Collabits Journal
Vol. 2 No. 2 May 2025 : 73-81
E-ISSN : 1979-5254, P-ISSN : 3062-8601
https://publikasi.mercubuana.ac.id/index.php/collabits

Evaluation of the Effectiveness of Hybrid Learning Based on Linear Algebraic Hybrid Model in the Online-Offline Lecture System in the Digital Era

Andre Meyro Ritonga[1*], Nicholas Sulistio[2], Guruh Pandhu Anggriawan[3], Mohamad Yusuf[4]

[1,2,3,4] Informatics Engineering, Universitas Mercu Buana, Indonesia

*Coressponden Author : 41522120024@student.mercubuana.ac.id

Abstract - Optimal class division is a crucial aspect of academic planning to ensure the effectiveness of the learning process. The main challenges in class division lie in the limited capacity of space, balanced distribution of students, and the fulfillment of varied academic needs. This study proposes a Linear Programming-based approach to optimize class division by considering various constraints, such as the maximum capacity of the room, the number of students, and the distribution of subjects according to curriculum needs. The developed applications are designed to produce optimal solutions that minimize student distribution gaps and ensure efficient classroom utilization. A case study is applied to an educational institution to evaluate the performance of the application in real situations. The results of the experiment show that this approach is able to improve the efficiency of classroom allocation, reduce imbalances in the distribution of students, and optimize the use of educational facilities. Thus, this research contributes to more effective and data-based academic management in decision-making related to class division.

Keywords:
Class Optimization;
Linear Programming;
Room Capacity;
Student Distribution;
Academic Planning;

Article History:
Received: 05-03-2025
Revised: 12-04-2025
Accepted: 21-05-2025

Article DOI : 10.22441/collabits.v2i2.32548

1. INTRODUCTION

1.1 Background

In the world of education, class division is a crucial aspect in academic planning that has a direct impact on the effectiveness of the learning process. Optimal class division not only ensures that students get a conducive learning environment but also supports the efficient use of educational facilities. However, in practice, educational institutions often face challenges in determining the allocation of students into the classroom. Some of the common obstacles faced include the limited number of classrooms, varying room capacity, and the distribution of subjects that must be adjusted to the needs of the curriculum. One of the problems that often occurs is the imbalance in the number of students in a class, which can have an impact on teaching effectiveness. Classes that are too crowded can lead to a lack of interaction between students and teachers, as well as lower students' understanding of the material being taught. Conversely, classes with too few students can lead to inefficiencies in the use of resources, such as classrooms and teaching staff. Therefore, a systematic and data-driven method is needed to optimize class division to align with academic capacity and needs.

Linear Programming is one of the mathematical methods that can be used to solve this problem. This method allows the design of optimal solutions by considering various constraints and parameters, such as the maximum capacity, the number of students, and the distribution of subjects according to the curriculum structure. By implementing Linear Programming-based optimization applications, class division can be carried out more efficiently and accurately, resulting in a more balanced distribution of students and more optimal classroom utilization.

This research aims to develop an application of class division optimization using the Linear Programming method with a case study on an educational institution. The application developed will be tested to assess its effectiveness in solving the problem of class division in real life. It is hoped

that the results of this research can contribute to academic management, especially in improving the efficiency of class division, optimizing educational resources, and creating a more effective learning environment for students and teaching staff.

1.2 Supporting Research

1. Linear Programming
   A linear programming problem is an optimization problem that meets the following criteria (Winston, 2004):
   a. Aim to maximize (or minimize) a linear function of a decision variable. This function is referred to as an objective function.
   b. The value of the decision variable must meet certain constraints. Each constraint must be a linear equation or inequality.

1.3 Purpose

This study aims to develop and implement an application of class division optimization based on space capacity and academic needs using the Linear Programming method. With this application, it is hoped that the process of allocating students into the classroom can be carried out more efficiently, balanced, and in accordance with curriculum needs, thereby supporting the creation of an optimal learning environment.

The purpose of this research is as a
next:

1. Analyze problems in class division
   Identify factors that affect imbalances in the distribution of students into classes, such as room capacity, number of students, and subject needs.
1. Develop an optimization application based Pemrograman Linier
   Designing mathematical formulations that can be used to optimally determine the allocation of students into the classroom by taking into account various constraints and parameters.
3. Testing the effectiveness of the App in real-world case studies
   Implementing a class division optimization application in an educational institution to evaluate its level of efficiency and accuracy in distributing students.
4. Improve the efficiency of classroom use and academic resources
   Ensuring that classroom capacity can be used optimally and supporting a more effective distribution of teaching staff.
5. Provide data-driven recommendations for academic management
   Provide solutions that can be implemented by educational institutions in future classroom division planning, so that decision-making can be done in a more structured manner and based on quantitative analysis.

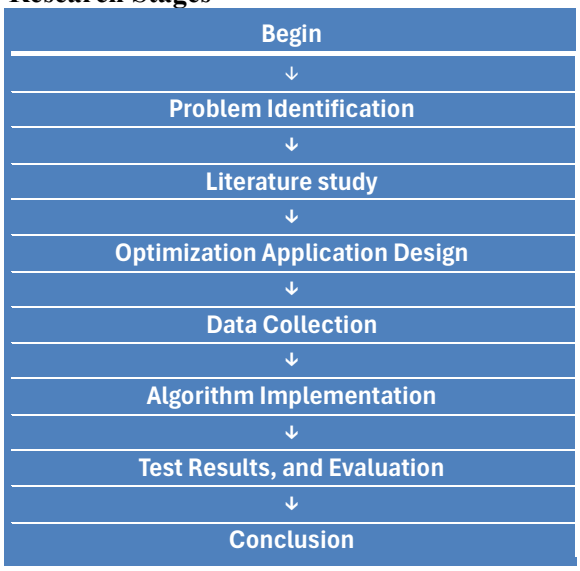
Through this research, it is hoped that a more efficient class division system can be created, so that the teaching and learning process can run more effectively, by utilizing educational resources optimally.

2. METODOLOGI

2.1 Method

- **Type of Research Data:** Primary Data is the type of data used in this study, the data obtained, namely Courses, Number of Students, Lecturers, and Classrooms.
- **Research Object:** Making Schedule for Class Division, and Lecturers at Mercu Buana University
- **Research Location:** Mercu Buana University

2.2 Research Stages



2.3 Research Steps

1. Decision Variables

The code defines two decision variables :

- **Variabel Biner** $x_{mk,d,r,s,t}$

$$x_{mk,d,r,s,t} = \begin{cases} \mathbf{1}, \\ \mathbf{0}, \end{cases}$$

- 1 *if the course is taught by lecturer* d *in room* r *in session* s *at time* t 0 *if not*

  Variabel Integer $y_{mk,d,r,s,t}$

$y_{mk,d,r,s,t}$ = *number of students attending the MK course in room r at session s at t hourt*

2. Objective Function (Minimization)

Although there is no explicit objective function, the application has the primary purpose of minimizing the number of constraint violations, which implicitly minimizes the number of classes that are scheduled inefficiently.

$$\textbf{min} \quad 0$$

(PuLP requires objective function, but in this case only limitations are the main focus.)

3. Constraints

(a) Distribution of class sizes according to the number of students

Each course must have a sufficient number of classes to accommodate all of its students. If the number of students jml_mhsjml_mhsjml_mhs more than 25 per class, then more than one class must be created :

$$\sum_{d,r,s,t} x_{mk,d,r,s,t} = \frac{jml\_mhs}{25}$$

(b) Students must be divided into available classes

The total number of students scheduled must be equal to the number of students enrolled in:

$$\sum_{d,r,s,t} y_{mk,d,r,s,t} = jml\_mhs$$

(c) Maximum capacity limit per class

Each class must not have more than 25 students:

$$y_{mk,d,r,s,t} \leq 25 \cdot x_{mk,d,r,s,t}$$

(d) The distribution of students is even if there is more than one class

If there is more than one class for a single course, the number of students in each class should not differ too much:

$$y_{mk,d,r,s,t} \geq \frac{jml\_mhs}{total\_kelas} \cdot x_{mk,d,r,s,t}$$

(e) One room can only be used for one course at a time

In a specific session and one hour, a room must not have more than one class:

$$\sum_{mk,d} x_{mk,d,r,s,t} \leq 1, \forall r, s, t$$

(f) Maximum of 3 concurrent classes in one session

At any one time, there should only be a maximum of 3 classes that take place at the same time:

$$\sum_{mk,d,r} x_{mk,d,r,s,t} \leq 3, \forall s, t$$

(g) Lecturers are only allowed to teach one class per session

Each lecturer can only teach one class in one session:

$$\sum_{mk,r,t} x_{mk,d,r,s,t} \leq 1, \forall d, s$$

Conclusion

This code creates **a Lecture Schedule Optimization Application** based on **Linear Programming** with:

- **Decision variables** to determine class schedules and number of students.
- **Implicit objective function** that ensures that boundaries are not violated.
- **Restrictions** to ensure there is no excessive room, lecturers, or student capacity.

2.4 Constraints

1. Computational Complexity

- **Scalability**: With so many decision variables and constraints, this application can be very large and complex to complete, especially if the number of courses, lecturers, rooms, sessions, and hours increases.
- **Completion Time**: Application Solving with optimization solvers can take a long time if the number of variables and constraints is large.

2. Student Imbalance

- **Class-splitting students**: While there is a limit to dividing students evenly if there is more than one class, in practice it can be difficult to completely evenly distribute the number of students per class.
- **Classes are not full**: If the number of students for a course is not multiples of 25, then the last class may have a much smaller number of students than the other classes.

3. Resource Limitations

- **Space Limitations**: If there is not enough room, the App may have difficulty finding a workable solution.
- **Limited Number of Lecturers**: If there are not enough lecturers available to meet the needs of the course teaching, then scheduling may become impossible or result in suboptimal solutions.

- **Limited Number of Sessions and Hours**: The app can only work within the available time capacity, which may not be enough to accommodate all classes.

4. Schedule Conflicts

- **Lecturers Teaching More Than One Class**: While there is a restriction that a lecturer can only teach one class in a single session, if a lecturer teaches many courses, scheduling can become more difficult.
- **Concurrent Classes**: If there are many courses with a large number of students, the maximum limit of 3 concurrent classes in one session can be an obstacle.

5. Low Flexibility

- **Not Considering Lecturer and Student Preferences**: This application does not consider the wishes of lecturers or students regarding certain schedules, which can lead to dissatisfaction.
- **Does Not Take Into Consideration Class Sustainability**: This app does not consider whether the same class can be scheduled in a slot time is close together, so it can happen that the same class has a scattered schedule far in a week.

6. Limitations in Objective Functions

- **Not Maximizing Efficiency**: Objective functions don't really pursue optimal time efficiency or resource utilization, only minimizing boundary violations.
- **No Priority Weight**: If any constraint is more important than the other, the App does not have a mechanism to give greater weight to a particular constraint.
- **2.5 Data Collection**

Table 1.1 - Classroom Data

CLASS NAME
CLASS A11
CLASS A22
CLASS A33
CLASS A44
CLASS A55

Table 1.2 – Lecturer Data

Lecturer Name	MATSPEND	ADDITIONAL MATKUL
Aji Subroto	RPL	Alg Lanjut R2
M.Iskandar Lawoly	PBO	Data Structure
Joko Tingkir	Alg Lanjut R2	PBO
Heru Sabar	Machine Learning	RPL
Rivaldo Pakpahan	Deep Learning	RPL
Ferdy Firmansyah	OS	Deep Learning
M.Firly Tubagus	Pem Lanjut R1	OS
Ratna Handoko	Data Structure	Web 1
Hesti Purwadinata	Web 1	OS
Vania Yeastin	Pem Lanjut R1	Data Structure
Monika Anisa	Web 1	Data Structure
Ayu Astuti	Pemrograman Lanjut	Pem Lanjut R1
Putri Maya	RPL	Alg Lanjut R2
Anwar Iskandar	Alg Lanjut R2	PBO

Table 1.3 – Total Student Data

Courses	Number of Students
PBO	28
RPL	43
Alg Lanjut R2	11
Machine Learning	10
Deep Learning	29
OS	50
Advanced Programming	15
Data Structure	43
Web 1	41
Pem Lanjut R1	42

2.6 Results of the Discussion
Code

```
import pulp
import pandas as pd
```

```
import streamlit as st
import seaborn as sns
import matplotlib.pyplot as plt

st.title("Optimasi Jadwal Kuliah")

uploaded_file = st.file_uploader("Upload file Excel", type=["xlsx"])
if uploaded_file:
    xls = pd.ExcelFile(uploaded_file)
    df_kelas = xls.parse("kelas")
    df_dosen = xls.parse("dosen")
                df_totalmahasiswa       = xls.parse("totalmahasiswa")

    st.write("Data Dosen:")
    st.dataframe(df_dosen)
     st.write("Total Mahasiswa per Mata Kuliah:")
    st.dataframe(df_totalmahasiswa)

                          model           = pulp.LpProblem("Optimasi_Jadwal_Kuliah", pulp.LpMinimize)

     sesi_waktu = {"Pagi": range(7, 13, 2), "Sore": range(14, 20, 2)}
           ruangan    =    df_kelas["NAMA KELAS"].dropna().tolist()

    dosen_mk = {}
    for _, row in df_dosen.iterrows():
                             mk_utama      = row["MATKULUTAMA"].split(",     ")     if pd.notna(row["MATKULUTAMA"]) else []
                          mk_tambahan     = row["MATKULTAMBAHAN"].split(",    ")    if pd.notna(row["MATKULTAMBAHAN"]) else []
           dosen_mk[row["Nama  Dosen"]]  = mk_utama + mk_tambahan

     x = pulp.LpVariable.dicts("Jadwal", [(mk, d, r, s, t)
        for mk in df_totalmahasiswa["Mata Kuliah"]
         for d in df_dosen["Nama Dosen"] if mk in dosen_mk.get(d, [])
         for r in ruangan
         for s in sesi_waktu.keys()
              for  t  in  sesi_waktu[s]], cat=pulp.LpBinary)

                              y           = pulp.LpVariable.dicts("Mahasiswa", [(mk, d, r, s, t)
        for mk in df_totalmahasiswa["Mata Kuliah"]
         for d in df_dosen["Nama Dosen"] if mk in dosen_mk.get(d, [])
         for r in ruangan
         for s in sesi_waktu.keys()
              for  t  in  sesi_waktu[s]], lowBound=0, cat=pulp.LpInteger)

            for     mk,     jml_mhs     in zip(df_totalmahasiswa["Mata    Kuliah"], df_totalmahasiswa["Jumlah Mahasiswa"]):
         total_kelas = -(-jml_mhs // 25)
          model += pulp.lpSum(x[mk, d, r, s, t] for d in df_dosen["Nama Dosen"] if mk in dosen_mk.get(d, [])
                                      for r in ruangan
                                      for s in sesi_waktu.keys()
                                      for t in sesi_waktu[s]) == total_kelas

          model += pulp.lpSum(y[mk, d, r, s, t] for d in df_dosen["Nama Dosen"] if mk in dosen_mk.get(d, [])
                                      for r in ruangan
                                      for s in sesi_waktu.keys()
                                      for t in sesi_waktu[s]) == jml_mhs

         for d in df_dosen["Nama Dosen"]:
             if mk in dosen_mk.get(d, []):
                  for r in ruangan:
                                     for  s  in sesi_waktu.keys():
                                      for  t  in sesi_waktu[s]:
                                  model += y[mk, d, r, s, t] <= 25 * x[mk, d, r, s, t]

                                  if total_kelas > 1:
                                       model += y[mk,  d,  r,  s,  t]  >=  (jml_mhs  // total_kelas) * x[mk, d, r, s, t]

    for r in ruangan:
        for s in sesi_waktu.keys():
```

```
            for t in sesi_waktu[s]:
                model += pulp.lpSum(x[mk, d, r, s, t] for mk in df_totalmahasiswa["Mata Kuliah"]
                                    for d in df_dosen["Nama Dosen"] if mk in dosen_mk.get(d, [])
                                    if (mk, d, r, s, t) in x) <= 1

    for s in sesi_waktu.keys():
        for t in sesi_waktu[s]:
            model += pulp.lpSum(x[mk, d, r, s, t] for mk in df_totalmahasiswa["Mata Kuliah"]
                                for d in df_dosen["Nama Dosen"] if mk in dosen_mk.get(d, [])
                                for r in ruangan if (mk, d, r, s, t) in x) <= 3

    for d in df_dosen["Nama Dosen"]:
        for s in sesi_waktu.keys():
            model += pulp.lpSum(x[mk, d, r, s, t] for mk in df_totalmahasiswa["Mata Kuliah"]
                                for r in ruangan
                                for t in sesi_waktu[s] if (mk, d, r, s, t) in x) <= 1

    model.solve()

    schedule = []
    for mk, d, r, s, t in x.keys():
        if x[mk, d, r, s, t].value() == 1:
            schedule.append([mk, d, r, s, t])

    heatmap_df = pd.DataFrame(schedule, columns=["Mata Kuliah", "Dosen", "Ruangan", "Sesi", "Jam"])

    fig, ax = plt.subplots(figsize=(10, 6))

                    pivot = heatmap_df.pivot_table(index="Ruangan", columns="Jam", aggfunc="size", fill_value=0)
        sns.heatmap(pivot, annot=True, cmap="coolwarm", linewidths=0.5, ax=ax)

    st.pyplot(fig)

        st.write("Berikut adalah hasil optimasi jadwal kuliah:")

                    heatmap_df = heatmap_df.reset_index(drop=True)

        heatmap_df.index = range(1, len(heatmap_df) + 1)
    st.dataframe(heatmap_df)
```

Test and Evaluation Results

Figure 1.0 - Early UI of Lecture Schedule Optimization Application

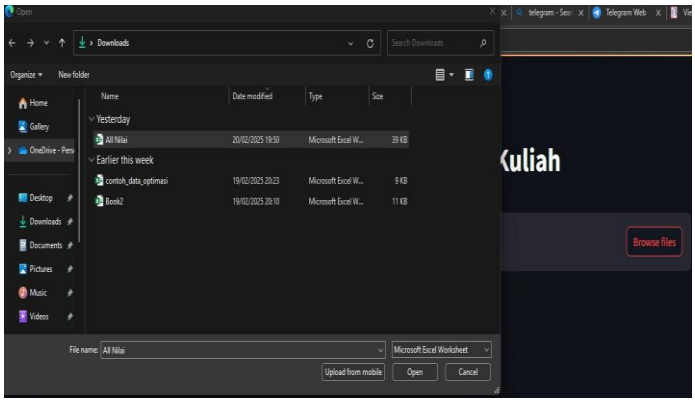
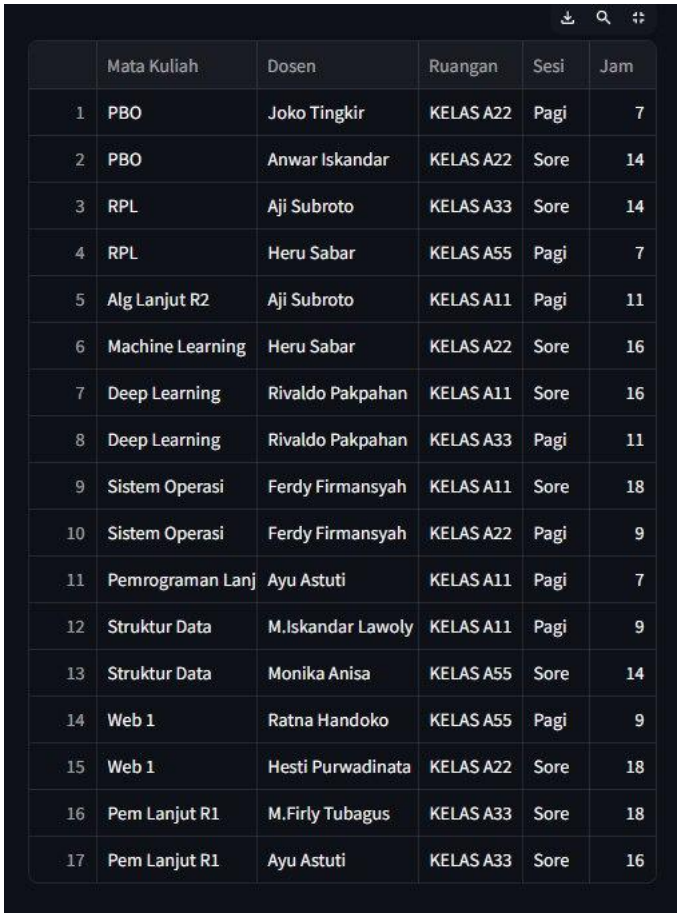


Figure 1.1 - Upload Lecturer, Class, and Student Course Data

Figure 1.2 - Successfully uploaded Lecturer Data

Figure 1.3 - Course Data, and Number of Students Successfully Uploaded



	Mata Kuliah	Dosen	Ruangan	Sesi	Jam
1	PBO	Joko Tingkir	KELAS A22	Pagi	7
2	PBO	Anwar Iskandar	KELAS A22	Sore	14
3	RPL	Aji Subroto	KELAS A33	Sore	14
4	RPL	Heru Sabar	KELAS A55	Pagi	7
5	Alg Lanjut R2	Aji Subroto	KELAS A11	Pagi	11
6	Machine Learning	Heru Sabar	KELAS A22	Sore	16
7	Deep Learning	Rivaldo Pakpahan	KELAS A11	Sore	16
8	Deep Learning	Rivaldo Pakpahan	KELAS A33	Pagi	11
9	Sistem Operasi	Ferdy Firmansyah	KELAS A11	Sore	18
10	Sistem Operasi	Ferdy Firmansyah	KELAS A22	Pagi	9
11	Pemrograman Lanj	Ayu Astuti	KELAS A11	Pagi	7
12	Struktur Data	M.Iskandar Lawoly	KELAS A11	Pagi	9
13	Struktur Data	Monika Anisa	KELAS A55	Sore	14
14	Web 1	Ratna Handoko	KELAS A55	Pagi	9
15	Web 1	Hesti Purwadinata	KELAS A22	Sore	18
16	Pem Lanjut R1	M.Firly Tubagus	KELAS A33	Sore	18
17	Pem Lanjut R1	Ayu Astuti	KELAS A33	Sore	16

Figure 1.4 - Optimization results of uploaded data

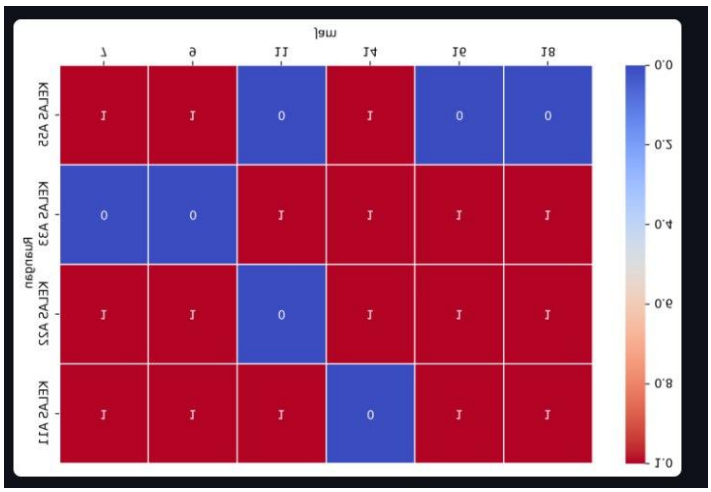


Figure 1.5 - Heatmap visualization based on Optimization results

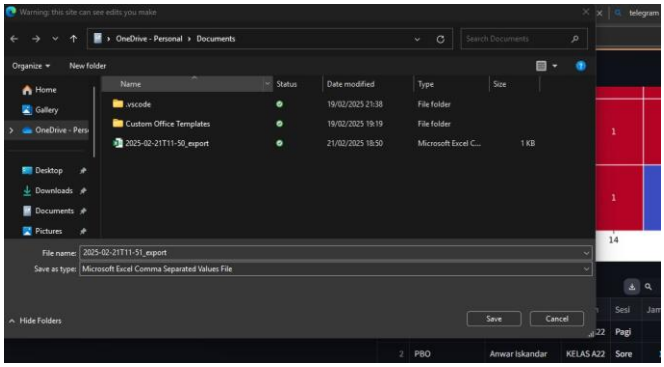


Figure 1.6 - Optimization Results data can be downloaded for further use

No	Mata Kuliah	Dosen	Ruangan	Sesi	Jam
1	PBO	Joko Tingkir	KELAS A22	Pagi	7
2	PBO	Anwar Iskandar	KELAS A22	Sore	14
3	RPL	Aji Subroto	KELAS A33	Sore	14
4	RPL	Heru Sabar	KELAS A55	Pagi	7
5	Alg Lanjut R2	Aji Subroto	KELAS A11	Pagi	11
6	Machine Learning	Heru Sabar	KELAS A22	Sore	16
7	Deep Learning	Rivaldo Pakpahan	KELAS A11	Sore	16
8	Deep Learning	Rivaldo Pakpahan	KELAS A33	Pagi	11
9	Sistem Operasi	Ferdy Firmansyah	KELAS A11	Sore	18
10	Sistem Operasi	Ferdy Firmansyah	KELAS A22	Pagi	9
11	Pemrograman Lanjut	Ayu Astuti	KELAS A11	Pagi	7
12	Struktur Data	M.Iskandar Lawoly	KELAS A11	Pagi	9
13	Struktur Data	Monika Anisa	KELAS A55	Sore	14
14	Web 1	Ratna Handoko	KELAS A55	Pagi	9
15	Web 1	Hesti Purwadinata	KELAS A22	Sore	18
16	Pem Lanjut R1	M.Firly Tubagus	KELAS A33	Sore	18
17	Pem Lanjut R1	Ayu Astuti	KELAS A33	Sore	16

Figure 1.7 - Successfully downloaded Optimization results

2. CONCLUSION

3.1 Conclusion

This research shows that **Linear Programming** can optimize class division by ensuring a more balanced distribution of students, efficient classroom utilization, and avoiding schedule conflicts between classes, lecturers, and rooms. This application helps increase learning effectiveness by reducing the inequality in the number of students per class and optimizing the allocation of academic resources.

However, there are some **challenges**, such as **computational complexity**, especially if the number of courses, lecturers, and lecture sessions increases. This application also does not take into account **the preferences of lecturers and students**, and has the potential to produce classes with an uneven number of students if it does not match the maximum capacity multiples. **Limited resources**, such as the number of rooms and teaching staff, can also affect optimization results.

Overall, this research contributes to **data-driven academic management**, supporting decision-making in a more efficient and systematic class division. In the future, further development is needed to increase the flexibility of the Application and reduce the complexity of calculations so that it can be applied more widely and optimally.

3. REFERENSI

[1] D. Wungguli and N. Nurwan, "Penerapan Model Integer Linear Programming dalam Optimasi Penjadwalan Perkuliahan secara Otomatis," Jurnal Matematika dan Aplikasinya, vol. 28, no. 1, pp. 281-285, 2021.

[2] Z. Mahrijal, A. Sumarsa, and M. Widyastiti, "Optimasi Penjadwalan Mata Pelajaran Menggunakan Metode Integer Linear Programming (Studi Kasus: SMA Al-Hikmah)," Jurnal Teknik Informatika dan Sistem Informasi, vol. 9, no. 2, pp. 155-167, 2020.

[3] D. S. Anggraini, Syaripuddin, and Q. A'yun, "Optimasi Penjadwalan Menggunakan Pemrograman Linier Integer pada Masalah Penjadwalan Perawat UPT Dinas Kesehatan Puskesmas Jonggon Jaya," Jurnal Sistem Informasi dan Teknik Industri, vol. 4, no. 2, pp. 112-125, 2020.

[4] R. Fitriani, T. Santoso, and M. Wahyudi, "Optimasi Penjadwalan Dosen Menggunakan Pemrograman Linier di Universitas XYZ," Jurnal Informatika dan Riset Operasi, vol. 12, no. 1, pp. 34-45, 2020.

[5] L. Kurniawan, B. Sugiarto, and A. Pratama, "Model Optimasi Pembagian Kelas dengan Pendekatan Integer Linear Programming," Jurnal Teknologi dan Manajemen Pendidikan, vol. 15, no. 2, pp. 89-102, 2019.

[6] R. Rachmatika, "Penerapan aplikasi program linear dengan menggunakan metode simpleks untuk mendukung kegiatan UMKM," Kajian Ilmiah Informatika dan Komputer, vol. 3, no. 2, pp. 194-202, 2022.

[7] A. T. Kusuma, R. A. Pertiwi, and D. S. Sari, "Implementasi Algoritma Genetic Algorithm dalam Optimasi Penjadwalan Kuliah," Jurnal Teknologi Informasi dan Ilmu Komputer, vol. 7, no. 3, pp. 456-467, 2021.

[8] M. I. Pratama and S. H. Putra, "Penggunaan Simulated Annealing untuk Optimasi Jadwal Perkuliahan," Jurnal Rekayasa Sistem dan Teknologi Informasi, vol. 10, no. 2, pp. 98-110, 2020.

[9] B. Wijayanto, "Optimasi Penjadwalan dengan Metode Integer Linear Programming pada Industri Manufaktur," Jurnal Teknik Industri dan Manajemen, vol. 5, no. 1, pp. 23-30, 2019.

[10] R. Hendrawan, A. P. Nugroho, and T. S. Putri, "Penerapan Machine Learning dalam Optimasi Penjadwalan Karyawan," Jurnal Sistem Cerdas dan Informatika, vol. 8, no. 1, pp. 67-80, 2021.