(1)} = (A_{ls})$ and $\mathcal{S}^{(2)} = (B_{ls})$ be optimal schedules for r_1 and r_2 respectively.
Similar to the proof of Lemma 3 we construct a new $([n], k, r_1 + r_2)$-schedule to create an upper bound for $f(n,k,r_1 + r_2)$. We define $\mathcal{S}' = (C_{ls})$ by concatenating $\mathcal{S}^{(1)}$ and $\mathcal{S}^{(2)}$:

$$C_{ls} = \begin{cases} A_{ls}, & 1 \leq l \leq r_1 \\ B_{l-r_1,s} & r_1 < l \leq r_1 + r_2 \end{cases} \quad \text{and therefore} \quad S' = \left(\frac{S^1}{S^2}\right)$$

For our new schedule we get

$$\begin{aligned}\lambda_{ij}(\mathcal{S}') &= \sum_{l=1}^{r_1+r_2} 1_{S'_{li}}(S'_{lj}) \\ &= \sum_{l=1}^{r_1} 1_{S'_{li}}(S'_{lj}) + \sum_{l=r_1+1}^{r_1+r_2} 1_{S'_{li}}(S'_{lj}) \\ &= \sum_{l=1}^{r_1} 1_{S^{(1)}_{li}}(S^{(1)}_{lj}) + \sum_{l=1}^{r_2} 1_{S^{(2)}_{li}}(S^{(2)}_{lj}) \\ &= \lambda_{ij}(\mathcal{S}^{(1)}) + \lambda_{ij}(\mathcal{S}^{(2)})\end{aligned}$$

and therefore

$$\Delta(\mathcal{S}') = \Delta(\mathcal{S}^{(1)}) + \Delta(\mathcal{S}^{(2)}) \implies f(n,k,r_1+r_2) \leq f(n,k,r_1) + f(n,k,r_2)$$

□

From now on we will mainly focus on the case $n = 2k$, as needed in Theorem 2, since this allows for easier analysis, due to the binary behavior of blocks in an arrangement.

3.2 Formulation as programming problem

Based on the definition of our problem we will be able to formulate a optimization problem, that we can then solve using known algorithms.

We start by defining binary variables

$$b_{li} = 1_{\{1\}}(S_{li}) \qquad \text{for } l \in [r], i \in [2k]$$

This allows for b_{li} to determine if the team i is in the first heat of flight l. (The nature of $n = 2k$ allows for two cases, that can be represented using binary variables). Technically this is enough to fully characterize a schedule. Now we are only left with the task of somehow representing $\Delta(\mathcal{S})$ in terms of these b_{li} and restricting each heat to exactly k teams.

The restriction to k teams can be done by simply counting how often $b_{li} = 1$ for a fixed flight l. This yields the following first set of constraints:

$$\sum_{i=1}^{2k} b_{li} \qquad \text{for } l \in [r]$$

From the prior chapter we know that we are able to represent $\lambda_{ij}(\mathcal{S})$ in terms of S_{li} and S_{lj} for $l = 1, \ldots, r$. This implies that we can also represent it similar in terms of b_{li} and b_{lj} for $l = 1, \ldots, r$. A trivial way of calculating $\lambda_{ij}(\mathcal{S})$ is by simply looping over each flight and counting how often b_{li} and b_{lj} are equal. We will model this process using a new variable

$$c_{lij} = \begin{cases} 1, & b_{li} = b_{lj} \\ 0, & b_{li} \neq b_{lj} \end{cases} \qquad \text{for } l \in [r], i, j \in [2k]$$

Since this way of defining is only good for an intuition but not suitable as an actual constraint we will rewrite it. Based on the way that the constraint is formulated we will get different problem formulations that perform differently, which is why we will cover two such approaches.

The first way is by formulating a quadratic constraint. We can check for equality by evaluating $b_{li} - b_{lj}$, which will yield 0 if they are equal and ± 1 if not. This results in the following constraint:

$$c_{lij} = 1 - (b_{li} - b_{lj})^2$$

An alternative way is by modeling this constraint using multiple linear inequalities. From exhausting all possible combinations:

b_{li}	b_{lj}	c_{lij}
0	0	1
0	1	0
1	0	0
1	1	1

we can form a tetrahedron with the different combinations as edges:

$$P = \text{conv}\left\{ \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \right\}$$

By rewriting this set using the intersection of half-spaces and fixing $c_{li} \in \{0, 1\}$ we get our constraints:

$$\begin{aligned} (b_{li}, b_{lj}, c_{lij})^T \in P \iff \quad & b_{li} + b_{lj} + c_{lij} \geq 1 \\ & b_{li} + b_{lj} - c_{lij} \leq 1 \\ & b_{li} - b_{lj} + c_{lij} \leq 1 \\ & -b_{li} + b_{lj} + c_{lij} \leq 1 \end{aligned}$$

This now allows us for counting c_{lij} to get $\lambda_{ij}(\mathcal{S}) = \sum_{l=1}^{r} c_{lij}$ and therefore allowing us to formulate a problem that minimizes $\Delta(\mathcal{S})$.
Since $\lambda^+(\mathcal{S})$ and $\lambda^-(\mathcal{S})$ form upper and lower bounds respectively, we can define them using inequalities.

We therefore get the following two possible formulations:

Binary quadratic problem:

$$
\begin{aligned}
\text{Min} \quad & \lambda^+ - \lambda^- && \\
\text{s.t.} \quad & \sum_{i=1}^{2k} b_{li} = k, && \text{for } l \in [r] \\
& c_{lij} = 1 - (b_{li} - b_{lj})^2, && \text{for } l \in [r],\ i,j \in [2k] \\
& \lambda_{ij} = \sum_{l=1}^{r} c_{lij}, && \text{for } i,j \in [2k] \\
& \lambda^+ \geq \lambda_{ij}, && \text{for } i,j \in [2k] \\
& \lambda^- \leq \lambda_{ij}, && \text{for } i,j \in [2k] \\
& b_{li} \in \{0,1\}, && \text{for } l \in [r],\ i \in [2k] \\
& c_{lij} \in \{0,1\}, && \text{for } l \in [r],\ i,j \in [2k] \\
& \lambda_{ij} \in \mathbb{N}, && \text{for } i,j \in [2k] \\
& \lambda^+, \lambda^- \in \mathbb{N}, && \text{for } i,j \in [2k]
\end{aligned}
$$

Linear integer problem:

$$
\begin{aligned}
\text{Min} \quad & \lambda^+ - \lambda^- && \\
\text{s.t.} \quad & \sum_{i=1}^{2k} b_{li} = k, && \text{for } l \in [r] \\
& b_{li} + b_{lj} + c_{lij} \geq 1, && \text{for } l \in [r],\ i,j \in [2k] \\
& b_{li} + b_{lj} - c_{lij} \leq 1, && \text{for } l \in [r],\ i,j \in [2k] \\
& b_{li} - b_{lj} + c_{lij} \leq 1, && \text{for } l \in [r],\ i,j \in [2k] \\
& -b_{li} + b_{lj} + c_{lij} \leq 1, && \text{for } l \in [r],\ i,j \in [2k] \\
& \lambda_{ij} = \sum_{l=1}^{r} c_{lij}, && \text{for } i,j \in [2k] \\
& \lambda^+ \geq \lambda_{ij}, && \text{for } i,j \in [2k] \\
& \lambda^- \leq \lambda_{ij}, && \text{for } i,j \in [2k] \\
& b_{li} \in \{0,1\}, && \text{for } l \in [r],\ i \in [2k] \\
& c_{lij} \in \{0,1\}, && \text{for } l \in [r],\ i,j \in [2k] \\
& \lambda_{ij} \in \mathbb{N}, && \text{for } i,j \in [2k] \\
& \lambda^+, \lambda^- \in \mathbb{N}, && \text{for } i,j \in [2k]
\end{aligned}
$$

In practice, we can simplify both problems in multiple ways, by using the symmetry of c_{lij} or by eliminating variables that do not need to occur in the constraints at all (λ_{ij} as example).

Such simplifications lead to the following systems:

Simplified binary quadratic problem (P1*):

$$
\begin{aligned}
\text{Min} \quad & \lambda^+ - \lambda^- & & \\
\text{s.t.} \quad & \sum_{i=1}^{2k} b_{li} = k, & & \text{for } l \in [r] \\
& r - \lambda^+ \leq \sum_{l=1}^{r} (b_{li} - b_{lj})^2, & & \text{for } 1 \leq i < j \leq 2k \\
& r - \lambda^- \geq \sum_{l=1}^{r} (b_{li} - b_{lj})^2, & & \text{for } 1 \leq i < j \leq 2k \\
& b_{li} \in \{0, 1\}, & & \text{for } l \in [r],\ i \in [2k] \\
& \lambda^+, \lambda^- \in \mathbb{N}, & & \text{for } i, j \in [2k]
\end{aligned}
$$

Simplified linear integer problem (P2*):

$$
\begin{aligned}
\text{Min} \quad & \lambda^+ - \lambda^- && \\
\text{s.t.} \quad & \sum_{i=1}^{2k} b_{li} = k, && \text{for } l \in [r] \\
& b_{li} + b_{lj} + c_{lij} \geq 1, && \text{for } l \in [r],\ 1 \leq i < j \leq 2k \\
& b_{li} + b_{lj} - c_{lij} \leq 1, && \text{for } l \in [r],\ 1 \leq i < j \leq 2k \\
& b_{li} - b_{lj} + c_{lij} \leq 1, && \text{for } l \in [r],\ 1 \leq i < j \leq 2k \\
& -b_{li} + b_{lj} + c_{lij} \leq 1, && \text{for } l \in [r],\ 1 \leq i < j \leq 2k \\
& \lambda^+ \geq \sum_{l=1}^{r} c_{lij}, && \text{for } 1 \leq i < j \leq 2k \\
& \lambda^- \leq \sum_{l=1}^{r} c_{lij}, && \text{for } 1 \leq i < j \leq 2k \\
& b_{li} \in \{0,1\}, && \text{for } l \in [r],\ i \in [2k] \\
& c_{lij} \in \{0,1\}, && \text{for } l \in [r],\ 1 \leq i < j \leq 2k \\
& \lambda^+, \lambda^- \in \mathbb{N}, && \text{for } i, j \in [2k]
\end{aligned}
$$

There are also ways to restrict the problem more without loosing a minimal solution by adding symmetry breaking constraints and therefore decreasing the feasible region. Since $\Delta(S)$ is invariant under similarity (cf. Theorem 1) we can force an order of teams, flights and heats that may lead to better performance in solving the problems:

To restrict permutations of team labeling we can fix the first flight by adding

$$
b_{1i} = \begin{cases} 1, & 1 \leq i \leq k \\ 0, & k+1 \leq i \leq 2k \end{cases} \tag{C1}
$$

Preventing permutations of heats inside a flight is possible by fixing the first team to always be in the first heat

$$
b_{l1} = 1 \quad \text{for } 1 \leq l \leq r \tag{C2}
$$

These two constraints turn out to be always beneficial to add. This changes with the restricting the permutation of flights. The natural way of doing so is by defining an order on the set of all possible flights:

Definition 9. *For finite sets $A, B \subset \mathbb{N}$ (i.e. $A, B \in [\mathbb{N}]^{<\omega}$) we define a binary relation:*

$$A \leq B \iff \sum_{a \in A} 2^a \leq \sum_{b \in B} 2^b$$

Lemma 5. *The relation from Definition 9 is a total order, i.e.*

1. $\forall A \in [\mathbb{N}]^{<\omega} : A \leq A$
2. $\forall A, B, C \in [\mathbb{N}]^{<\omega} : A \leq B$ *and* $B \leq C \implies A \leq C$
3. $\forall A, B \in [\mathbb{N}]^{<\omega} : A \leq B$ *and* $B \leq A \implies A = B$
4. $\forall A, B \in [\mathbb{N}]^{<\omega} : A \leq B$ *or* $B \leq A$

Proof. The properties 1,2 and 4 are trivial.

For proving 3 assume that $A \neq B$, if A and B are not equal but both finite there exist a biggest element $k = \max(A \Delta B)$, If this k is in $A \backslash B$. If $k \in A$ and $k > b$ for all $b \in B$, we can conclude a contradiction to $A \leq B$:

$$\sum_{a \in A} 2^a \geq 2^k \geq 2^k - 1 = \sum_{i=0}^{k-1} 2^i \geq \sum_{b \in B} 2^b$$

If instead $k \in B \backslash A$, analogous reasoning leads to $B \leq A$. Resulting in either case contradicting $A \leq B$ and $B \leq A$. □

We can use this total order for adding the constraint $A_{l,1} \leq A_{l+1,1}$ for $1 \leq l < r$, rewriting this in terms of b_{li} gives

$$\sum_{i=1}^{2k} b_{l,i} \cdot 2^i \leq \sum_{i=1}^{2k} b_{l+1,i} \cdot 2^i \quad \text{for } 1 \leq l < r \tag{C3}$$

As said this constraint does not always increase the performance, mainly because of the high coefficients 2^i. An alternative way is by adding the weaker constraint

$$\sum_{i=1}^{2k} b_{l,i} \cdot i \leq \sum_{i=1}^{2k} b_{l+1,i} \cdot i \quad \text{for } 1 \leq l < r \tag{C4}$$

This is not an order and therefore will not prevent all permutations, but we hope that giving a compromise for not having too high coefficients while still restricting a good enough portion of permutations may increase performance in some cases (we will see, that this is not the case).

Considering these new constraints we have a total of 6 different optimization problems which based on previous thoughts and Theorem 1 all have the same optimal value:

(P1): Problem (P1*) with constraints (C1) and (C2)

(P1s): Problem (P1*) with constraints (C1), (C2), and (C3)

(P1w): Problem (P1*) with constraints (C1), (C2), and (C4)

(P2): Problem (P2*) with constraints (C1) and (C2)

(P2s): Problem (P2*) with constraints (C1), (C2), and (C3)

(P2w): Problem (P2*) with constraints (C1), (C2), and (C4)

When it comes to solving these Problems we start by focusing on the problems $P2, P2s$ and $P2w$. The linearity allows for easier solving then the quadratic problems $P1, P1s$ or $P1w$. To revisit the procedure we start by looking at linear optimization problems with real valued variables.

Definition 10. *An linear optimization problem (LOP) in standard form is given by*

$$\begin{aligned} \text{Max} \quad & c^T x \\ s.t. \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

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

Such LOP can be solved using the simplex method. When working with arbitrary LOPs that are not in standard form we can convert them into such:

1. If we have a minimization problem it can be converted to a maximization by going

from c to $-c$:

$$\min\{c^T x \mid Ax = b, x \geq 0\} = -\max\{-c^T x \mid Ax = b, x \geq 0\}$$
$$\operatorname{argmin}\{c^T x \mid Ax = b, x \geq 0\} = \operatorname{argmax}\{-c^T x \mid Ax = b, x \geq 0\}$$

2. If we have an inequality $A_i^T x \leq b_i$ they can be converted into an equality by adding so called „slack“ variables and considering $A_i^T x + s \leq b_i$ with $s \geq 0$ instead. For the case of $A_i^T x \leq b_i$ we use $A_i^T x - s \leq b_i$ with $s \geq 0$.

3. If we have constraints $x \geq c$ we can shift $x' = x - c$ to get $x' \geq 0$ and have to shift it back after solving. Similar can be done if $x \leq c$ by taking $x' = -x + c$.

4. If we have free variables $x \in \mathbb{R}$ they can be represented by $x = x' - x''$ with $x', x'' \geq 0$.

5. If we have completely bounded variables $c_1 \leq x \leq c_2$ we write $x' = x - c_1 \geq 0$ and $x'' = (c_2 - c_1) - x' \geq 0$.

To work with integer linear optimization problems (ILP) where some (or all) variables have the restriction to be integer a widely used algorithm to solve such is the branch-and-cut algorithm. It is a combination of branch-and-bound algorithm and cutting-plane algorithm.

Both algorithms are best explained on an example[6]:

Example 2 (Branch-and-Bound).
Consider the following ILP:

$$\begin{aligned} \text{Max} \quad & z = 5x_1 + 6x_2 \\ \text{s.t.} \quad & x_1 + x_2 \leq 5 \\ & 4x_1 + 7x_2 \leq 28 \\ & x_1, x_2 \geq 0 \\ & x_1, x_2 \in \mathbb{Z} \end{aligned}$$

Branch-and-Bound works by relaxing the system by its integer constraints to find bounds and allow for creating new constraints for subproblems.

We therefore start by solving the problem without restricting $x_1, x_2 \in \mathbb{Z}$ with the

simplex method. This yields an optimal value of $\frac{83}{3} \approx 27.7$ at $x_1 = \frac{7}{3} \approx 2.3$ and $x_2 = \frac{8}{3} \approx 2.7$.

This already gives us an upper bound of $\frac{83}{3}$ for the ILP. The idea of branch-and-bound is to split our problem and add two cuts $x_2 \geq 3$ or $x_2 \leq 2$, since $x_1 = \frac{7}{3} \notin \mathbb{Z}$ (alternatively one could have created cuts $x_1 \geq 3$ and $x_1 \leq 2$, which would yield to the same final result, but does not cover all cases that are needed for a good example). When solving the first subproblem

$$\begin{aligned} (1)\text{Max} \quad & 5x_1 + 6x_2 \\ \text{s.t.} \quad & x_1 + x_2 \leq 5 \\ & 4x_1 + 7x_2 \leq 28 \\ & x_1, x_2 \geq 0 \\ & x_2 \geq 3 \end{aligned}$$

we get an optimal solution of 26.75 at $x_1 = 1.75$ and $x_2 = 3$. Since x_1 is still not yet an integer solution we create new cuts $x_1 \geq 2$ and $x_1 \leq 1$ while still holding onto $x_2 \geq 3$. In the case of $x_1 \geq 2$ we do not have any feasible solution ($4x_1+7x_2 \geq 8+21 = 29 > 28$) therefore we only consider $x_1 \leq 1$ which yields an optimal solution of 25.57 at $x_1 = 1$ and $x_2 = 3.43$. We again add cuts $x_2 \geq 4$ and $x_2 \leq 3$ and get 24 at $x_1 = 0, x_2 = 4$ and 23 at $x_1 = 1, x_2 = 3$ respectively.

Now that this branch is exhausted we return to our first time we cuttet (at $x_2 \geq 3$ or $x_2 \leq 2$) and keep in mind that our ILP has a lower bound of 24.

We now add the cut $x_2 \leq 2$ and consider the LP

$$\begin{aligned} (1)\text{Max} \quad & 5x_1 + 6x_2 \\ \text{s.t.} \quad & x_1 + x_2 \leq 5 \\ & 4x_1 + 7x_2 \leq 28 \\ & x_1, x_2 \geq 0 \\ & x_2 \leq 2 \end{aligned}$$

which yields an optimal solution of 27 at $x_1 = 3$ and $x_2 = 2$. This increases our lower bound to 27 and also exhaust this branch.

Since we have now covered all possible integer pair x_1, x_2 in one of the final subproblems we can say that our maximum of the original ILP is at 27 with $(x_1, x_2) = (3, 2)$.

Example 3. Cutting Plane For the cutting plane algorithm we consider the same ILP as in the first example and also relax the integer restriction.

The idea relies on the resulting LP form of after the simplex iterations, not only the solution, which is why we convert our problem into standard form:

$$\begin{aligned} \text{Max} \quad & z = 5x_1 + 6x_2 \\ \text{s.t.} \quad & x_1 + x_2 + s_1 = 5 \\ & 4x_1 + 7x_2 + s_2 = 28 \\ & x_1, x_2, s_1, s_2 \geq 0 \end{aligned}$$

After performing the simplex iterations we get

$$\begin{aligned} \text{Max} \quad & z = \tfrac{83}{3} - \tfrac{11}{3}s_1 - \tfrac{1}{3}s_2 \\ \text{s.t.} \quad & x_1 + \tfrac{7}{3}s_1 - \tfrac{1}{3}s_2 = \tfrac{7}{3} \\ & x_2 - \tfrac{4}{3}s_1 + \tfrac{1}{3}x_2 = \tfrac{8}{3} \\ & x_1, x_2, s_1, s_2 \geq 0 \end{aligned}$$

We can now choose an arbitrary equation constraint with non integer right-hand side, we choose the the second constraint, and rewrite the occurring numbers (coefficients and rhs) in the form of $z + q$ where $z \in \mathbb{Z}$ and $q \in \mathbb{Q}$ with $0 \leq q < 1$:

$$x_2 + (-2 + \tfrac{2}{3})s_1 + (0 + \tfrac{1}{3})x_2 = 2 + \tfrac{2}{3} \iff x_2 - 2s_1 - 2 = \tfrac{2}{3} - \tfrac{2}{3}s_1 - \tfrac{1}{3}s_2 \leq \tfrac{2}{3}$$

Since in our ILP $x_2, s_1, s_2 \in \mathbb{Z}$ a rhs of $\frac{2}{3}$ is not possible and we can further reduce it to

$$x_2 - 2s_1 - 2 \leq 0$$

To bring this back to standard form we add a new slack variable s_3 and get the cut

$$x_2 - 2s_1 + s_3 = 2, \quad s_3 \geq 0$$

Adding this to our LP and applying the simplex iteration again we get

$$\begin{aligned} \text{Max} \quad & z = 27 - 3s_1 - s_3 \\ \text{s.t.} \quad & x_1 + 3s_1 - s_3 = 3 \\ & x_2 - 2s_1 + s_3 = 2 \\ & 2s_1 + s_2 - 3s_3 = 2 \\ & x_1, x_2, s_1, s_2, s_3 \geq 0 \end{aligned}$$

Here x_1, x_2 and s_1 form our basic variables that yield integer solutions $x_1 = 3, x_2 = 2, s_2 = 2$ and therefore an optimal value of $z = 27$

If we would have non integer basic variables we could create the next cut and do another iteration until we find the solution to our ILP.

To summarize both algorithms:

Definition 11. *Branch-and-bound*

1. *Relax integrality and solve the LP*
2. *If the LP solution is integer update best known integer solution*
3. *Otherwise choose one fractional variable $x_i = q \in \mathbb{Q}\backslash\mathbb{Z}$ and branch by creating two subproblem with the additional bounds $x_i \leq \lfloor q \rfloor$ or $x_i \geq \lceil q \rceil$*
4. *Repeat steps for each subproblem, if integer solution is found, trace back to the other subproblems*

Definition 12. *Cutting-Plane*

1. *Relax integrality and solve the LP*
2. *If the LP solution is integer stop*
3. *Otherwise choose simplex tableau row with fractional right-hand side*
4. *Create cut and add new constraint by rounding down coefficients and right-hand side*
5. *Repeat the steps until a integer solution is found*

In practice the cutting plane is strong in the first few iterations, but finding a solution by only cutting plane my take many iterations. In contrast branch-and-bound cuts are not as effective early on, which is why the branch-and-cut algorithm combines both advantages by starting with doing cuts from cutting plane and then branching when the cuts are no longer effective.

Solving quadratic problems, especially in the case of non-convex feasible sets, is generally way more complicated which is why we won't go into detail on how to solve such. Instead we will use an established solver Gurobi for comparing our models with different parameters. All experiments were conducted on a server running Ubuntu 24.04.3 LTS, equipped with an Intel Xeon Gold 5218R CPU (80 cores) and 125 GB RAM as well as an runtime limit of 60 Minutes per Model.

Examining the parameters $k = 3, \ldots, 9$ and $r = 3, \ldots, 20$ we got the following results:

The left table shows the solution to the optimization process in the case that Gurobi found such in the given time limit and the right table gives information about which model found said solution the fastest.

The results show that for higher values of k, i.e. $k \geq 7$ it turns out to be surprisingly hard to find optimal solutions for $r \geq 8$. Even after increasing the time limit to multiple hours we are still only left with lower and upper bounds for most of these cases in each of the six model.

We also see that there is not one best model to fit every parameter set. An obvious result is that the models P1w and P2w turned out to perform poorly for moderate values of k and r and most of the time did not find a optimal solution in 30 minutes where other models succeeded in only a few minutes. Similar, but inconsistent behavior can be observed when taking a closer look in some of the results. Take the cases $(k, r) = (6, 13)$

r \ k	3	4	5	6	7	8	9
3	2	2	3	2	3	2	3
4	2	2	3	3	3	3	3
5	2	2	3	2	4	2	4
6	2	1	2	2	3	2	3
7	2	0	3	3	3	3	3
8	2	1	3	3			
9	1	2	2	2			
10	0	2	2	1			
11	1	2	2	0			
12	2	2	2	1			
13	2	1	2	2			
14	2	0	2	2			
15	2	1	2	2			
16	2	2	2	2			
17	2	2	1	2			
18	2	2	0	2			
19	1	2	1	2			
20	0	1	2	2			

Table 1: Optimal Values

r \ k	3	4	5	6	7	8	9
3	P1w	P2s	P2s	P1	P1s	P2s	P2
4	P2w	P1s	P1	P1	P1	P1	P1
5	P1	P1s	P1	P1	P1	P1s	P1
6	P1	P1	P1	P1	P2	P1	P1
7	P2s	P2s	P1	P1s	P1s	P1s	P1
8	P1	P1	P1s	P1s			
9	P2	P1s	P1s	P1			
10	P2	P1s	P2s	P1			
11	P1	P1s	P1s	P1			
12	P2s	P1	P1	P1			
13	P2s	P1	P1	P1			
14	P1s	P1	P1s	P1s			
15	P1s	P1	P2s	P1s			
16	P2s	P1s	P1	P1			
17	P1s	P1s	P1	P1			
18	P1	P1s	P1	P1s			
19	P1	P1s	P1	P1s			
20	P1	P1	P1s	P1			

Table 2: Best Model

and $(k, r) = (6, 14)$ as an example. The model P1 and P1s both needed a few seconds to find a optimal solution when r was 13. Increasing its value to 14 resulted in P1 taking more than an hour while P1s still found a solution in under a minute. Conversely when looking at the pairs $(k, r) = (9, 4)$ and $(k, r) = (9, 5)$ the first pair again required a few seconds to be solved with both model P1 and P1s and for the second pair model P1 needed only 2 minutes where P1s took an hour for finding an optimal solution. The same behavior can be found when comparing P2 and P2s on different examples.

When comparing P1/P1s with P2/P2s we can see that in general its better to choose the models P1 or P1s, since most of the time it performs better than the models P2 or P2s and in in the cases where P2/P2s do perform better, the performance gain is just by a small amount. Take the case $k = 7$ and $r = 6$ for example, where the table suggests, that P2 was the best choice, but in reality the model P1 took 162 seconds when P2 took 145 seconds, what may seem like a good improve (10 percent), but can in generally be neglected, since we do not focus on increasing performance by small amounts, but finding a approach that works well in general.

So to conclude this section the results of the experiments above show that the best way

to tackle general parameters is to use the models P1 and P1s. Unfortunately for now we cannot say beforehand what of both will work better.

3.3 Some special cases

As seen in Table 1, we recognize a common pattern (periodic behavior) in the case of $k = 3$ and $k = 4$. We will now continue proving this observed pattern.

For $k = 3$ the sequence $0, 1, 2, 2, \ldots, 2, 1$ seems to repeat every 10 steps, which leads to the assumption of 10-periodic behavior.

Lemma 6. *If r is a multiple of 10, then $f(6, 3, r) = 0$ and if $r \equiv \pm 1 \mod 10$ then $f(6, 3, r) = 1$*

Proof. We are using the fact, that $f(6, 3, 10) = 0$ by giving an explicit example of a fair $([6], 3, 10)$-schedule:

$\mathcal{S}$	T1	T2	T3	T4	T5	T6
F1	1	1	1	2	2	2
F2	1	1	2	1	2	2
F3	1	2	2	2	1	1
F4	1	2	2	1	1	2
F5	1	2	1	2	2	1
F6	1	1	2	2	1	2
F7	1	2	1	2	1	2
F8	1	2	1	1	2	2
F9	1	2	2	1	2	1
F10	1	1	2	2	2	1

The given schedule satisfies $\lambda_{ij}(\mathcal{S}) = 4$ for all $i, j \in [6]$ and is therefore a fair schedule.

This allows us to use the subadditivity (Lemma 4) for $r = 10q$ with $q \in \mathbb{N}$:

$$f(6, 3, 10q) = f\left(6, 3, \sum_{j=1}^{q} 10\right) \leq \sum_{j=0}^{q} f(6, 3, 10) = 0$$

Combined with the fact, that $f(6, 3, r) \geq 0$ and $f(6, 3, 0) = 0$ by definition, this completes the first statement.

For the second part, we consider $r' = r \pm 1 = 10q$ with $q \in \mathbb{N}$. Using the first part we can see that $f(6,3,r') = 0$ and therefore, using Corollary 5, it follows that $f(6,3,r') = 1$ □

Lemma 7. *For $r \geq 0$ we have an upper bound of 2, i.e. $f(6,3,r) \leq 2$*

Proof. The case $r = 0, 1$ are again trivial, so assume $r \geq 2$:
There exists $([6],3,p)$-schedules $\mathcal{S}$ for $2 \leq p \leq 8$ with $\Delta(\mathcal{S}) = 2$:

$\mathcal{S}_2$	T1	T2	T3	T4	T5	T6
F1	1	1	1	2	2	2
F2	1	1	1	2	2	2

$\mathcal{S}_3$	T1	T2	T3	T4	T5	T6
F1	1	1	1	2	2	2
F2	1	2	2	2	1	1
F3	1	1	2	2	2	1

$\mathcal{S}_4$	T1	T2	T3	T4	T5	T6
F1	1	1	1	2	2	2
F2	1	2	1	2	1	2
F3	1	1	2	2	2	1
F4	1	2	2	2	1	1

$\mathcal{S}_5$	T1	T2	T3	T4	T5	T6
F1	1	1	1	2	2	2
F2	1	2	1	2	1	2
F3	1	1	2	1	2	2
F4	1	2	1	2	2	1
F5	1	1	2	2	2	1

$\mathcal{S}_6$	T1	T2	T3	T4	T5	T6
F1	1	1	1	2	2	2
F2	1	1	2	2	2	1
F3	1	2	2	1	1	2
F4	1	2	1	2	2	1
F5	1	2	1	1	2	2
F6	1	2	1	2	1	2

$\mathcal{S}_7$	T1	T2	T3	T4	T5	T6
F1	1	1	1	2	2	2
F2	1	2	1	1	2	2
F3	1	1	2	2	1	2
F4	1	1	2	2	2	1
F5	1	1	2	1	2	2
F6	1	2	1	2	2	1
F7	1	2	2	2	1	1

$\mathcal{S}_8$	T1	T2	T3	T4	T5	T6
F1	1	1	1	2	2	2
F2	1	2	1	2	1	2
F3	1	1	2	2	1	2
F4	1	1	2	2	2	1
F5	1	2	1	2	2	1
F6	1	2	1	2	1	2
F7	1	1	2	1	2	2
F8	1	2	2	1	1	2

Let $r = 10q + p$ for $2 \leq p \leq 8$ and use the contraction property (Corollary 4), we get:

$$f(6, 3, 10q + p) \leq f(6, 3, 10q) + f(6, 3, p) \leq 0 + 2 = 2$$

For the remaining case of $r = 10q$ or $r = 10q \pm 1$, the Lemma 6 gives $f(6, 3, r) \leq 2$. □

The last simple case is $r \equiv 5 \mod 10$:

Lemma 8. *If* $r \equiv 5 \mod 10$ *then* $f(6, 3, r) = 2$.

Proof. From Lemma 7 we have $f(6, 3, r) \leq 2$.
When $r \equiv 5 \mod 10$ then $r \equiv 0 \mod 5$ and therefore $2r \equiv 0 \mod 5$. This allows us to use Corollary 3 for $f(6, 3, r) \neq 1$ and Theorem 2 for $f(6, 3, r) \neq 0$, since k is odd and $r \cdot (k - 1) = 2k$ is divisible by $n - 1 = 5$. Leaving us with $f(6, 3, r) \geq 2$ and therefore $f(6, 3, r) = 0$. □

Lemma 9. *For* $r = 10q + p$, *where* $2 \leq p \leq 8$, *there exists no fair* $([n], k, r)$*-schedule, i.e.* $f(6, 3, r) \neq 0$.
If $p = 5$ *there also does not exists a almost fair* $([n], k, r)$*-schedule, leaving* $(6, 3, 10q + 5) = 2$.

Proof. When $p = 5$, i.e. $r \equiv 5 \mod 10$ then $r \equiv 0 \mod 5$ and therefore $2r \equiv 0 \mod 5$. This allows us to use Corollary 3 for $f(6, 3, r) \neq 1$ and Theorem 2 for $f(6, 3, r) \neq 0$, since k is odd and $r \cdot (k-1) = 2k$ is divisible by $n-1 = 5$. Leaving us with $f(6, 3, r) \geq 2$ and therefore $f(6, 3, r) = 0$.

If $p \in \{2, 3, 4, 6, 7, 8\}$ we have $p \mod 5 \in \{1, 2, 3, 4\}$, which means

$$(k - 1) \cdot r = 2r = 20q + 2p \equiv 2p \mod 5 \implies (k - 1) \cdot r \mod (n - 1) \in \{2, 4, 1, 3\}$$

Using Corollary 2 this prohibits the existence of a fair $([n], k, r)$-schedule. □

This leaves us with a current characterization of:

$$f(6,3,r) = \begin{cases} 0, & r \equiv 0 \mod 10 \\ 1, & r \equiv \pm 1 \mod 10 \\ 1 \text{ or } 2, & \text{otherwise} \end{cases}$$

For the remaining cases we will need the helper function used in the proof of Theorem 2, it's recommended to revisit the proof to remind of the idea behind it.

Definition 13. *Let $\mathcal{S} = (A_{ls})$ be a schedule, $s \in [t]$, $i \in [n]$ and $1 \leq a \leq b \leq r$, the helper function g is defined as*

$$g_s^i(a,b) := \left|\{l \,|\, a \leq l \leq b, i \in A_{ls}\}\right| = \sum_{l=a}^{b} 1_{A_{ls}}(i) = \sum_{l=a}^{b} 1_{\{s\}}(S_{li})$$

Lemma 10. *Let g be defined as in Definition 13, then:*

$$\sum_{s=1}^{t} g_s^i(a,b) = b - a + 1$$

Proof. Using the Definition of S_{li} we get

$$\sum_{s=1}^{t} g_s^i(a,b) = \sum_{s=1}^{t}\sum_{l=a}^{b} 1_{\{s\}}(S_{li}) = \sum_{l=a}^{b}\sum_{s=1}^{t} 1_{\{s\}}(S_{li}) = \sum_{l=a}^{b} 1 = b - a + 1$$

□

Theorem 3. *If $r \equiv 2 \mod 5$ there exists no almost fair $([6], 3, r)$-schedule and therefore $f(6,3,r) = 2$ for $r \mod 10 \in \{2, 7\}$.*

Proof. Assume there exists an almost fair $([6], 3, r)$-schedule. From $r \equiv 2 \mod 5$ it follows that $2r \equiv 4 \mod 5$ and Corollary 3 gives

$$\left|\left\{j \in N \backslash \{i\} : \lambda_{ij} = q + 1\right\}\right| = 4 \quad \text{and} \quad \left|\left\{j \in N \backslash \{i\} : \lambda_{ij} = q\right\}\right| = 1.$$

This leaves every node in the q-co-occurrence graph with degree 1.
Up to permutations, which do not impact f by Corollary 1, the only way to achieve

this is by

$$\lambda_{12} = \lambda_{21} = \lambda_{34} = \lambda_{43} = \lambda_{56} = \lambda_{65} = q$$

and $\lambda_{ij} = q + 1$ otherwise.
Using Corollary 1 again, we set $S_{l1} = 1$ and

$$S_{l3} = \begin{cases} 1, & 1 \leq l \leq q+1 \\ 2, & q+2 \leq l \leq r \end{cases}.$$

This ensures, that 1 and 3 are in exactly $q + 1$ blocks together which was given by $\lambda_{13} = q + 1$.
We now define $\mu := g_1^2(1, q+1)$ as the number of times 2 is in the first block of the first $q + 1$ assignments. Using Lemma 1 to represent λ_{12} and λ_{13} gives

$$\begin{aligned} \lambda_{12} &= \sum_{l=1}^{r} 1_{\{S_{l1}\}}(S_{l2}) \\ &= \sum_{l=1}^{q+1} 1_{\{S_{l1}\}}(S_{l2}) + \sum_{l=q+2}^{r} 1_{\{S_{l1}\}}(S_{l2}) \\ &= \sum_{l=1}^{q+1} 1_{\{1\}}^{S_{l2}} + \sum_{l=q+2}^{r} 1_{\{1\}}^{S_{l2}} \\ &= g_1^2(1, q+1) + g_1^2(q+2, r) \end{aligned}$$

$\implies g_1^2(q+2, r) = q - \mu$
and

$$\begin{aligned} \lambda_{32} &= \sum_{l=1}^{r} 1_{\{S_{l3}\}}(S_{l2}) \\ &= \sum_{l=1}^{q+1} 1_{\{S_{l3}\}}(S_{l2}) + \sum_{l=q+2}^{r} 1_{\{S_{l3}\}}(S_{l2}) \\ &= \sum_{l=1}^{q+1} 1_{\{1\}}^{S_{l2}} + \sum_{l=q+2}^{r} 1_{\{2\}}^{S_{l2}} \\ &= g_1^2(1, q+1) + g_2^2(q+2, r) \end{aligned}$$

$\implies g_2^2(q+2, r) = q + 1 - \mu$

Lemma 10 gives $g_1^2(q+2,r) + g_2^2(q+2,r) = r - (q+2) + 1$ and therefore

$$\big(q-\mu\big) + \big(q+1-\mu\big) = r - (q+2) + 1 \implies \mu = \frac{3q+2-r}{2}$$

Since $2r = 5q + 4$ this gives us:

$$q = \frac{2r-4}{5} \implies \mu = \frac{3 \cdot \frac{2r-4}{5} + 2 - r}{2} = \frac{r-2}{10} \tag{2}$$

As q has to be a natural number we can conclude $r \equiv 2 \mod 10$, which is a contradiction for $r \mod 10 = 7$. [2]

From $\mu := q_1^2(1, q+1)$ we may assume w.l.o.g. that $A_{l1} \subset \{1,2,3\}$ for $1 \leq l \leq \mu$. Note that $|A_{ls}| = k = 3$ which implies equality.

Similar as before we define $\nu := g_1^4(\mu+1, q+1)$ and get $g_2^4(q+2,r) = q - \nu$ for securing $\lambda_{34} = q$, repeating the same as above we get $\nu = \frac{r-2}{10}$ by $\lambda_{14} = q+1$.

Introducing another variable

$$b := \sum_{l=q+2}^{r} 1_{\{S_{l2}\}}(S_{l4})$$

gives us a full characterization of 4:

$$\begin{aligned}
q+1 &= \lambda_{24} \\
&= \sum_{l=1}^{r} 1_{\{S_{l2}\}}(S_{l4}) \\
&= \sum_{l=1}^{\mu} 1_{\{S_{l2}\}}(S_{l4}) + \sum_{l=\mu}^{q+1} 1_{\{S_{l2}\}}(S_{l4}) + \sum_{l=q+2}^{r} 1_{\{S_{l2}\}}(S_{l4}) \\
&= \sum_{l=1}^{\mu} 1_{\{1\}}(S_{l4}) + \sum_{l=\mu+1}^{q+1} 1_{\{2\}}(S_{l4}) + \sum_{l=q+2}^{r} 1_{\{S_{l2}\}}(S_{l4}) \\
&= 0 + g_2^4(\mu+1, q+1) + b \\
&= (q+1-(\mu+1)-1) - g_1^4(\mu+1, q+1) + b \\
&= q+1-\mu-\nu+b
\end{aligned}$$

$$\implies b = \mu + \nu = 2\mu$$

[2] Note that rearranging to $\mu = q/4$, which forces q to be a multiple of 4 may seem like a stronger statement, but would just change the problem to cases of $r = 40q + p$.

We can further analyze by writing

$$\alpha = \sum_{l=q+2}^{r} 1_{\{1\}}(\{S_{l2}\}) \cdot 1_{\{1\}}(S_{l4}), \quad \beta = \sum_{l=q+2}^{r} 1_{\{2\}}(\{S_{l2}\}) \cdot 1_{\{2\}}(S_{l4})$$

as the amount of times that we have the blocks $\{1, 2, 4\}$ and $\{3, 2, 4\}$ in the arrangements from $q + 2$ to r.
This allows for:

$$\begin{aligned}
\lambda_{12} &= \sum_{l=1}^{q+1} 1_{\{S_{l1}\}}(S_{l2}) + \sum_{l=q+2}^{r} 1_{\{S_{l1}\}}(S_{l2}) \\
&= \mu + \sum_{l=q+2}^{r} 1_{\{1\}}(S_{l2}) \\
&= \mu + \sum_{l=q+2}^{r} 1_{\{1\}}(S_{l2}) \cdot \big(1_{\{1\}}(S_{l4}) + 1_{\{2\}}(S_{l4})\big) \\
&= \mu + \alpha + \sum_{l=q+2}^{r} 1_{\{1\}}(S_{l2}) \cdot 1_{\{2\}}(S_{l4})
\end{aligned}$$

and analogously

$$\begin{aligned}
\lambda_{34} &= \sum_{l=1}^{q+1} 1_{\{S_{l3}\}}(S_{l4}) + \sum_{l=q+2}^{r} 1_{\{S_{l3}\}}(S_{l4}) \\
&= \nu + \beta + \sum_{l=q+2}^{r} 1_{\{1\}}(S_{l2}) \cdot 1_{\{2\}}(S_{l4})
\end{aligned}$$

Using $\lambda_{12} = \lambda_{34}$ and $\mu = \nu$ then gives $\alpha = \beta$, the definition of b grants $b = \alpha + \beta$ and therefore $\alpha = \beta = b/2$.
This does not directly results in a contradiction, but since $k = 3$ this fills enough blocks, such that there can't exists placements for 5 and 6, which won't break $\Delta(\mathcal{S}) = 1$. To be specific this leaves us with $q + 1 - b$ incomplete arrangements in the first $q + 1$ and $r - (q + 1) - b$ in the remaining. Let

$$\eta = \sum_{l=1}^{q+1} 1_{\{1\}}(\{S_{l5}\}) \cdot 1_{\{2\}}(\{S_{l6}\}), \quad \eta' = \sum_{l=q+2}^{r} 1_{\{1\}}(\{S_{l5}\}) \cdot 1_{\{2\}}(\{S_{l6}\})$$

describe the amount of times a 5 is in the first block of such incomplete arrangements, split between the $(q + 1)$-th and $(q + 2)$-th arrangement for easier analysis ofλ_{15} and

λ_{35}.

When considering $\lambda_{15} = \lambda_{35} = q + 1$ we get

$$\begin{aligned}
\lambda_{15} &= \sum_{l=1}^{r} 1_{\{S_{l1}\}}(\{S_{l5}\}) \\
&= \sum_{l=1}^{\mu+\nu} 1_{\{S_{l1}\}}(\{S_{l5}\}) + \sum_{l=\mu+\nu+1}^{q+1} 1_{\{S_{l1}\}}(\{S_{l5}\}) + \sum_{l=q+2}^{r} 1_{\{S_{l1}\}}(\{S_{l5}\}) \\
&= 0 + \eta + \sum_{l=q+2}^{r} 1_{\{1\}}(\{S_{l5}\}) \cdot \big(1_{\{1\}}(S_{l6}) + 1_{\{2\}}(S_{l6})\big) \\
&= \eta + \eta' + \sum_{l=q+2}^{r} 1_{\{1\}}(\{S_{l5}\}) 1_{\{1\}}(S_{l6}) \\
&= \eta + \eta' + \sum_{l=q+2}^{r} 1_{\{2\}}(\{S_{l2}\}) 1_{\{2\}}(S_{l4}) \\
&= \eta + \eta' + \beta
\end{aligned}$$

and analogously

$$\lambda_{35} = \eta + \alpha + \big(r - (q+1) - b\big) - \eta'$$

Using $\lambda_{15} = \lambda_{35}$ and $\alpha = \beta$ gives:

$$0 = 2\eta' - r + q - 1 + b \implies \eta' = \frac{r - (q+1) - b}{2}$$

Since $r \equiv 2 \mod 10$ by assumption we know $2r - 4 \equiv 0 \mod 10$ and therefore $q = \frac{2r-4}{5}$ has to be even, but r and $b = 2\mu$ are also both even making $r - (q+1) - b$ odd which contradicts $\eta' \in \mathbb{N}$. □

Using the same procedure one is able to cover the cases $r \equiv 3 \mod 5$, since using Corollary 3 yields

$$\big|\{j \in N \backslash \{i\} : \lambda_{ij} = q + 1\}\big| = 1 \quad \text{and} \quad \big|\{j \in N \backslash \{i\} : \lambda_{ij} = q\}\big| = 4.$$

Corollary 6. *If $r \equiv 3 \mod 5$ there exists no almost fair $([6], 3, r)$-schedule and therefore $f(6, 3, r) = 2$ for $r \mod 10 \in \{3, 8\}$.*

The last cases missing to complete a characterization of $f(6,3,r)$ are $r \equiv 4 \mod 10$ and $r \equiv 6 \mod 10$ which makes the analysis more complex: By using Corollary 3 we get

$$\left|\left\{j \in N\backslash\{i\} : \lambda_{ij} = q+1\right\}\right| = 2 \quad \text{and} \quad \left|\left\{j \in N\backslash\{i\} : \lambda_{ij} = q\right\}\right| = 3.$$

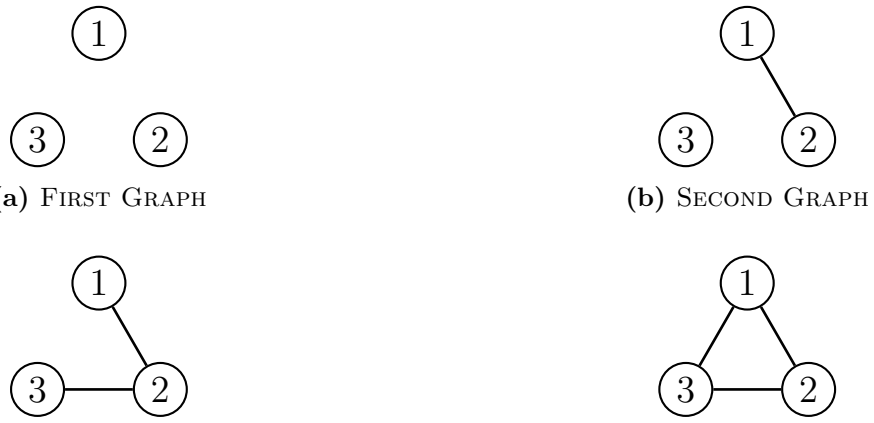
or

$$\left|\left\{j \in N\backslash\{i\} : \lambda_{ij} = q+1\right\}\right| = 3 \quad \text{and} \quad \left|\left\{j \in N\backslash\{i\} : \lambda_{ij} = q\right\}\right| = 2.$$

respectively, allowing for multiple possible co-occurrence graphs, that are not isomorph. We start by finding them all:

Lemma 11. *Let $\tilde{G}$ be a finite, undirected, simple and 2-regular (i.e. all nodes have degree 2) graph, then all components of $\tilde{G}$ have to be cyclic.*

Proof. Let $G = (V, E)$ be a component of G. Since $\tilde{G}$ is 2-regular the component is 2-regular too.
We do an induction over $n = |V|$:
There are no 2-regular graphs with less than 3 nodes, since the degree of a node $v \in V$ is bounded by $\deg v \leq n-1$. Up to permutations, there are a total of 4 different graphs with $n = 3$ nodes:



(a) First Graph **(b)** Second Graph **(c)** Third Graph **(d)** Fourth Graph

Figure 3: All graphs with 3 nodes.

Only the fourth graph is 2-regular, it's also cyclic proving the base step.

For the induction step, let $G = (V, E)$ be a 2-regular, connected graph with $V = \{v_1, \ldots, v_{n+1}\}$. Since $\deg v_{n+1} = 2$ there exists nodes v_l and $v_{l'}$, such that $l' > l$ and

$\{v_l, v_{n+1}\}, \{v_{l'}, v_{n+1}\} \in E$. This allows to construct a new graph $G' = (V', E')$ with $V' = V \backslash \{v_{n+1}\}$ and

$$E' = \big\{\{v_i, v_j\} \in E \,|\, i, j \in [n]\big\} \cup \big\{\{v_l, v_{l'}\}\big\}$$

by simply removing v_{n+1} and reconnecting v_l and $v_{l'}$. We can assure that $\{v_l, v_{l'}\} \notin E$ for $n > 3$, because otherwise $\{v_l, v_{l'}, v_{n+1}\}$ would form a component in G, which contradicts its connectivity.
We can conclude that G' is also connected and 2-regular and since $|V'| = n$ using the induction hypothesis it follows that G' is cyclic, i.e. there exists a permutation $\pi : [n] \to [n]$ with

$$\big\{\{v_{\pi(1)}, v_{\pi(2)}\}, \{v_{\pi(2)}, v_{\pi(3)}\}, \ldots, \{v_{\pi(n-1)}, v_{\pi(n)}\}, \{v_{\pi(n)}, v_{\pi(1)}\}\big\} = E'.$$

Since $\{v_l, v_{l'}\} \in E'$ and G' is 2-regular, on of these edges must be $\{v_l, v_{l'}\}$, because otherwise π wouldn't be a permutation.
This allows constructing a permutation $\sigma : [n+1] \to [n+1]$ on G:
w.l.o.g. $\pi(1) = l'$ and $\pi(n) = l$. Define

$$\sigma(i) = \begin{cases} \pi(i), & 1 \le i \le n \\ n+1, & i = n+1 \end{cases}$$

This results in

$$\begin{aligned} &\big\{\{v_{\sigma(1)}, v_{\sigma(2)}\}, \{v_{\sigma(2)}, v_{\sigma(3)}\}, \ldots, \{v_{\sigma(n-1)}, v_{\sigma(n)}\}, \{v_{\sigma(n)}, v_{\sigma(n+1)}\}, \{v_{\sigma(n+1)}, v_{\sigma(1)}\}\big\} \\ = &\big\{\{v_{\pi(1)}, v_{\pi(2)}\}, \{v_{\pi(2)}, v_{\pi(3)}\}, \ldots, \{v_{\pi(n-1)}, v_{\pi(n)}\}, \{v_l, v_{n+1}\}, \{v_{n+1}, v_{l'}\}\big\} = E. \end{aligned}$$

proving that G is cyclic. □

Corollary 7. *For a finite, undirected, simple and* 2*-regular graph with exactly* 6 *nodes, there are only two possible graphs up to isomorphism, a cyclic or the union of two* K_3 *graphs*[3]*.*

Proof. Using Lemma 11 we can partition G into its cyclic components, leaving us with 3 different cases:

1. There is only one component, which implies G is cyclic.

[3] The graph K_n is a complete Graph with $|V| = n$ Nodes, i.e. $E = \binom{V}{2}$.

2. There are two components of size 3, which are both cyclic and therefore equal K_3
3. There is at least one component that doesn't have have 3 or 6 nodes. This would result in an component of size 1 or 2, which contradicts that G was 2-regular.

□

Knowing the possible types of co-occurrence graphs we can now dedicate ourself to the cases $r \equiv 4 \mod 10$ and $r \equiv 6 \mod 10$:

Theorem 4. *If $r \equiv 4 \mod 10$ there exists no almost fair $([6], 3, r)$-schedule and therefore $f(6, 3, r) = 2$.*

Proof. Assuming the existence of a $([6], 3, r)$-schedule $\mathcal{S}$ with $\Delta(\mathcal{S}) = 1$ we can use Lemma 3 and get

$$\left|\left\{j \in N \backslash \{i\} : \lambda_{ij} = q + 1\right\}\right| = 2 \quad \text{and} \quad \left|\left\{j \in N \backslash \{i\} : \lambda_{ij} = q\right\}\right| = 3.$$

since $r \equiv 4 \mod 10$ implies $2r = 5q + 3$.
Looking at the (q)-co-occurrence graph G_q of $\mathcal{S}$ we observe that it's 2-regular allowing the usage of Corollary 7, i.e. G_q is either a cycle or the union of two K_3 graphs.

In the case of two K_3 graphs we can w.l.o.g. assume that

$$\begin{aligned}
&\lambda_{12} = \lambda_{21} = \lambda_{13} = \lambda_{31} = \lambda_{23} = \lambda_{32} = q \\
&\lambda_{45} = \lambda_{54} = \lambda_{46} = \lambda_{64} = \lambda_{56} = \lambda_{65} = q \\
&\lambda_{ij} = q + 1 \quad \text{otherwise}
\end{aligned}$$

If we have such schedule we are able to construct a new $([6], 3, r + 1)$-schedule $\mathcal{S}'$ by adding $(\{1, 2, 3\}, \{4, 5, 6\})$ as $(r + 1)$-th arrangement.
Now we have $\lambda_{ij} = q + 1$ for all $(i, j) \in N_*^2$ resulting in $f(6, 3, r + 1) = 0$. But since $r + 1 \equiv 5 \mod 10$ this is a contradiction to Lemma 8 allowing us to focus on a cycle as (q)-co-occurrence graph.

The proof then follows analogously to the one of Theorem 3 by assuming

$$\lambda_{12} = \lambda_{23} = \lambda_{34} = \lambda_{45} = \lambda_{56} = \lambda_{61} = q$$
$$\lambda_{21} = \lambda_{32} = \lambda_{43} = \lambda_{54} = \lambda_{65} = \lambda_{16} = q$$
$$\lambda_{ij} = q + 1 \quad \text{otherwise}$$

We then set $S_{l1} = 1$ and

$$S_{l3} = \begin{cases} 1, & 1 \leq l \leq q+1 \\ 2, & q+2 \leq l \leq r \end{cases}.$$

writing $\mu = g_1^2(1, q+1)$ and $\nu = g_1^4(\mu+1, q+1)$ and following from similar calculation as before that

$$\mu = \frac{3q - r + 1}{2}, \quad \text{and} \nu = \frac{3q - r + 2}{2}$$

Since both μ and ν are natural numbers this implies $3q - r + 1$ and $3q - r + 2$ both being even which is impossible. □

The last case is again analogously proven and only really differs in the reasoning why two K_3 graphs are not viable co-occurrence graphs.

Corollary 8. *If* $r \equiv 6 \mod 10$ *there exists no almost fair* $([6], 3, r)$*-schedule and therefore* $f(6, 3, r) = 2$.

Proof. From Lemma 3 we can again conclude a 2-regular graph as $(q+1)$-co-occurrence graph, when assuming an almost fair $([6], 3, r)$-schedule $\mathcal{S}$.
By looking at the case of two K_3 components of this graph we have

$$\lambda_{12} = \lambda_{23} = \lambda_{34} = \lambda_{45} = \lambda_{56} = \lambda_{61} = q + 1$$
$$\lambda_{21} = \lambda_{32} = \lambda_{43} = \lambda_{54} = \lambda_{65} = \lambda_{16} = q + 1$$
$$\lambda_{ij} = q \quad \text{otherwise}$$

Taking an arbitrary fair $([6], 3, 10)$-schedule $\mathcal{S}' = (A'_{ls})$ we find a permutation $\pi : [6] \to [6]$ such that $\pi(A_{10,1}) = \{1, 2, 3\}$ and $\pi(A_{10,1}) = \{4, 5, 6\}$. From Theorem 1 this does not change the values of λ_{ij} and therefore the $([6], 3, 9)$-schedule $\mathcal{S}'' = (\pi(A_{ls})')_{l=1..9, s=1,2}$

has the following co-occurrence numbers:

$$\lambda_{12} = \lambda_{23} = \lambda_{34} = \lambda_{45} = \lambda_{56} = \lambda_{61} = q$$
$$\lambda_{21} = \lambda_{32} = \lambda_{43} = \lambda_{54} = \lambda_{65} = \lambda_{16} = q$$
$$\lambda_{ij} = q + 1 \quad \text{otherwise.}$$

When concatenating $\mathcal{S}$ and $\mathcal{S}''$ we get a $([6], 3, r+9)$-schedule with constant co-occurrence number $2q+1$. This would result in $f(6, 3, r+9) = 0$ which contradicts Lemma 8 since $r + 9 \equiv 5 \mod 10$.
Now focusing on a cyclic G_{q+1} works as before in Theorem3. □

Combining the now proven Characteristics for each case of $r \mod 10$ we can summarize it in an final Corollary:

Corollary 9. *For an arbitrary $r \in \mathbb{N}$ the following characterization of $f(6, 3, \cdot)$ holds:*

$$f(6,3,r) = \begin{cases} 0, & r \equiv 0 \mod 10 \\ 1, & r \equiv \pm 1 \mod 10 \\ 2, & \textit{otherwise} \end{cases}$$

It is noteworthy that we can proof a similar result for the special case of $k = 4$ and $n = 8$:

Theorem 5. *For an arbitrary $r \in \mathbb{N}$ the following characterization of $f(6, 3, \cdot)$ holds:*

$$f(8,4,r) = \begin{cases} 0, & r \equiv 0 \mod 7 \\ 1, & r \equiv \pm 1 \mod 7 \\ 2, & \textit{otherwise} \end{cases}$$

This theorem is stated without giving full proof, since the deviation would be significantly longer than for the case of $k = 3$ without showing any new methods of proving or other interesting properties.

The idea follows the same techniques as used for proving the above case. By providing example schedules for schedules fulfilling our hypothesis when $r \leq 7$ we can ensure an upper bound of 2. Following Corollary 3 we can further give an lower bound of 1

for the case of r being no multiple of 7. For proving the characterization we are left with contradicting the existence of a almost fair schedules in the cases of $r \mod 7 \in \{2, 3, 4, 5\}$ which would give $p \in \{6, 2, 5, 1\}$.

Contradicting such existence works similar to Theorem 4 and Theorem 3, while keeping the simplicity of these proofs the case $k = 4$ turns out to be much longer, especially for the cases of $p = 2$ and $p = 5$ since we have to cover multiple different co-occurrence-graphs where before we were able to break it down to cover only one of such.

Conjecture 1. *It remains unclear if such periodic behavior can be generalized for the cases of $k > 4$ and $n = 2k$ or for n being another multiple of k than $2k$.*

Our method is limited to $k \leq 4$, because after that we have to contradict the existence of schedules with a fairness deviation of 2 or more, which, sheer to the amount of potential co-occurrence-graphs, is not possible with our approach neither by hand nor using an computer.

Despite this rather demotivating limitation we can still use the proofs ideas to give lower bounds to parameters too big for our computers to find solution using the presented optimization formulations from Section 3.2.

One such case that we wish to highlight is $(n, k, r) = (18, 9, 15)$. This seemingly arbitrary parameters have an actual application in sailing league problems. The polish sailing league normally consists of four rounds, each containing a number of flights. A flight splits the 18 competing teams into two smaller heats. The number of flights per round is not always the same as weather conditions may influence the total amount of races possible, but it is generally aimed to have 15 flights. Scheduling such 15 flights leaves us with a setup of $n = 18, k = 9$ and $r = 15$.

To analyze the current situation we can look at an example of their schedule. We are therefore inspecting the fourth round of the 2021 Ekstraklasa[8] which consisted of exactly 15 flights. We can construct the corresponding schedule table to analyze the fairness deviation of given round:

$\mathcal{S}$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
F1	1	1	2	1	1	2	2	1	1	1	1	2	2	2	2	2	2	1
F2	2	2	1	2	2	1	2	2	1	1	2	1	1	1	2	1	2	1
F3	1	1	2	2	1	1	2	1	2	2	1	2	1	1	2	1	2	2
F4	2	2	1	1	1	2	1	1	1	2	1	2	2	1	2	2	2	1
F5	2	2	2	1	1	1	2	1	2	1	1	2	1	2	2	2	1	1
F6	1	1	1	2	2	1	1	2	2	1	2	2	2	1	2	2	1	1
F7	2	1	1	1	2	2	2	1	1	1	1	2	2	2	2	1	1	2
F8	2	2	1	2	2	1	1	2	2	2	2	1	1	1	1	1	1	2
F9	1	2	2	1	2	1	2	1	1	2	1	1	2	1	2	2	1	2
F10	2	2	1	2	2	1	1	1	2	1	1	1	2	2	1	1	2	2
F11	2	1	1	1	1	1	1	2	1	2	2	1	1	2	2	2	2	2
F12	2	2	2	2	1	1	1	2	1	1	1	2	2	1	1	2	1	2
F13	2	1	2	1	1	1	2	1	2	1	2	1	2	1	1	2	2	2
F14	1	2	1	1	2	1	2	1	1	1	2	2	1	2	1	2	2	2
F15	2	2	1	2	2	1	1	2	2	2	2	1	1	1	1	1	1	2

For simplicity we have ordered the team labels based on the result, i.e. team number 1 corresponds to „Yacht Klub Polski Gdynia“ which placed first, while team 18 represents „Yacht Club Sopot“ which placed last. Analyzing the co-occurrence numbers of this schedule yields the following table:

λ_{ij}	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	-	10	5	8	7	6	5	9	8	7	8	5	7	7	6	6	7	9
2	10	-	6	9	10	5	6	8	7	8	7	6	6	6	5	7	6	8
3	5	6	-	6	3	8	11	5	8	7	4	9	9	7	8	10	7	7
4	8	9	6	-	10	5	4	12	11	8	9	6	6	4	5	3	6	8
5	7	10	3	10	-	6	7	9	8	7	10	5	7	7	6	4	5	9
6	6	5	8	5	6	-	8	6	5	8	5	10	10	10	9	7	8	4
7	5	6	11	4	7	8	-	3	6	5	6	9	7	9	10	8	9	7
8	9	8	5	12	9	6	3	-	8	9	12	5	5	5	6	6	5	7
9	8	7	8	11	8	5	6	8	-	8	9	6	6	6	5	5	6	8
10	7	8	7	8	7	8	5	9	8	-	8	5	5	5	8	6	7	9
11	8	7	4	9	10	5	6	12	9	8	-	4	4	6	5	7	8	8
12	5	6	9	6	5	10	9	5	6	5	4	-	9	9	10	10	7	5
13	7	6	9	6	7	10	7	5	6	5	4	9	-	7	8	10	7	7
14	7	6	7	4	7	10	9	5	6	5	6	9	7	-	8	8	9	7
15	6	5	8	5	6	9	10	6	5	8	5	10	8	8	-	9	8	4
16	6	7	10	3	4	7	8	6	5	6	7	10	10	8	9	-	8	6
17	7	6	7	6	5	8	9	5	6	7	8	7	7	9	8	8	-	7
18	9	8	7	8	9	4	7	7	8	9	8	5	7	7	4	6	7	-

We can therefore extract that $\lambda^+(\mathcal{S}) = 12$ and $\lambda^-(\mathcal{S}) = 3$, giving $\Delta(\mathcal{S}) = 9$

From our optimization part we know that these parameters are too big to expect finding an optimal schedule using the formulated programs. We still aim to improve the presented schedule.

Using the presented branch-and-cut algorithm one is able to find a $(18, 9, 15)$-schedule with fairness deviation of 4 creating an upper bound $f(18, 9, 15) \leq 4$.

Using the created technique from above we are also able to find a lower bound of 2.

Theorem 6. *For the fairness deviation of the parameters* $(n, k, r) = (18, 9, 15)$ *the following bounds hold:*

$$2 \leq f(18, 9, 15) \leq 4$$

Proof. We start by presenting an example of a $(18, 9, 15)$-schedule with fairness deviation of 4 to proof the upper bound:

$\mathcal{S}$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
F1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2
F2	1	2	1	1	2	2	2	2	1	1	1	1	1	2	2	1	2	2
F3	1	1	1	2	1	2	2	2	2	1	2	1	1	1	2	1	2	2
F4	1	1	2	1	2	1	2	2	2	2	1	1	2	1	1	2	2	1
F5	1	1	2	1	2	2	1	1	1	2	2	1	2	1	2	2	1	2
F6	1	2	2	1	1	2	1	2	2	2	2	2	1	1	2	1	1	1
F7	1	1	2	1	1	2	2	1	2	1	2	1	2	2	1	1	2	2
F8	1	2	2	2	1	2	1	1	2	2	1	2	1	2	1	1	2	1
F9	1	2	1	2	1	1	2	2	2	2	1	2	2	1	1	1	1	2
F10	1	2	1	2	1	2	1	2	2	2	1	1	1	2	2	2	1	1
F11	1	2	1	1	2	2	2	1	2	2	2	1	2	2	1	1	1	1
F12	1	1	1	2	2	1	2	1	2	2	2	1	1	2	2	1	2	1
F13	1	1	1	1	2	2	1	2	2	1	1	2	1	1	2	2	2	2
F14	1	1	2	2	2	2	1	2	1	1	2	1	2	2	2	1	1	1
F15	1	1	1	2	2	1	2	1	1	2	1	2	2	1	2	2	1	2

Table 3: $(18, 9, 15)$-schedule with fairness deviation 4

Next, we can ensure $f(18, 9, 15) \neq 0$ because Corollary 2 requires $r \cdot (k-1) \equiv 0 \mod (n-1)$ for the existence of a fair schedule and $r \cdot (k-1) = 120$ is not divisible by $n - 1 = 17$.

For showing that no almost fair schedule exists we again assume such existence and use Corollary 3 to conclude

$$\left|\left\{j \in N \backslash \{i\} : \lambda_{ij} = 8\right\}\right| = 1 \quad \text{and} \quad \left|\left\{j \in N \backslash \{i\} : \lambda_{ij} = 7\right\}\right| = 16.$$

Up to permutation this yields

$$\lambda_{12} = \lambda_{21} = \lambda_{34} = \lambda_{43} = \cdots = \lambda_{17,18} = \lambda_{18,17} = 8$$

and $\lambda_{ij} = 7$ for the remaining pairs.

w.l.o.g. we set

$$S_{l1} = 1, \quad \text{and} \quad S_{l2} = \begin{cases} 1, & 1 \leq l \leq 8 \\ 2, & 9 \leq l \leq 15 \end{cases}$$

From $7 = \lambda_{13} = \lambda_{23}$ we get

$$7 = \lambda_{13} = g_1^3(1, 15) = g_1^3(1, 8) + g_1^3(9, 15)$$
$$7 = \lambda_{13} = g_1^3(1, 8) + g_2^3(9, 15) = g_1^3(1, 8) - g_1^3(9, 15) + 7$$

Solving this system for the unknown $g_1^3(1, 8)$ and $g_1^3(9, 15)$ yields

$$g_1^3(1, 8) = 7/2 \quad \text{and} \quad g_1^3(9, 15) = 7/2$$

From $g_1^3(1, 8) \in \mathbb{N}$ we already have a contradiction, resulting in $f(18, 9, 15) \neq 1$. □

For practical application we have to consider that while having 15 flights is the goal, this often gets shortened and the last few flights get canceled. A good schedule should therefore cover this case in a sense that the cut schedule is still optimal. A perfect solution would be a schedule $\mathcal{S}$ where each subschedule $\mathcal{S}_r$ consisting of the first r arrangements fulfills $\Delta(\mathcal{S}_r) = f(18, 9, r)$ for $r = 3, \dots 15$. Finding such perfect schedule would be increasingly harder than finding an optimal schedule and at this time it is not even clear if such perfect schedule must exist, which is why we propose an alternative way of of finding a „good“ schedule.

By optimizing only a few flights per iteration while fixing the previous ones, we are creating a schedule that is robust against cancellations of later flights. For example the schedule from table 3 was created using an optimization with step-size 2, i.e. starting by taking an optimal $(18, 9, 3)$-schedule $\mathcal{S}_3$, we find a $(18, 9, 5)$-schedule with minimal fairness deviation under all such schedules where the first three flights are predetermined by $\mathcal{S}_5$. Continuing this idea we take this newly found schedule $\mathcal{S}_5$ and optimize for a $(18, 9, 7)$-schedule given the first 5 flights are fixed, repeating this step gives a robust $(18, 9, 15)$-schedule and even allows for extending more flights if needed without having to restart the computation.

Using such step-by-step optimization may break optimality but provides a more practical relevant solution. For some step-sizes even this method even provided upper bounds with short computation time in cases where Gurobi took significantly longer finding the same bound. One may experiment with different step-sizes or a controllable step-size which might lead to an even better bound.

Another way of increasing the robustness of a schedule after its found, is by permuting

the arrangements. Given a (n, k, r)-schedule we can cross out each arrangement and check for the fairness deviation that the remaining $r - 1$ arrangements form, placing the one providing best results at the r-th position. Repeating this recursively ensures that the flights that are less important for overall fairness are scheduled at the end.

3.4 Algorithmic Abstraction

To a certain extend it is possible to abstract the work from previous section for creating a constructive algorithm, that allows for either finding an optimal fair / almost fair schedule or returning that there does not exists such schedule for given parameters k and r, resulting in $f(2k, k, r) \geq 2$. While it may be technically possible to also exhaust the possibilities of a schedule with $\Delta(\mathcal{S}) = 2$ or even higher but we will see that this is not realistic from a computational point of view, at least not for the idea behind the presented algorithm.

As first step, revisiting the problems of Theorem 4, we need a way of finding all possible assignments for the co-occurrence numbers. Since we know that there can't exist a fair and an almost fair schedule at the same time we can start by writing $r(k-1) = (2k-1)q + p$ for $0 \leq p \leq 2k-2$.

If $p = 0$ we will check for the existence of a fair schedule. In this case $\lambda_{ij} = q$ for all $i, j \in N_*^2$ by Corollary 2.

Checking for the existence of an almost fair schedule in the case of $p \neq 0$ does turn out as a harder task. Using Corollary 3 we know that the co-occurrence graphs are going to be regular:

$$\left|\left\{j \in N \backslash \{i\} : \lambda_{ij} = q+1\right\}\right| = p \quad \text{and} \quad \left|\left\{j \in N \backslash \{i\} : \lambda_{ij} = q\right\}\right| = 2k-1-p,$$

resulting in the $(q+1)$-co-occurrence graph being p-regular and the q-co-occurrence graph being $(2k-1-p)$-regular. Luckily both graphs are complementary to each other, allowing us to construct one by the other. We can therefore focus on finding, up to isomorphism, all p-regular graphs (w.l.o.g. we can assume $p \leq 2k-1-p$). Markus Meringer already did some work on constructing regular graphs [7] that we can use

for constructing possible co-occurrence graphs. His algorithm only returns complete graphs, but a small modification will yield arbitrary p-regular graphs. For the purpose of completion we will shortly discuss the approach of this generation:

We start by revisiting the idea of isomorphic graphs. For simplicity we can assume to have vertices 1 through n.

Definition 14. *Let $\mathcal{G}_n$ describe the set of all simple, labeled and undirected graphs with vertices $[n]$. Since the vertices are fixed, we can identify a graph $\Gamma \in \mathcal{G}_n$ by its edges and will therefore write*

$$\Gamma = \{e_1, e_2, \ldots, e_t\} \subset \binom{[n]}{2} =: X_n$$

where X_n describes the set of all possible edges (w.l.o.g. we assume $v < w$ for an edge $e = (v, w) \in X_n$).
The neighborhood of a vertex $v \in [n]$ is defined as all vertices w that are connected to v, i.e.

$$N_\Gamma(v) := \{w \in [n] \,|\, (v, w) \in \Gamma \vee (w, v) \in \Gamma\}$$

Definition 15. *$\mathcal{R}_{n,k} \subset K$ describes the set of all k-regular n graphs, i.e. $\Gamma \in \mathcal{R}_{n,k}$ if $\deg v = |N_\Gamma(v)| = k$ for all $v \in \Gamma$.*

For a formal definition of isomorphism we also need to remember the idea of a group operating on sets by a group action:

Definition 16. *Let X be a set and $(G, \circ)$ be a group with identity element* id. *A function $\alpha : G \times X \to X$ is called group action, if*

1. $\forall x \in X : \alpha(\mathrm{id}, x) = x$
2. $\forall x \in X, g, h \in G : \alpha(g, \alpha(h, x)) = \alpha(g \circ h, x)$

The orbit of such group action on a $x \in X$ is the given by

$$Gx := \{\alpha(g, x) \,|\, g \in G\}.$$

We denote $G\backslash\backslash X := \{Gx \,|\, x \in X\}$ as the set of all orbits.

By trivial reasoning the symmetric group S_n acts on $\mathcal{G}_n$ with the group action $\alpha : S_n \times \mathcal{G}_n \to \mathcal{G}_n$ defined by

$$\alpha(\pi, \{(v_1, w_1), \ldots, (v_t, w_t)\}) := \{\tilde{\alpha}(\pi, (v_1, w_1)), \ldots, \tilde{\alpha}(\pi, (v_t, w_t))\}$$

where

$$\tilde{\alpha}(\pi, (v, w)) := (\min(\pi(v), \pi(w)), \max(\pi(v), \pi(w)))$$

This group action also allows S_n to act on $\mathcal{R}_{n,k}$:

Lemma 12. *Let $\Gamma \in \mathcal{R}_{n,k}$ where $e_i = (v_i, w_i)$, then $\alpha(\pi, \Gamma) \in \mathcal{R}_{n,k}$, which implies that $\alpha_{|S_n \times \mathcal{R}_{n,k}}$ is a corresponding group action.*

Proof. Let $\pi \in S_n$, i.e. π is a permutation on $[n]$. For a fixed $u, w \in [n]$ there exist $v, x \in [n]$ with $\pi(v) = u, \pi(x) = w$ and therefore:

$$(u, w) \in \alpha(\pi, \Gamma) \iff (u, w) = \tilde{\alpha}(\pi, ((v, x))) \text{ and } ((v, x) \in \Gamma \text{ or } (x, v) \in \Gamma)$$

This implies

$$\begin{aligned}(u, w) \in \alpha(\pi, \Gamma) \text{ or } (w, u) \in \alpha(\pi, \Gamma) &\iff (v, x) \in \Gamma \text{ or } (x, v) \in \Gamma \\ &\iff (\pi^{-1}(u), \pi^{-1}(w)) \in \Gamma \text{ or } (\pi^{-1}(w), \pi^{-1}(u)) \in \Gamma,\end{aligned}$$

which gives

$$\begin{aligned}N_{\alpha(\pi,\Gamma)}(u) :&= \{w \,|\, (u, w) \in \alpha(\pi, \Gamma) \text{ or } (w, u) \in \alpha(\pi, \Gamma)\} \\ &= \{w \,|\, (\pi^{-1}(u), \pi^{-1}(w)) \in \Gamma \text{ or } (\pi^{-1}(w), \pi^{-1}(u)) \in \Gamma\} \\ &= \{\pi(w) \,|\, (\pi^{-1}(u), w) \in \Gamma \text{ or } (w, \pi^{-1}(u)) \in \Gamma\} \\ &= \pi(N_\Gamma(\pi^{-1}(u))).\end{aligned}$$

By definition of regularity it follows that $\alpha(\pi, \Gamma)$ is also k-regular since

$$\deg u = |N_{\alpha(\pi,\Gamma)}(u)| = |\pi(N_\Gamma(\pi^{-1}(u)))| = |N_\Gamma(\pi^{-1}(u))| = k$$

□

Definition 17. *Two graphs $\Gamma_1, \Gamma_2 \in \mathcal{G}_n$ are isomorph if they have in the same orbit, i.e. $S_n\Gamma_1 = S_n\Gamma_2$*

For our goal of finding all k-regular graphs we aim to find a set of orbit representatives of $S_n \backslash\backslash \mathcal{R}_{n,k}$, which satisfies the task of finding all such graphs up to isomorphism.
For algorithmic purpose the proposed paper narrows this down to finding minimal representatives.

Definition 18. *For $e_1, e_2 \in X_n$ where $e_1 = (v_1, w_1)$ and $e_2 = (v_2, w_2)$ we define a lexicographic order on X_n by*

$$e_1 < e_2 \iff v_1 < v_2 \text{ or } (v_1 = v_2 \text{ and } w_1 < w_2)$$

and therefore an order on $\mathcal{G}_n$. Let $\Gamma_1, \Gamma_2 \in \mathcal{G}_n$ where $\Gamma_1 = \{e_1, \dots, e_t\}, \Gamma_2 = \{f_1, \dots, f_r\}$. w.l.o.g. assume that these edges are ordered by above logic, then

$$\begin{aligned}\Gamma_1 < \Gamma_2 \iff &(\exists i \leq \min t, r : \forall j < i : e_j = f_j \text{ and } e_i < f_i) \\ &\text{or } (t < r \text{ and } \forall j \leq t : e_j = f_j)\end{aligned}$$

We therefore define a set of minimal orbit representatives by

$$\begin{aligned}\mathrm{rep}_<(S_n \backslash\backslash \mathcal{G}_n) &:= \{\Gamma \in \mathcal{G}_n \,|\, \forall \pi \in S_n : \Gamma \leq \alpha(\pi, \Gamma)\} \\ \mathrm{rep}_<(S_n \backslash\backslash \mathcal{R}_{n,k}) &:= \{\Gamma \in \mathcal{R}_{n,k} \,|\, \forall \pi \in S_n : \Gamma \leq \alpha(\pi, \Gamma)\}\end{aligned}$$

Using minimal representatives allows for an simple backtracking algorithm, that is based on the following theorem.

Theorem 7. *If $\Gamma \in \mathcal{G}_n$ is a minimal orbit representative, i.e. $\Gamma \in \mathrm{rep}_<(S_n \backslash\backslash \mathcal{G}_n)$, than every smaller subset of Γ is also minimal.*

Proof. Suppose, for a proof by contradiction, that $\Gamma = \Gamma_1 \cup \Gamma_2$ where $\Gamma_1 < \Gamma$ but $\Gamma_1 \notin \mathrm{rep}_<(S_n \backslash\backslash \mathcal{G}_n)$. Since Γ_1 is not minimal there exists a permutation $\pi \in S_n$ such that $\alpha(\pi, \Gamma_1) < \Gamma_1$.
Let $\alpha(\pi, \Gamma_1) = \{e_1, \dots, e_t\}, \Gamma_1 = \{f_1, \dots, f_t\}$, by definition of the order there must exists a i with $e_i < f_i$ and $e_j = f_j$ for each $j < i$.
If we now consider $\alpha(\pi, \Gamma) = \alpha(\pi, \Gamma_1) \cup \Gamma_3$ (where $\Gamma_3 = \{b_1, \dots, b_r\}$ are just the remaining edges), then we get two different cases:

1. If $\min_j b_j > e_i$ then we can simply follow that $\alpha(\pi, \Gamma) < \Gamma_1 < \Gamma$.

2. If $\min_j b_j < e_i$, i.e. there exists $2 \leq k \leq i$ such that $e_{k-1} < \min_j b_j < e_k$ then in the compare of $\alpha(\pi, \Gamma)$ and Γ_1 the edge $\min_j b_j$ is the k-th biggest edge in $\alpha(\pi, \Gamma)$. Since for all $j < k$ we know that $e_j = f_j$ and $\min_j b_j < e_k = f_k$ its also clear that $\alpha(\pi, \Gamma) < \Gamma_1 < \Gamma$.

Either case resulting in a contradiction to the minimality of Γ. □

We can now formulate an algorithm for finding $\text{rep}_<(S_n \backslash\backslash \mathcal{R}_{n,k})$:

Algorithm 1 Recursive enumeration of k-regular graphs on n vertices

```
1: procedure ORDREK(Γ)
2:     if not EXTENDABLE(Γ) then
3:         return
4:     end if
5:     if Γ ∉ rep_<(S_n \ G_n) then
6:         return
7:     end if
8:     if Γ ∈ R_{n,k} then
9:         add Γ to output
10:        return
11:    end if
12:    for all e ∈ X_n with e > max Γ do
13:        ORDREK(Γ ∪ {e})
14:    end for
15: end procedure
```

The function EXTENDABLE(Γ) checks for multiple necessary conditions to allow filtering out graphs, that can not be extended to an k-regular graph. An simple example for such necessary condition is that all vertices have degree at most k, more complex conditions can be found in chapter 2 and 3 of [7].

The most time consuming part of the algorithm is found in line 5 where one has to check for minimality of a graph. We won't go into detail on how this can be done effectively and only refer to the work of Markus Meringer again.

After finding all k-regular n-graphs, we can go back to our primary goal of checking for the existence of almost fair schedules by iterating over these graphs and using them as candidates for the co-occurrence-graphs.

For a outline of the idea we will try to fill the assignment Matrix $(S_{li})_{l\in[r],i\in[2k]}$ iterative column after column while not breaking the conditions that we got from the co-occurrence-graph for the co-occurrences between two teams. Each step will give us an linear diophantine system of equation with some bounds. Solving this system will give a finite solution set that is going to be explored by an divide-and-conquer approach until we either find a full assignment matrix or a system yields no solutions allowing us for discard this branch and backtrack to the next solution to explore.

Since every step will create a finite amount of smaller cases we can ensure the algorithm is deterministic and will either output a assignment matrix of a almost fair schedule or will exhaust all possible placements and ensures that no such schedule can exist.

To finish up we are left with two tasks:

1. Formulating the systems needed solve in a similar, but abstracted way to the proofs from Section 3.3
2. Efficiently getting all solutions to bounded linear diophantine system of equation

Assuming we want to place the i column of the assignment matrix, where the columns 1 to $i-1$ are already set, one could think of simply trying out all possible combinations. In the long run this would be highly cost intensive which is why we are going to use the symmetry of our problem to place bigger blocks of 1s and 2s. Lets revisit the proof of Theorem 3 with the explicit value of $r = 12$ as an example:

Let $n = 2k$ where $k = 3$ and let $r = 12$. Then $(k-1)r = q(n-1)+p$ where $q = 4$ and $p = 4$. We want to check for the existence of an almost fair schedule and observe the following co-occurrence-graphs:

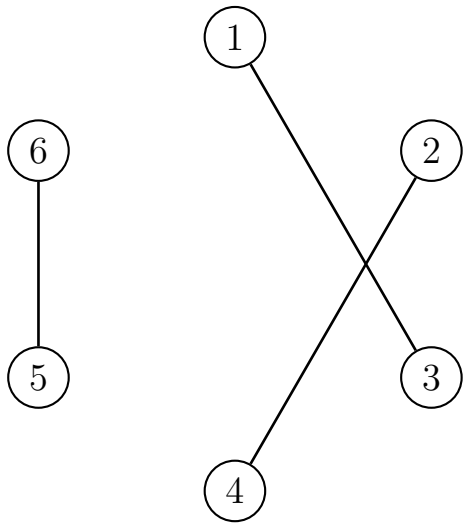


Figure 4: 4-co-occurrence graph

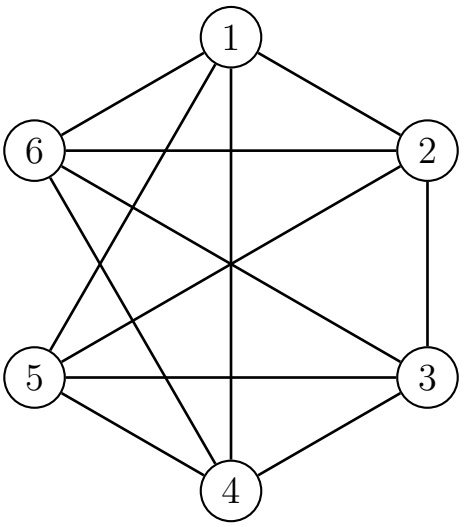


Figure 5: 5-co-occurrence graph

Note that this graph is not minimal nor the exact same as used in the proof of said theorem, but chosen because it demonstrate the idea of formulating the system of equations really well. One could choose arbitrary other fitting co-occurrence-graph but potentially need more steps to see the abstraction.

We start by placing the first column of the assignment matrix and see that there are no real restrictions, since no other team has been places, and therefore start with

$$S_{l1} = 1 \qquad \text{for all } 1 \leq l \leq 2k$$

For placing the second column we have to take into account all co-occurrence restrictions with the columns that are already set, in this case only $4 = \lambda_{12} = g_1^{(2)}(1, r)$. We can therefore say that $S_{l2} = 1$ four times and $S_{l2} = 2$ eight times.
Now the idea 1- and 2-blocks start to show, because we will start with a 1-block of size 4 and then a 2-block of size 8. For our abstractions we will store these sizes in a vector p. And get:

$$p^{(2)} = (g_1^{(2)}(1, r), r - g_1^{(2)}(1, r)) \qquad \text{and by following this idea also } p^{(1)} = r$$

Considering the third column we start to have an actual system of equations since we now have to consider λ_{13} and λ_{23}:

$$\begin{aligned} \lambda_{13} &= g_1^{(3)}(1, r) \\ \lambda_{23} &= g_1^{(3)}(1, p_1^{(2)}) + g_2^{(3)}(p_1^{(2)} + 1, r) \end{aligned}$$

By using the properties of g and trivial bounds we get the following system:

$$\begin{aligned} \lambda_{13} &= g_1^{(3)}(1, p_1^{(2)}) + g_1^{(3)}(p_1^{(2)} + 1, r) \\ \lambda_{23} - (r - (p_1^{(2)} + 1) - 1) &= g_1^{(3)}(1, p_1^{(2)}) - g_1^{(3)}(p_1^{(2)} + 1, r) \\ 0 \leq g_1^{(3)}(1, p_1^{(2)}) &\leq p_1^{(2)} \\ 0 \leq g_1^{(3)}(p_1^{(2)} + 1, r) &\leq r - p_1^{(2)} \end{aligned}$$

Solving this yields a new p vector for placing the third column:

$$p^{(3)} = (g_1^{(3)}(1, p_1^{(2)}), p_1^{(2)} - g_1^{(3)}(1, p_1^{(2)}), g_1^{(3)}(p_1^{(2)} + 1, r), r - p_1^{(2)} - g_1^{(3)}(p_1^{(2)} + 1, r))$$

When following this idea it may seem like the system gets exponentially harder to solve since each steps only adds one equation, but doubles the amount of variables. A new constraint counting 1s and 2s each row and ensuring that there can only be k teams per flight helps with this problem when j, the column that has to be placed, gets higher.

For an more readable system we define the vector P as partial sums of p for $j = 0, \ldots, 2^{t-1}$:

$$P_j^{(t)} = \sum_{l=1}^{t} p_l^{(t)} \quad \text{and therefore} \quad p_j^{(t)} = P_j^{(t)} - P_{j-1}^{(t)} \quad \text{where} \quad P_0^{(t)} = 0, P_{2^{t-1}}^{(t)} = r$$

As final abstraction, consider placing the t-th column. We get the following equations:

$$\lambda_{lt} = \sum_{j=1}^{2^{t-2}} g_{\omega_l^{(t)}(j)}^{(t)}(P_{j-1}^{(t-1)} + 1, P_j^{(t-1)}) \quad \text{for } l < t$$

where $\omega_l^{(t)}(j)$ describes wether to consider $g_1^{(t)}$ or $g_2^{(t)}$.

To understand the behavior of $\omega_l^{(t)}(j)$, let us go one step further in our example and consider placing the fourth column:

$$\begin{aligned}
\lambda_{14} &= g_1^{(4)}(1, r) \\
\lambda_{24} &= g_1^{(4)}(1, p_1^{(2)}) + g_2^{(4)}(p_1^{(2)} + 1, r) \\
\lambda_{34} &= g_1^{(4)}(1, p_1^{(3)}) \\
&\quad + g_2^{(4)}(p_1^{(3)} + 1, p_1^{(3)} + p_2^{(3)}) \\
&\quad + g_1^{(4)}(p_1^{(3)} + p_2^{(3)} + 1, p_1^{(3)} + p_2^{(3)} + p_3^{(3)}) \\
&\quad + g_2^{(4)}(p_1^{(3)} + p_2^{(3)} + p_3^{(3)} + 1, r)
\end{aligned}$$

this simplifies to:

$$\begin{aligned}
\lambda_{14} &= g_1^{(4)}(P_0^{(4)} + 1, P_1^{(4)}) + g_1^{(4)}(P_1^{(4)} + 1, P_2^{(4)}) + g_1^{(4)}(P_2^{(4)} + 1, P_3^{(4)}) + g_1^{(4)}(P_3^{(4)} + 1, P_4^{(4)}) \\
\lambda_{24} &= g_1^{(4)}(P_0^{(4)} + 1, P_1^{(4)}) + g_1^{(4)}(P_1^{(4)} + 1, P_2^{(4)}) + g_2^{(4)}(P_2^{(4)} + 1, P_3^{(4)}) + g_2^{(4)}(P_3^{(4)} + 1, P_4^{(4)}) \\
\lambda_{34} &= g_1^{(4)}(P_0^{(4)} + 1, P_1^{(4)}) + g_2^{(4)}(P_1^{(4)} + 1, P_2^{(4)}) + g_1^{(4)}(P_2^{(4)} + 1, P_3^{(4)}) + g_2^{(4)}(P_3^{(4)} + 1, P_4^{(4)})
\end{aligned}$$

This shows that $\omega_{t-1}^{(t)}$ seems to alternate between 1 and 2, and the period length doubles when going from $\omega_l^{(t)}$ to $\omega_{l-1}^{(t)}$ (this can be shown by induction but is left out for the sake of clarity), leaving us with

$$\omega_l^{(t)} = 2^{t-l}\text{-periodic alternating sequence between 1 and 2, starting at 1}$$

We can achieve such sequence by modifying $(-1)^j$ and get

$$\omega_l^{(t)}(j) = \frac{1}{2}\left(3 - (-1)^{\left\lfloor \frac{j-1}{2^{t-l}} \right\rfloor}\right)$$

We can now rewrite our system of equations into a system where the only relevant variables are $g_1^{(t)}(P_{j-1}^{(t-1)} + 1, P_j^{(t-1)})$. From Lemma 10 we get

$$\begin{aligned}
g_2^{(t)}(P_{j-1}^{(t-1)} + 1, P_j^{(t-1)}) &= P_j^{(t-1)} - (P_{j-1}^{(t-1)} + 1) + 1 - g_1^{(t)}(P_{j-1}^{(t-1)} + 1, P_j^{(t-1)}) \\
&= p_j^{(t-1)} - g_1^{(t)}(P_{j-1}^{(t-1)} + 1, P_j^{(t-1)})
\end{aligned}$$

and therefore

$$g^{(t)}_{\omega_l^{(t)}(j)}(P_{j-1}^{(t-1)}+1, P_j^{(t-1)}) = (-1)^{\left\lfloor \frac{j-1}{2^{t-l}} \right\rfloor} \cdot g_1^{(t)}(P_{j-1}^{(t-1)}+1, P_j^{(t-1)}) + \frac{1}{2}\left(1-(-1)^{\left\lfloor \frac{j-1}{2^{t-l}} \right\rfloor}\right) \cdot p_j^{(t-1)}$$

Inserting this back in our original system of equations we have

$$\lambda_{lt} - \sum_{j=1}^{2^{t-2}} \frac{1}{2}\left(1-(-1)^{\left\lfloor \frac{j-1}{2^{t-l}} \right\rfloor}\right) \cdot p_j^{(t-1)} = \sum_{j=1}^{2^{t-2}} (-1)^{\left\lfloor \frac{j-1}{2^{t-l}} \right\rfloor} \cdot g_1^{(t)}(P_{j-1}^{(t-1)}+1, P_j^{(t-1)}) \text{ for } l < t$$

The system gets paired with the following restrictions:

$$0 \leq g_1^{(t)}(P_{j-1}^{(t-1)}+1, P_j^{(t-1)}) \leq p_j^{(t-1)} \tag{1}$$

$$g_1^{(t)}(P_{j-1}^{(t-1)}+1, P_j^{(t-1)}) \in \mathbb{Z} \tag{2}$$

$$\text{If } \sum_{i=1}^{t-1} S_{li}^{(t-1)} - 1 = k \text{ then } \forall j \; : g_1^{(t)}(P_{j-1}^{(t-1)}+1, P_j^{(t-1)}) = p_j^{t-1} \tag{3}$$

$$\text{If } \sum_{i=1}^{t-1} S_{li}^{(t-1)} - 1 = (t-1) - k \text{ then } \forall j \; : g_1^{(t)}(P_{j-1}^{(t-1)}+1, P_j^{(t-1)}) = 0 \tag{4}$$

A solution to this system allows for constructing the new values for $P_j^{(t)}$, $p_j^{(t)}$ and $S_{li}^{(t)}$, needed to perform the next iteration:

$$
\begin{aligned}
p_{2j-1}^{(t)} &= g_1^{(t)}(P_{j-1}^{(t-1)} + 1, P_j^{(t-1)}), && \text{for } j = 1, \ldots, 2^{t-2} \\
p_{2j}^{(t)} &= p_j^{(t-1)} - g_1^{(t)}(P_{j-1}^{(t-1)} + 1, P_j^{(t-1)}), && \text{for } j = 1, \ldots, 2^{t-2} \\
P_0^{(t)} &= 0, \\
P_j^{(t)} &= P_{j-1}^{(t)} + p_j^{(t)}, && \text{for } j = 1, \ldots, 2^{t-1} \\
S_{li}^{(t)} &= S_{li}^{(t-1)} && \text{for } l = 1, \ldots, 2k \text{ and } i = 1, \ldots, r \text{ with } i \neq t, \\
S_{lt}^{(t)} &= \begin{cases} 1, & \text{if } j \text{ is odd} \\ 2, & \text{if } j \text{ is even} \end{cases} && \text{with } j \text{ such that } l \in \{P_{j-1}^{(t)} + 1, \ldots, P_j^{(t)}\}
\end{aligned}
$$

The last step is to actually solve such system under the given constraints.

The constraints (3) and (4) can simply be evaluated using if statement as they stand. But to cover (1) and (2) we need the use of an algorithm that allows returning all possible solutions, not just one (which would be fairly easy).

Since we have a finite amount of possible solutions (at most $\prod_j (p_j^{(t-1)} + 1)$) we can exhaust them by using a box enumeration. This process can also be sped up using small modifications:

Let us suppose we have a system of equations with bounded integer variables

$$Ax = b, 0 \leq x \leq c, \quad A \in \mathbb{Z}^{m \times n}, x, c \in \mathbb{Z}^n, b \in \mathbb{Z}^m, m \gg n$$

we can find all solutions to this system using the following algorithm:

Algorithm 2 Finding all solutions to bounded linear diophantine system

```
1: procedure SolveBounded(A,b,c)
2:     (U,H,r) := RowHermite(A)
3:     b' := U·b

4:     x_par := BackSubstitution(H, b'[1:r])
5:     (V,d) := NullSpaceBasis(H)
6:     if x_par is None then
7:         return ∅
8:     end if

9:     if d = 0 then
10:        if 0 ≤ x_par ≤ c then
11:            return {x_par}
12:        else
13:            return ∅
14:        end if
15:    end if

16:    lb, ub := ProjectBounds(x_par, V, c)
17:    sol := ∅
18:    for all y ∈ ℤ^d with lb <= y <= ub do
19:        x := x_par + V·y
20:        if 0 ≤ x ≤ c then
21:            append sol, x
22:        end if
23:    end for
24: end procedure
```

We start by calculating the Row-Hermite Normal Form and therefore reducing the system into a triangular system $Hx = b'$ using the unimodular transform U (Now H has dimensions $r \times n$). This allows for a simple way of finding a single solution x_par and a basis of the Nullspace of H. After finding the Nullspace and a particular solution we can exhaust all other possible solutions by simply iterating over our new box bounds since we know that every combinations $x_{par} + V \cdot y$ is a solution to the real-valued

problem.

There are a few things to watch out for when implementing, that we won't cover here. Just to name an example, the basis V has to be a lattice basis of the integer kernel of H. This can be seen when considering the space $\{x \in \mathbb{R}^3 \mid x_1 + x_2 - 2x_3 = 0\}$. One possible basis would be $V = \{(2,0,1)^T, (0,2,1)^T\}$, but we can clearly see that we would miss integer solutions when only considering $V\mathbb{Z}^d$. Instead the needed lattice basis would be $V = \{(2,0,1)^T, (-1,1,0)^T\}$.

There are also multiple ways of speeding up the algorithm, for example the exhaustion of solutions $x_{par} + V \cdot y$ can be done by either pruning single coordinates after a bound violation or by a similar approach as used in the branch-and-bound-algorithm by doing interval propagation. Since these are again steps that are relevant in the implementation and crucial for the general idea, we will not cover explanation of steps in this work.

Now, having a way of generating possible co-occurrence-graphs, formulated a system to solve and found a way to actually solve it, we have a constructive and deterministic algorithm to check for the existence of fair and almost fair schedules.

4 Conclusion

Summarizing the work we have done we will take a look back at the introduction rounding up our plans.

We started by defining the fairness deviation $f(n, k, r)$ as the span between the co-occurrence numbers between two teams Using concepts of number theory we showed that a fair schedule, i.e. $f = 0$, is only possible if $r(k-1) \equiv 0 \mod (n-1)$. We strengthened this necessary condition to $r(k-1) \equiv 0 \mod 2(n-1)$ for the case of $n = 2k$ where k is odd, even though numerical experiment yet show no contradiction it remains unclear if this condition is sufficient. We also provided the distribution of co-occurrence numbers in case of the existence of an almost fair schedule.

After this we looked at two types of optimization formulation for the case of $n = 2k$, a linear and a quadratic one. By providing different types of symmetry breaking constraints we got a total of 6 different formulations that we then tested on $k = 3, \ldots, 9$ and $r = 3, \ldots, 20$ under even conditions. The results showed that for practical applications either the basic quadratic formulation or the quadratic formulation with additional binary order of flights are most reliable. The experiments did not show a pattern when one of these two performed better than the other. What they did show, was that for $k \geq 7$ and $r \geq 8$ a clear hardness bump occurred making it hard for finding optimal schedules for larger values of k and r.

While we proved that $f(n, k, r) = 1$ occur if either $f(n, k, r-1) = 0$ or $f(n, k, r+1) = 0$ and numerical results suggests that these are the only times that $f = 1$, we could not proof that this is true in general.

Motivated by the numerical results we took a closer look at fixed k and proved 10-periodic behavior for $k = 3$ and 7-periodic behavior for $k = 4$, allowing for complete characterization of these cases. Proving these was done by using a helper function g to count the occurrences of a team in a fixed flight of consecutive arrangements. Properties of this function lead to diophantine systems of equations, contradicting the existence of fair or almost fair schedules in the relevant cases.

We used this idea to formulate an abstract algorithm for doing so in the case of arbitrary $n = 2k$ and r using an existing algorithm to find possible co-occurrence graphs and applying the results to our abstracted system of equations from previous thoughts.

In the analysis of the polish sailing league we took a closer look at the 4th round of 2021 (Ekstraklasa) and observed a fairness deviation of 9. Using different techniques we first proofed a lower bound of 2 in the given case and presented a robust schedule with fairness deviation of 4, that can withstand cutting the last flights without loosing too much fairness deviation.

5 Literature

[1] R. Schüler and A. Schürmann: *Sailing League Problems*, Journal of Combinatorial Designs, vol. 32, no. 4, April 2024, pp. 171-189.

[2] C.J. Colbourn and J.H. Dinitz (eds.): *Handbook of Combinatorial Designs*, 2nd ed., Chapman & Hall, 2007.

[3] R.E.A.C. Paley: *On orthogonal matrices*, Journal of Mathematical Physics, vol. 12, no. 1-4, April 1933, 311-320.

[4] H. Kharaghani and B. Tayfeh-Rezaie: *A Hadamard matrix of order 428*, Journal of Combinatorial Designs, vol. 13, No. 6, November 2005, pp. 435-440.

[5] A. Hedayat: Hadamard matrices and their applications, The Annals of Statistics, vol. 6, no. 6, 1978, pp. 1184-1238.

[6] S. Mirzaei: *Linear Programming: An Introduction*, 1st ed., Kendall/Hunt, 2019

[7] M. Meringer: *Erzeugung regulärer Graphen*. Master's thesis, Universität Bayreuth (1996).

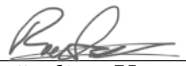
[8] SAP Sailing: Regatta Overview of Ekstraklasa 2021 Round 4, available at: `https://www.sapsailing.com/gwt/Home.html#/regatta/overview/:eventId=323b9067-a905-47bc-9fe9-12919cc48c6c` (accessed June 11, 2025).

6 Erklärung über die selbständige Abfassung einer schriftlichen Arbeit

Hiermit erkläre ich, Cedric Rönnfeld, Matrikel-Nr. 221200104, dass ich die vorliegende Arbeit selbständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Rostock

20.08.2025 (Abgabedatum)

 (Vollständige Unterschrift)