Optimizing Socket Allocation for Enhanced Laptop Usage in Resource-Constrained Study Environments

ASSAF Mohammad Mahdi S319819

Politecnico di Torino

Abstract

This study addresses the optimization of socket allocation in a university study room to maximize student satisfaction by prolonging laptop usage. We developed an optimization model that we solved using the Gurobi solver, a heuristic algorithm, and a hybrid algorithm that combines both methods to manage socket allocation, laptop usage, and battery levels effectively over a defined operational period. The heuristic algorithm provided great results up to the tested cases, offering a significant reduction in computational effort compared to the Gurobi solver. The hybrid algorithm further enhanced efficiency by using the heuristic solution as an initial guess for Gurobi, speeding up the convergence process. Our experiments revealed a linear relationship between the number of students (N) and the minimum number of sockets (s) required to ensure continuous laptop usage. These findings provide valuable insights for practical applications, particularly in environments with limited computational resources and time constraints.

Table of Contents

1 Problem Definition

In Aula Studio Verdi, students rely heavily on their laptops, but the availability of electrical sockets is limited. Our goal is to develop an optimal strategy for socket allocation to maximize laptop usage time.

We aim to ensure the longest possible minimum operational period among all students.

Given the operational constraints and varying battery levels, it is essential to manage the allocation of sockets effectively. By developing an efficient allocation strategy, we can significantly enhance student satisfaction and productivity in resource-limited environments.

2 Modeling of the Problem

To address the socket allocation problem, we define the following parameters, decision variables, objective function, and constraints:

2.1 Parameters

- N: Number of students.
- s: Number of available sockets.
- T: Total operational time.
- Δ_T: Time interval for each decision point.
- $B_{i,0}$: Initial battery level of student i.
- r_i: Recharge rate of student i's laptop.
- d_i: Discharge rate of student i's laptop.

2.2 Decision Variables

- $U_{i,t}$: Binary variable indicating whether student i is using their laptop during interval t.
- $Y_{i,t}$: Binary variable indicating whether student i is charging their laptop during interval t.

2.3 State Variables

- $B_{i,t}$: Continuous matrix where $B_{i,t}$ represents the battery level of student i's laptop after time interval t.

$$B \in \mathbb{R}^{N \times \left(\left\lceil \frac{T}{\Delta_T} \right\rceil + 1\right)}$$

- Z: Integer representing the minimum number of intervals a student was able to use their laptop during the entire time period T.

$$Z \in \mathbb{Z}$$

- A: Continuous variable representing the average usage time across all students and intervals.

$$A \in \mathbb{R}$$

2.4 Objective Function

The objective is to maximize the combined measure of the minimum operational period and the average usage time for all students:

$$\max(Z + A)$$

2.5 Constraints

1. **Battery Level Update**:

$$\begin{aligned} B_{i,t} = B_{i,t-1} &+ Y_{i,t-1} \cdot r_i \cdot \Delta_T \\ &+ Y_{i,t-1} \cdot d_i \cdot \Delta_T \\ &- U_{i,t-1} \cdot d_i \cdot \Delta_T \quad \forall i,\ \forall t - \{0\} \end{aligned}$$

2. **Battery Capacity**:

$$0 \leq B_{i,t} \leq 100 \quad \forall i,\ \forall t$$

3. **Socket Usage**:

$$\sum_{i=1}^{N} Y_{i,t} \leq s \quad \forall t$$

4. **Minimum Usage Intervals**: Ensure that Z is the minimum number of intervals a student is able to use their laptop.

$$Z \leq \sum_{t=1}^{\lceil \frac{T}{\Delta_T} \rceil} U_{i,t} \quad \forall i$$

5. **Average Usage Time**: Ensure that A represents the average usage time across all students and intervals.

$$A = \frac{1}{N \cdot \lceil \frac{T}{\Delta_T} \rceil} \sum_{i=1}^{N} \sum_{t=1}^{\lceil \frac{T}{\Delta_T} \rceil} U_{i,t}$$

6. **Charging and Usage Constraint**: Ensure that a student charging their laptop is also using it.

$$U_{i,t} \geq Y_{i,t} \quad \forall i,\ \forall t$$

2.6 Logical Explanation

Originally, we aimed to maximize the minimum battery level for any student at any time interval, i.e.,

$$\max \min B_{i,t}$$

However, this posed a problem: for some settings, the problem became infeasible because there was no solution that could guarantee the battery levels would remain above 0 for the entire time period T.

To address this issue, we introduced the binary variable U, where a student i is not allowed to use their laptop during time interval t if $U_{i,t} = 0$. This modification allows us to find a feasible solution. Consequently, the objective function was redefined to maximize the minimum number of intervals a student was able to use their laptop during the entire period.

However, to further enhance the allocation strategy, we incorporated the average usage time A. Thus, the new objective function is:

$$\max(Z + A)$$

where Z represents the minimum number of intervals a student was able to use their laptop during the entire period, and A represents the average usage time across all students and intervals. A was tailored to be a value between 0 and 1, ensuring that the scheduling prioritizes increasing the minimum guaranteed usage time for a student over increasing the overall average usage time. By adding A, the solver attempts to further increase student satisfaction by improving the average usage time across all students, once the minimum usage time can no longer be increased.

3 Methodology

3.1 Initialization

Variables were initialized with realistic values:

- **Recharge Rate:** Recharge rates were assigned based on a normal distribution with a mean of 33.33% per hour and a standard deviation of 5%. Values were clipped to be within 10% and 50%.
- **Discharge Rate:** Discharge rates were assigned based on a normal distribution with a mean of 20% per hour and a standard deviation of 5%. Values were clipped to be within 10% and 30%.
- **Initial Battery Levels:** Initial battery levels were assigned using a uniform distribution between 20% and 80%.

3.2 Gurobi Optimization

For the Gurobi Optimization algorithm, the following steps were taken:

1. **Model Initialization:** The Gurobi model was initialized with the appropriate parameters and decision variables.
2. **Objective Function:** The objective function was set to maximize the Fair and Maximized Usage Score ($Z + A$).
3. **Constraints:** Constraints were added to ensure that the battery dynamics were respected, the number of students using sockets did not exceed the available sockets, and the battery levels stayed within the valid range. Additionally, Z was constrained to be at most equal to the minimum total usage time, and A was defined as the average usage time across all students and intervals.
4. **Optimization:** The model was optimized using Gurobi's solver, and the results were extracted, including the Fair and Maximized Usage Score, the allocation matrices **Y** and **U**, and the battery levels matrix **B**.

This approach ensured that the problem was formulated correctly and solved efficiently using Gurobi's optimization capabilities.

3.3 Heuristic Optimization

For the heuristic algorithm, the following steps were followed:

1. **Initialization:** Initial values for the decision variables and battery levels were set. Initial

battery levels for all students were recorded in B_{matrix}, with the first column representing the initial battery levels.

2. **Battery Calculation:** The current battery levels were updated using the equation:

$$\begin{aligned} B[i,t] = B[i,t-1] &+ Y[i,t-1] \cdot r[i] \cdot \Delta_T \\ &+ Y[i,t-1] \cdot d[i] \cdot \Delta_T \\ &- U[i,t-1] \cdot d[i] \cdot \Delta_T \end{aligned}$$

3. **Forecasting Battery Levels:** The next battery level for each student was forecasted assuming no charging:

$$B[i,t+1] = B[i,t] - d[i] \cdot \Delta_T$$

4. **Initialize Operational Status:** The operational status **U** is set to 1 for all students with forecasted battery levels above zero.

5. **Socket Allocation:**

   - Sockets were allocated to students based on their forecasted battery levels. Students with forecasted battery levels below zero were chosen and the others were dropped.
   - If the number of such students was less than or equal to the available sockets, all such students were allocated sockets.
   - Otherwise, students were sorted based on the sum of their operational status up to the current time, their forecasted battery levels, and the difference between their recharge and discharge rates. The lowest-ranked students were then allocated sockets.

6. **Distribute Remaining Sockets:** Remaining sockets were distributed to students without sockets in the current time slot, prioritizing those with the lowest forecasted battery levels and ensuring their battery levels did not exceed 100%.

7. **Update Operational Status:** The operational status **U** was updated for students receiving sockets, ensuring they could use their laptops during the allocated intervals.

4 Heuristic vs Gurobi Data

4.1 Experimental Setup

Our experiments involved the following steps:

1. For $N = 39, 40$ (number of students), we incrementally increased the number of sockets s starting from 1 up to the number that guarantees continuous laptop usage.

2. For each combination of N and s, we executed the heuristic and Gurobi algorithms for 10 different seeds to ensure consistent results. For the Gurobi solver, we limited the execution time using a time limit of 180 seconds.

3. We recorded the minimum usage time Z, optimization time, and model build time for each algorithm and seed.

4. The results for each combination of N and s were collected for all the seeds to obtain multiple records per combination.

5. For each s, we computed the minimum, maximum, and average optimization time for Gurobi and the heuristic algorithm. Additionally, we recorded the model build time for Gurobi only as the heuristic algorithm does not have a separate model build time.

Figures 2 and 4 show the minimum number of usage intervals a student has received among all the students Z and the average number of usage intervals per student M for the Gurobi and heuristic algorithms.

Figures 1 and 3 record the optimization times for Gurobi and heuristic, as well as the Gurobi model build time.

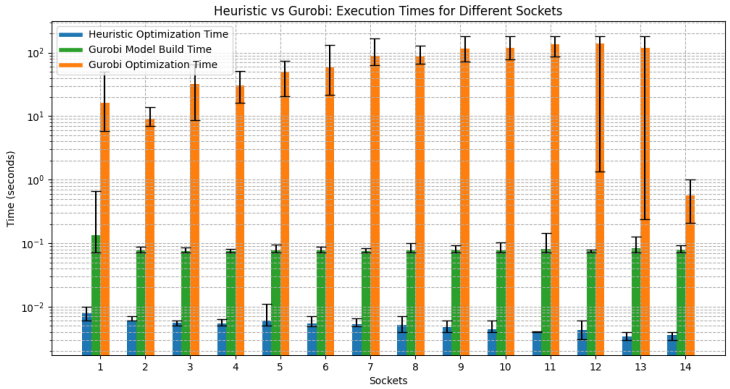

Figure 1: Execution Times Comparison for N=39

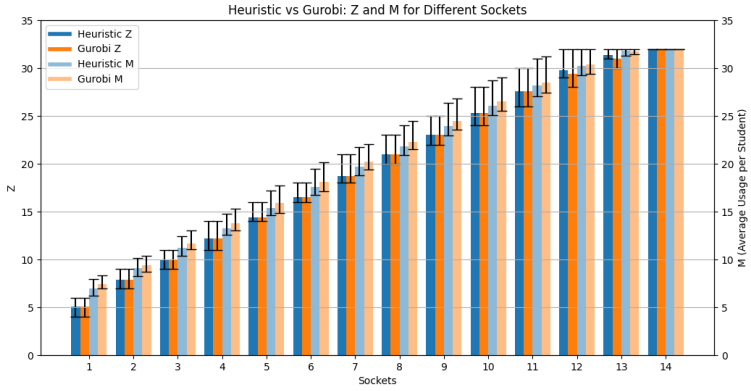

Figure 2: Performance Comparison for N=39

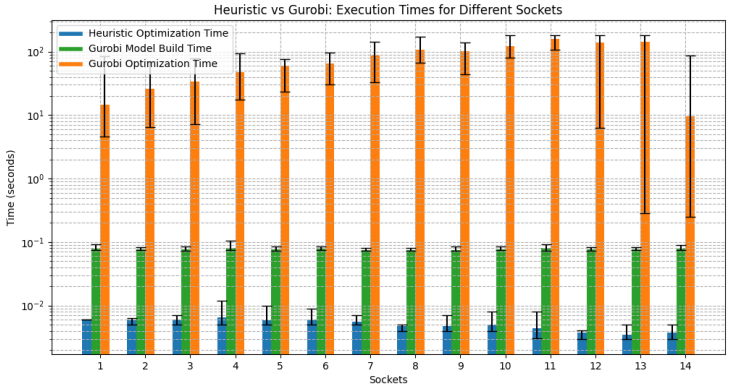

Figure 3: Execution Times Comparison for N=40

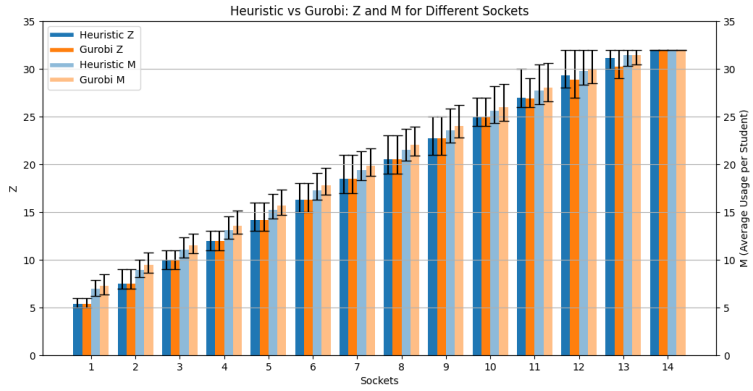

Figure 4: Performance Comparison for N=40

4.2 Performance Comparison

Heuristic Optimization Time: The heuristic algorithm consistently required less optimization time compared to the Gurobi solver. As shown in Figures 1 and 3, the heuristic optimization time remained relatively low and stable across different values of N, demonstrating its efficiency and scalability.

Gurobi Optimization Time: The Gurobi solver exhibited significantly higher optimization times. This highlights the computational expense associated with exact optimization methods for larger problem sizes.

Optimization Results: Importantly, the heuristic algorithm was able to provide solid results which were either comparable to or even better than the Gurobi solution given the time limit.

Figures Explanation: The figures show the averages in the bars, and the minimum and maximum values as error bars. This helps to visualize the variability and the range of the results for both algorithms across different seeds.

4.3 Allocation Strategy Comparison

In this section, we provide a detailed comparison of the socket allocation strategies used by the heuristic algorithm and the Gurobi solver. This comparison aims to highlight the discrepancies in scheduling sockets between the two algorithms.

To better visualize these discrepancies, we conducted a new experiment where a very simplified instance was solved using both the heuristic and the Gurobi algorithms. The results include matrices B (battery levels across the time intervals of the students), U (indicating laptop usage), and Y (indicating connection to a socket for charging).

In Figures 5 and 6, for the case of $N = 10$, with $s = 4$ and $s = 2$ respectively, seed = 1, $T = 16$ hours, and $\Delta T = 1$ hour, yellow cells represent intervals where laptop usage is possible (red otherwise), and green cells represent intervals where the laptop is connected to a socket for charging.

The results demonstrate that in Figure 5, while both algorithms achieved the optimal solution, the Gurobi solution did not utilize all available sockets during time intervals 1 and 2. This suggests that the heuristic algorithm attempts to allocate all available sockets in every scenario, whereas the Gurobi solver does not necessarily use all sockets to achieve optimal results.

On the other hand, Figure 6 shows how the Gurobi solution, by providing the optimal solution, outperforms the heuristic allocation. While maintaining the same minimum number of usage intervals a student has received among all the students (Z), the Gurobi optimal solution guarantees a higher average number of usage intervals per student with the possibility to use their laptops.

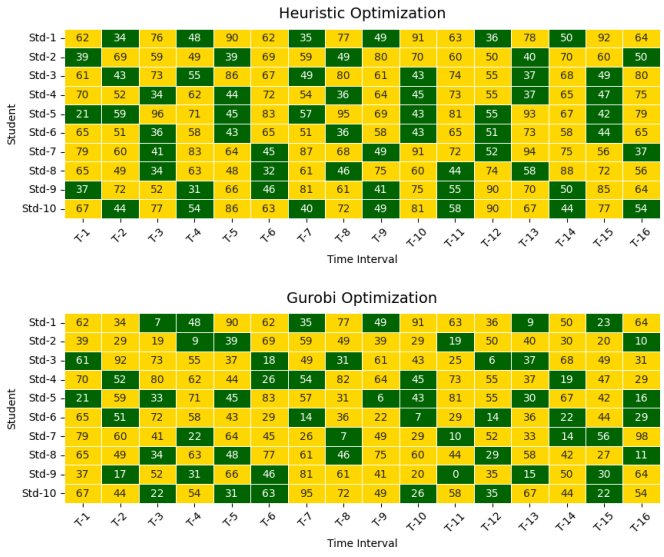

Figure 5: Allocation Comparison for $N = 10$, $s = 4$, seed = 1, $T = 16$ hours, and $\Delta T = 1$ hour

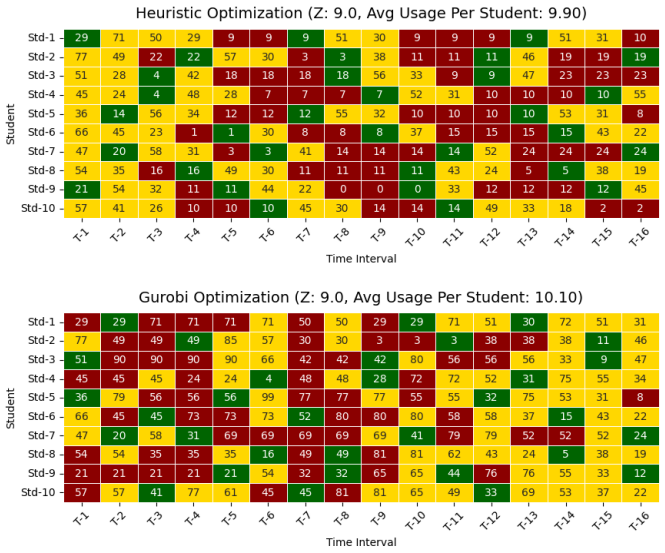

Figure 6: Allocation Comparison for $N = 10$, $s = 2$, seed = 1, $T = 16$ hours, and $\Delta T = 1$ hour

4.4 Conclusion

Our comparative analysis shows that the heuristic optimization algorithm is a practical and efficient alternative to the Gurobi solver for the problem of optimizing laptop charging schedules. The heuristic algorithm's ability to provide consistent and good results, coupled with its significantly reduced optimization times, makes it well-suited for real-world applications where computational resources and time are limited.

In summary, the heuristic algorithm is strongly comparable to the optimization performance of Gurobi given the time limit, while also offering substantial improvements in computational efficiency. This makes it a valuable tool for managing laptop charging in resource-constrained environments. Additionally, the heuristic algorithm attempts to allocate all available sockets in every scenario, whereas the Gurobi solver does not necessarily utilize all sockets to achieve optimal results.

5 Heuristic Findings

In this section, we investigate the relationship between the number of students (N) and the number of sockets (s) required to achieve a minimum usage time (Z) that equals the total number of intervals ($\left\lceil \frac{T}{\Delta T} \right\rceil$) (i.e continuous laptop usage). Our goal is to deter-

mine the minimum number of sockets necessary to ensure that all students can use their laptops for the entire operational time T.

To achieve this, we implemented our heuristic algorithm and ran experiments varying the number of students from 10 to 10000. For each value of N, we incrementally increased the number of sockets s starting from 1 until we found the minimum s that allows all students to achieve $Z = \left\lceil \frac{T}{\Delta T} \right\rceil$. This process was repeated 10 times for each (N, s) with different random seeds to ensure the consistency of the results.

The steps for each experiment were as follows:

1. **Initialization:** For each value of N, we generated the initial battery levels, recharge rates, and discharge rates for all students using normal distributions with predefined means and standard deviations. This ensures that the values are realistic and varied.

2. **Increment s:** Starting from $s = 1$, we incremented the number of sockets until we found the minimum s that allowed all students to achieve the required minimum usage time $Z = \left\lceil \frac{T}{\Delta T} \right\rceil$.

3. **Consistency Check:** For each value of N and s, we repeated the experiment 10 times using different random seeds to ensure the robustness of our results. We recorded the optimization time for each run.

4. **Result Compilation:** For each N, we recorded the optimal s and the 10 execution times, compiling the data into a comprehensive table for analysis.

Through this investigation, we aim to identify a pattern or relationship between N and s, which can provide valuable insights for optimizing socket usage in scenarios with varying numbers of students and available resources.

The detailed results of this investigation are presented in the Figures 7 and 8:

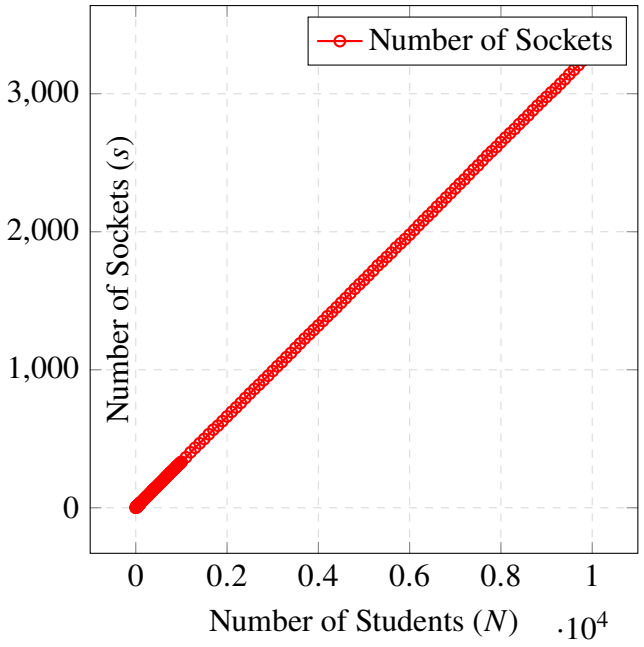

Figure 7: Number of Students vs Number of Sockets

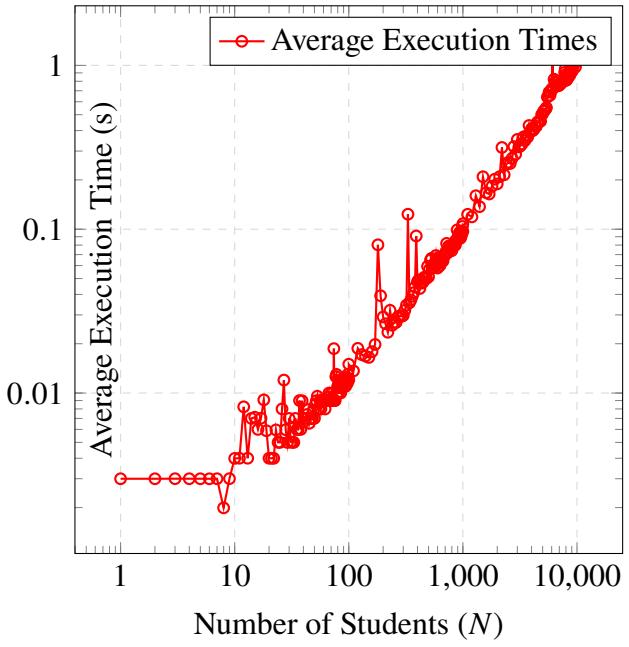

Figure 8: Number of Students vs Average Execution Time

6 Case Study

In this case study, we examine a scenario at Verdi Aula Studio, which has the capacity to accommodate 200 students and is equipped with 52 sockets. The operational hours of the facility are 16 hours, and for the sake of practicality, we set $\Delta T = 0.5$ hours.

We initially attempted to find the optimal socket allocation using the Gurobi optimization algorithm. However, even after 24 hours of continuous execution, the algorithm had not yet converged to a solution. This highlighted the need for a more efficient approach to solving the problem.

To address this issue, we developed a hybrid algorithm that combines the heuristic optimization with the Gurobi optimizer. The hybrid algorithm first executes the heuristic optimization to generate an initial guess, which is then used as the starting point for the Gurobi optimization. This significantly speeds up the convergence process.

Our analysis revealed that in order to ensure continuous laptop usage for all 200 students, an additional 13 sockets need to be installed, bringing the total number of sockets to 65.

Below, we present the allocation results for the Verdi case study using both the heuristic and hybrid algorithms. The figures illustrate the usage (U), and charging (Y) patterns over the operational period.

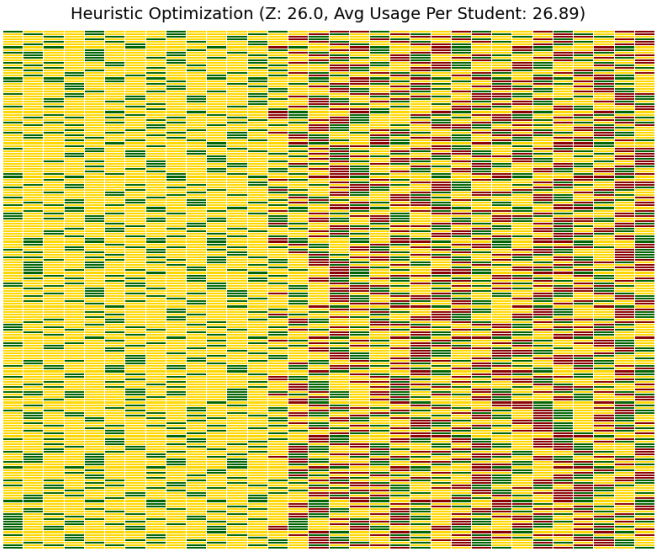

Figure 9: Socket Allocation Results for Verdi Case Using Heuristic Algorithm

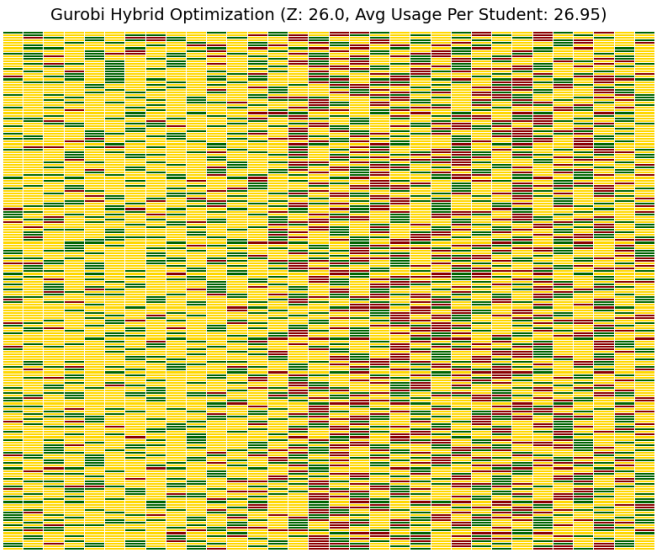

Figure 10: Socket Allocation Results for Verdi Case Using Hybrid Algorithm

Additionally, we present the allocation results for the scenario where the number of sockets is increased to 65:

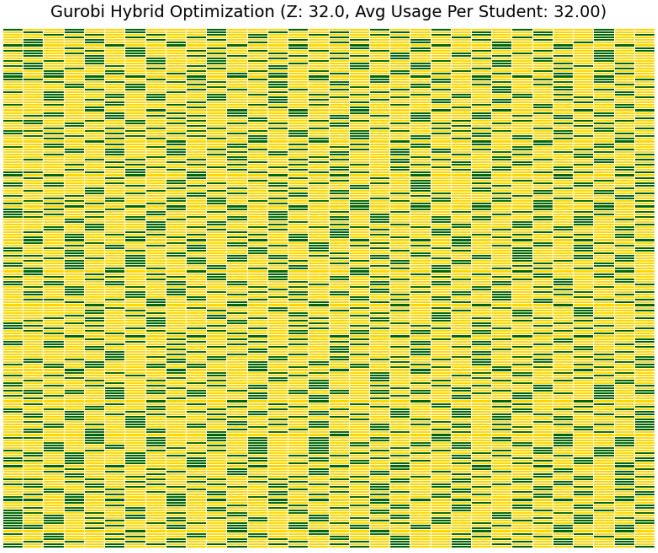

Figure 11: Socket Allocation Results for Verdi Case with 65 Sockets Using Hybrid Algorithm

These figures show the laptop usage (U), and charging (Y) over the 16-hour operational period with 0.5-hour intervals. The hybrid algorithm successfully achieves the optimal solution within a reasonable timeframe.

7 Discussions

In this section, we delve into the insights and implications derived from our optimization study. The primary aim of our research was to develop efficient methods for allocating resources (sockets) to maximize laptop usage among students in a study room environment. During the course of our research, we also discovered a relationship between the number of students (N) and the minimum number of sockets (s) required to ensure that all students can use their laptops continuously for the entire operational period T. Several key observations and discussions emerged from our experiments:

7.1 Resource Allocation and Relationship Between N and s

Our findings indicate a clear relationship between the number of students and the required number of sockets. As expected, as the number of students increases, the number of sockets required to ensure continuous laptop usage also increases. The relationship observed is approximately linear, with the number of sockets increasing as a fraction of the number of students. This linearity is significant because it suggests that needed sockets scales predictably with the increase in the number of students, allowing for straightforward resource planning as student numbers grow.

7.2 Consistency and Robustness of the Heuristic Algorithm

The heuristic algorithm demonstrated consistent performance across different values of N. By conducting 10 runs for each N with different random seeds, we ensured that the results were not influenced by specific random instances. The consistency in the optimal s values across different seeds underscores the robustness of our heuristic approach. Additionally, the recorded optimization times were reasonably low, suggesting that the heuristic is computationally efficient even for large values of N.

7.3 Practical Implications

From a practical standpoint, our findings provide valuable guidelines for managing laptop charging in environments with limited sockets, such as study rooms or libraries. Facility managers can use the insights from our study to allocate an optimal number of sockets based on the expected number of users. This ensures efficient use of resources and maximizes the operational time for students' laptops, enhancing their productivity. For instance, in a library with 100 students, understanding that approximately one-third of that number in sockets would suffice can streamline resource allocation.

7.4 Limitations and Future Work

While our heuristic approach offers significant insights, it is essential to acknowledge its limitations. The model assumes that all students have similar usage patterns and that the discharge and recharge rates are constant over the course of the day. In reality, variations in user behavior and device specifications could affect the results. Future research could extend this work by incorporating more complex user behavior models and testing the algorithm in real-world scenarios.

Additionally, while our heuristic optimization has proven effective, there may be room for further improvements or alternative heuristic methods that could offer even better performance. Future work could explore different heuristic strategies, machine learning-based approaches, or hybrid methods combining heuristic and exact optimization techniques.

7.5 Comparison with Exact Optimization

A comparative analysis between our heuristic approach and exact optimization methods (e.g., using Gurobi) revealed that while the exact methods guarantee optimal solutions, they can be computationally expensive for large N. Our heuristic, on the other hand, provides comparable solutions with significantly reduced computational overhead, making it suitable for practical applications with large numbers of students. Additionally, Gurobi's solutions do not necessarily use all the available sockets, while

the heuristic attempts to allocate all sockets to ensure usage.

7.6 Impact of Δ_T Choice

The choice of Δ_T (0.5 hours) played a crucial role in our model. A smaller Δ_T allows for finer granularity in socket allocation but increases the computational complexity. Conversely, a larger Δ_T simplifies the computation but may lead to less precise allocation. Future studies could explore the impact of different Δ_T values on the model's performance and computational efficiency.

In summary, our heuristic optimization approach provides a practical and efficient solution for managing laptop charging in environments with limited resources. The insights gained from our study can guide facility management in optimizing socket allocation, ultimately enhancing user satisfaction and productivity.

8 Conclusions

This study aimed to optimize laptop charging schedules in environments with limited electrical sockets, such as study rooms, by developing a heuristic algorithm and comparing its performance with exact optimization methods using the Gurobi solver.

8.1 Key Findings

The research revealed several important insights:

- **Linear Relationship Between N and s**: Our experiments demonstrated a linear relationship between the number of students (N) and the minimum number of sockets (s) required to ensure continuous laptop usage. This finding provides a straightforward guideline for facility managers to allocate an adequate number of sockets based on the expected number of users.
- **Heuristic Algorithm Efficiency**: The heuristic algorithm consistently provided very good solutions compared to the optimal provided by Gurobi with significantly reduced computational times. This makes the heuristic approach practical for real-world applications, especially when dealing with large numbers of students.
- **Optimal Socket Allocation Strategy**: By implementing a strategy that prioritizes students with the lowest battery levels and considering both recharge and discharge rates, the heuristic algorithm effectively maximizes the operational time of laptops. This ensures that the available sockets are utilized efficiently.
- **Scalability and Robustness**: The heuristic algorithm's performance remained consistent across different values of N, showcasing its scalability and robustness. The algorithm's efficiency was validated through multiple runs with different random seeds, ensuring the reliability of the results.

8.2 Practical Implications

The findings from this study have significant practical implications. Facility managers can use the insights to make informed decisions about socket allocation in study rooms, libraries, or any other environments with limited charging resources. By ensuring that the number of sockets is approximately one-third of the expected number of users, managers can optimize resource usage and enhance user satisfaction.

8.3 Future Work

While our heuristic optimization approach has proven effective, there are several avenues for future research:

- **Complex User Behavior Models**: Incorporating more complex user behavior models, including varying usage patterns and device specifications, could provide a more accurate representation of real-world scenarios.
- **Alternative Heuristic Strategies**: Exploring different heuristic strategies, machine learning-based approaches, or hybrid methods combining heuristic and exact optimization techniques could further improve the performance and applicability of the algorithm.
- **Impact of Δ_T**: Investigating the impact of different Δ_T values on the model's performance

and computational efficiency could offer insights into the trade-offs between granularity and complexity.

- **Real-World Testing**: Testing the heuristic algorithm in real-world scenarios would validate its practical applicability and identify any potential challenges or limitations.

In conclusion, this study provides a comprehensive approach to optimizing laptop charging schedules in resource-constrained environments. The developed heuristic algorithm offers a practical and efficient solution, with the potential to significantly enhance resource management and user satisfaction in various settings.

9 Code Repository

The complete code for this project is available in a public GitHub repository. The repository can be found at the following link:

`https://github.com/mahdi-assaf/OR-Project.git`

The repository includes all the scripts, data, and documentation necessary to reproduce the results presented in this report.