Problem Chosen
E

2025
IMMC
Summary Sheet

Team Control Number
25902858

Data Center Scheduling: A Multi-Objective Dynamic Optimization Approach

Summary

This paper presents a novel optimization model for scheduling computational tasks and electricity usage in a data center over a 24-hour period. The model addresses the challenge of balancing the increasing demand for computational power with the need to minimize energy costs and maximize the use of renewable energy sources. Unlike previous studies that focused on resource allocation without considering different power sources, our model incorporates both traditional and green energy, along with their respective costs and carbon emissions.

We introduce carbon tax to quantify the environmental impact of traditional energy and prioritize green energy usage. The model operates on two different time scales – hourly and time periods task are delivered in– to provide finer-grained scheduling and ability to adapt to dynamic task arrivals and power fluctuations. We consider two different explanation on real-time computational demand: suitable to different data inputs and adapt with arrivals of unexpected tasks by time periods.

We explore two optimization conditions: prioritizing task completion without delays and prioritizing minimizing costs. We utilize linear programming and dynamic programming techniques to achieve these objectives, respectively. Further, we combine these objectives by introducing a delay penalty and propose a "Dynamic By-Steps Optimization" method to dynamically adjust the schedule and power allocation throughout the day. Finally, we conduct sensitivity analysis to both a provide a general picture in fluctuations to consider in real-life scenario and enhance the stability in real-time power scheduling by calculating number of inversions.

Overall, the results from our modified model demonstrate significant cost savings and improved green energy utilization compared to our initial no-delay scenario. As model output, we provide an hourly allocation of tasks and ratio of green to traditional energy use. The model is flexible and adaptable to different data center parameters, offering a practical solution for optimizing data center operations while promoting sustainability and cost-effectiveness.

Keywords: Optimization, Dynamic Programming, Multi-Objective Modeling, Computational Demand, Carbon Emission, Energy Consumption

Table of Contents

1 Introduction

1.1 Background

The demand for computational power is increasing rapidly with the development of information technology. The computing power market is expected to even increase be 6.8% over the next decade[1]. However, each computer has its own computing capacity, which is the maximum number of tasks it is able to swiftly and effectively carry out per unit time[1]. Computing capacity is commonly masked under or generalized to the energy use capacity, or the maximum amount of energy the computer can use at that moment. Power consumption also leads to environmental problems, since traditional energy, like coal and natural gases, causes significant amount of carbon emission[2]. But complete transition to renewable energy, such as solar and wind energy, is implausible. Although the costs of renewable energy are actually lower, there are limited supplies that is dependent to the time of the day[3].

It then becomes important to find a balance between the completion of tasks and the energy use capacity, while also keeping in mind the cost of different. The difficulty increments when each task consumes different amount of energy and has different level of urgency. Currently, studies had been done on various regarding optimal resource allocation. Specific to computational power, a 2016 study[4] developed an algorithm that by considering time, cost, and processor requests to efficiently assign resources to different tasks. Albeit its completeness and attentiveness, it did not address managing different power sources.

In this paper, we aim to create an optimization model that schedules the order of completion of tasks, while considering the urgency, the energy use capacity and the choice of energy source. We will concentrate the problem to one specific data center, while being able to generalize to all data centers by changing parameters. We consider the two objectives (minimize delay and energy cost) initially respectively with opposite priorities, using Linear Programming's concept and Dynamic Programming's algorithm. Then, we combine them through quantification of important variables. We finally implement the model to a dynamic environment, using our original method Dynamic By-Steps Programming. As the output of our model, we provide an hourly allocation of tasks and ratio of green to traditional energy use.

1.2 Problem Decomposition

This study completes the following tasks:

For Task 1, we build an optimization model that aims to schedule the tasks so all tasks are completed within the 24-hour period, despite the changes in power conditions. The input is the amount and urgency of tasks for the upcoming time period (from 2 to 6 hours) and the remaining

tasks from previous time period. The output is the total energy consumption and total energy cost for the upcoming hour.

The model will be made up of two parts. The task-demand section schedules the order of completion based on demand. The constrains to consider are the task priority, deadline, and energy use capacity. We quantify delayal penalty and combine it with the total cost. The energy-allocation section allocates power to tasks, while considering power consumption is linear with respect to computational load and each task have specific energy consumption. With enough power, it will complete task in the order of high, medium and low urgency, or in other words, in descending order of weights.

For Task 2, within the model, we introduce a condition based on dynamic programming that aims to minimize cost of electricity while maximizing green energy usage. It consists of two major variables: amount of green energy and amount of traditional energy used per time period. It considers the following constrains: total power usage cannot exceed supply limits and prioritizing green energy. We set a traditional energy penalty to quantify green energy usage and directly link it to the total cost. It will output the ratio of green energy to traditional energy usage for each hour that minimizes the cost.

For Task 3, we construct an original, efficient scheduling algorithm called Dynamic By-Steps Programming that balances computational power demand and electricity supply from Task 1 and optimize the cost by considering the results from Task 2. We quantify delay penalty as a part of total cost and combine energy cost with delayal. We also transfer the model from an environment where total tasks are known to a dynamic environment with new introduced tasks. We finally present sensitivity analysis to demonstrate the stableness of our mode.

1.3 Overview of Our Work

Our work is presented from Section 2 to 8. This paper is organized as followed: Section 2 discusses the assumption innately in problem and proposed for the model; Section 3 presents the key variables used in the equations and codes; Section 4 demonstrates the preliminary conditions, logic and of the model; Section 5 showcases the two perspectives of our multi-objective model; Section 6 modifies the model to fit in a dynamic environment; Section 7 performs sensitivity analysis and discusses the model's strengths and extensions; finally, Section 8 concludes the paper.

2 Assumptions and Justifications

Considering the data center proposed in the task, there are several innately given assumptions not presented as part of the data set:

(1) There is a fixed upper limit to the total renewable (green) energy available.

(2) While both energy sources may be used, green energy is the prioritized energy source.

(3) Power consumption by a server is linear to its computation load.

(4) Tasks have different levels of importance that must be respected.

(5) The optimization focuses on a 24-hour period.

(6) The time taken to complete a task may be ignored. It can be treated that with enough power, all tasks can be completed within that hour.

In addition to the assumptions provided within the task, to clarify the problem, we propose a series of additional assumptions:

Assumption 1: All tasks may be delayed over the time period they are assigned in, they must be completed within the 24-hour period, in order to meet the ultimate goal. High urgency tasks will be punished more than medium and low urgency tasks if they are delayed.
Justification: Since the optimization focuses on a 24-hour period, the goal of the model is to allow all tasks be completed within this time. High urgency task, being more critical than medium or low, is more sensitive to and affected by delayal. Thus, if more high urgency tasks are delayed, it will be less optimal.

Assumption 2: The schedule will be provided on an hourly basis. There is no difference in order of completion of tasks if all are completed within the same hour.
Justification: One hour is the smallest unit of consideration. No smaller unit can be used because the data provided are all specific to hour. The fluctuation of power generation supply and prices within an hour is unknown. They can be considered as discrete values. Therefore, it is not necessary to differentiate task competition order for a smaller unit of time.

Assumption 3: Although task demands may change on a time to time basis, the total number of tasks and their urgency for the current time period (ranging from 2 to 6 hour, according to the Task Quantity table) will be fixed.
Justification: As shown in the table that presents task quantity, tasks are assigned to intervals of 2 to 6 hours. Within each interval, the pre-destined number of tasks and urgency are fixed, so any real-time change will mirror that table. Assuming a fixed number of tasks and urgency within a short period both simplifies the optimization problem while still allowing for adjustments in the subsequent periods.

Assumption 4: For tasks that were not completed in the previous time period, their weights will increase linearly over time. The process order will be descending order of weights, not necessarily by urgency. When the weight of delayed lower urgency tasks has higher is higher than the medium urgency tasks of the current time period, the data center will process the delayed, high weighted low urgency tasks first.
Justification: Since tasks are given by time periods (Assumption 3), they are planned to be completed in that time period, so delayal should be punished. Assuming weights accumulation to

be linear initially simplifies the model. However, the model is free for modification in the weight accumulation algorithm.

Assumption 5: A single task must be completed in one specific hour. Tasks currently performing cannot be carried on to the next hour. Task completion is discrete.
Justification: Since all values provided are discrete, including the power consumption of one task, it is reasonable to not separate the completion of one task into sections. Also, task can be completed instantaneously, so the time do not need to be considered.

Assumption 6: Energy cannot be accumulated. Energy use for one hour can only be from the energy generated in that hour.
Justification: The data center model provided in the task description does not include a battery energy storage system, which inherently precludes the possibility for energy accumulation. The model focuses on short-term (hourly) optimization. Data centers need to cope with dynamic and rapid changes, so focusing on optimizing energy use of within one hour is more practical than considering energy storage and retrieval. Also, assuming no energy accumulation simplifies the model, which allows concentration on core aspects, such as task urgency and energy price.

3 Variables

Variable	Definition (Unit)	Unit
t	Time of the day	Hour
p	Time Period	N/A
$E_G(t)$	Available Green Energy Power at t	MW
$E_T(t)$	Traditional Energy Power used at t	MW
$P_G(t)$	Price of Green Energy at t	Yuan per kWh
$P_T(t)$	Price of Traditional Energy at t	Yuan per kWh
$C(p)$	Total Cost of Energy at p	Yuan
$D_H(p), D_M(p), D_L(p)$	Computational Demand for High, Medium, and Low-priority Tasks at p	kWh
$A(t)$	Assigned Computational Power at t	kWh
U_H, U_M, U_L	Weights of High, Medium, and Low-priority Tasks	N/A

Table 1. Key Variables in the Model

4 Model Setups

Figure 1 shows the general logic of our aim, by stating the inputs and output of each task. However, our paper is not strictly organized to presents the three tasks in order, but structured to show gradual advancement in the model. Task 1 and Task 2 approach the problem from two

perspectives, which will be respectively addressed in Section 5.1 and Section 5.2. The "energy penalty" and "delay penalty" will be introduced in Section 4.1 and Section 6.1 Both Task 1 and Task 2 will output the energy costs every hour based on the scheduling of tasks the computer will perform. Task 1 aims for minimal delay and completion of assigned tasks on time, while Task 2 targets the solution that generate the smallest energy cost. Finally, Task 3 incorporates the perspective from 1 and 2 and is addressed in Section 7 through quantification in Section 6 and sensitivity analysis in Section 7.

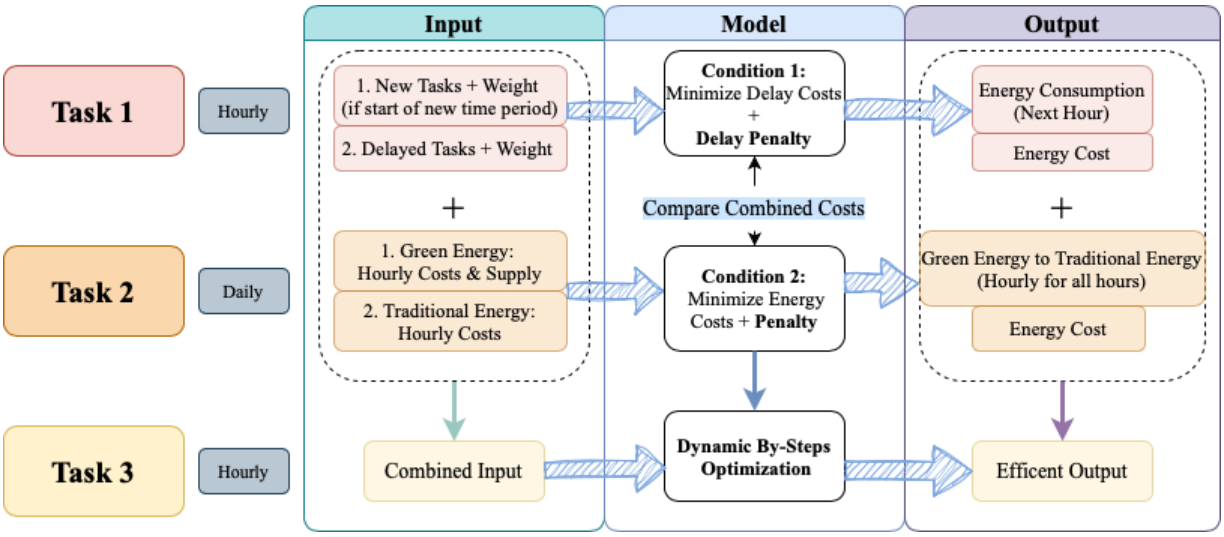

Figure 1. Input and Output by Tasks

Initially, we planned to use Linear Programming (LP)[5] that finds the best solution, or the best scheduling of tasks, while satisfying all constraints and optimizing the object function, or the urgency subtracted by the energy cost. However, standard (LP) assumes variables are continuous, while in our case all variables are discrete integer values, according to Assumption 5. Also, typically there are one object for LP, but we have to consider both urgency and energy cost. Their relationship cannot be simplified to scalar addition, as they may have different weights and order of magnitude. In addition, our model aims to cope with dynamic environment, where the total number of tasks for the 24-hour period is unknown. LP assumes all parameters are certain, including coefficients to object and the total number of tasks. It cannot optimize if the number of tasks for the next time period is unknown, which is the case for real-time optimization.

Nevertheless, some concepts in LP are still useful, especially objective functions. One of the objectives is energy cost, which also requires considering energy penalty. We notice that prioritizing green energy is almost completely fall under the condition with minimal cost, as green energy. Only costing more during the first three hours will not make significant difference when compared to the penalty for using traditional energy over green energy. The penalty is defined in Section 4.1.

4.1 Penalty of Energy Source Selection

We define the penalty for using traditional energy (natural gas, coal, etc.) over renewable energy by the international carbon tax[7], which is defined by this equation:

$$Carbon\ Tax\ Cost = Carbon\ Tax\ Rate\ \times Carbon\ Tax\ Intenity$$

The Carbon Tax Rate in the European Union is approximately €80-100 per ton of CO_2. In Canada, in 2024 it is \$80/ton CO_2. However, the carbon tax has not yet been implemented nationally in China. Thus, by finding a mean value of other nation's tax rate, we decide to assume Carbon Tax Rate to be 50 USD/ton CO_2. The Carbon Emission intensity is $0.9kgCO_2/kWh$ for coal power and $0.4kgCO_2/kWh$ for natural gas[8].

Therefore, for coal power, the Carbon Tax Cost is:

$$\$50/ton\ CO_2 \times (0.9kgCO_2/kWh)\ = \$0.045/kWh\ (\approx ¥0.32/kWh)$$

For natural gas, it is:

$$\$50/ton\ CO_2 \times (0.4kgCO_2/kWh)\ = \$0.02/kWh\ (\approx ¥0.14/kWh)$$

By taking the average of both energy sources, we get approximately $¥0.2/kWh$ as the Carbon Tax Cost that will be carried on in our calculations.

4.2 Non-Dynamic Environment

Our model aims for coping with real-time changes in task amounts and power supply, while tasks are delivered to fixed time periods. This brings two different understanding in real-time computational demand:

(1) The model should be flexible to be suitable for different parameters, meaning that algorithm should be delivered in the form of variables. The model can generate schedule specific to the computational demand and energy costs of the day. However, the data center will know the number of tasks given to each time period in advance, or at the beginning of the day. No new tasks will be unexpectedly added during any time.

(2) The model should be able to cope with unknown quantity of computational demand. The data center only know the computational demand of the current time period, but will have to adapt to the changes in demand for the upcoming periods, meaning that optimization should be done specific to each time period. However, as presented in Assumption 3, computational demand will be given at the start of each time period, not hours.

We initially consider Explanation (1), which does not involve optimization by steps. Section 5 presents two solution, in increasing order of optimization, to the problem in a non-dynamic environment.

5 Multi-Objective Optimization Model

For the ultimate optimization, we have two objectives: (1) to minimize the delay and (2) to minimize the cost of energy, while considering penalty from Section 4.1. These objectives are parallel and of similar importance. The problem now transitions to how to balance the two objectives, which cannot be done through LP. Instead of finding or assigning weights to balance the two objectives, we decide to consider the two extreme conditions, where completing on time or reducing cost is considered first, respectively. It is important to note that both conditions do

not only address one of the two objectives. Both are considered but with complete opposite priorities.

To demonstrates the differences in these two conditions, we used the same computational demand by time periods, hourly energy costs and hourly green energy supply. These data come from the tables within the question pdf. All data are reorganized in the form of table in Appendix 1 and 2. Since the model can adapt to real-time changes, as presented in Explanation (1), by altering the inputs, the model will generate the schedule specific to that circumstance.

Figure 2 shows a general modification and upgrading procedure of our model, beginning from stable environment Condition 1 in Section 5.1 and Condition 2 in Section 5.2, ending with dynamic optimization in Section 6.

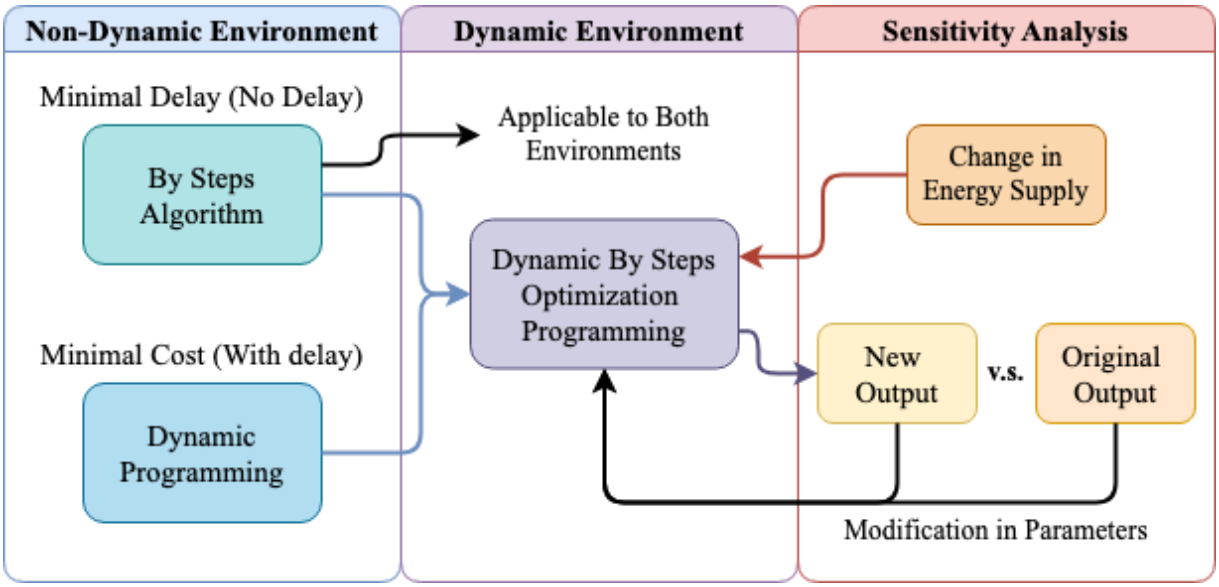

Figure 2. Flowchart Showing Model Improvement Procedure

5.1 Condition with No Delays

In this condition, Objective 1 is prioritized, so all tasks will be completed during the time period that it is assigned in. Under the premise that there are no delays, task completion order and choice of energy source will aim to minimize the cost of energy. For each time period (denoted by p and number from 1 to 7, the energy consumption of all tasks will be accumulated:

$$D(p) = D_H(p) + D_M(p) + D_L(p)$$

Then, we calculate the total green energy cost for time period p (from time m to n), which equal the accumulation of the supply multiplied by the cost at each hour:

$$C_G(p) = \sum_{t=m}^{n} (E_G(t) \times P_G(t))$$

Since we green energy costs less and is prioritized, only the remaining required power will be generated by traditional energy. And traditional energy will be used for the hour within the time period where traditional energy is cheapest. The cheapest time is t_{min}. The total cost will be:

$$C(p) = C_G(p) + \{[D(p) - \sum_{t=m}^{n} E_G(t)] \times P_T(t_{min})\}$$

The complete code of this model is in Appendix 3. The final output is the energy consumption for both green energy (GE) and traditional energy (TE) by hours, as shown in Table 2. The hourly costs are calculated by multiplying hourly consumption with the cost of energy at that hour, which is presented in Table 3. The total cost, compared with output of Condition 2, is shown again in Table 4.

Hour	GE /MW	TE /kWh	Hour	GE /MW	TE /kWh
0	–	–	**12**	3.4	–
1	–	–	**13**	3.3	9064.0
2	–	–	**14**	3.1	16151.0
3	–	–	**15**	2.9	–
4	–	–	**16**	2.6	–
5	–	–	**17**	2.5	–
6	1.8	4847.0	**18**	2.3	–
7	2.1	–	**19**	1.5	–
8	2.4	12711.0	**20**	1.0	–
9	2.4	–	**21**	–	7696.0
10	2.8	–	**22**	–	–
11	3.2	–	**23**	–	1450.0
Total					

Table 2. Energy Consumption by Hour

Hour	GE /¥	TE /¥	Hour	GE /¥	TE /¥
0	–	–	**12**	1020.0	–
1	–	–	**13**	1320.0	11783.2
2	–	–	**14**	1550.0	19381.2
3	–	–	**15**	1450.0	–
4	–	–	**16**	1300.0	–
5	–	–	**17**	1250.0	–
6	720.0	4847.0	**18**	1380.0	–
7	840.0	–	**19**	900.0	–
8	960.0	15253.2	**20**	600.0	–
9	720.0	–	**21**	–	9235.2
10	840.0	–	**22**	–	–
11	960.0	–	**23**	–	6156.8
Total	10813.2	66656.6	**Combined Total:** 77469.8		

Table 3. Energy Costs by Hour

The condition can be viewed as the simplest scenario, but it is also applicable to the dynamic environment. All calculations of cost per time period are independent from the computational demand of the upcoming time periods because it aims for no delays.

5.2 Condition with Minimal Energy Costs

On the other hand, Condition 2 considers Objective 2 first, where tasks may be delayed to any time within that day when energy is the cheapest. After finding the distribution of power by hours, delayal will be minimized by completing high urgency tasks first.

We decided to use Dynamic Programming (DP)[6] to present this condition. DP can tackle with overlapping subproblems (energy cost and delay). It begins with defining a State, which is all essential information used to decide at the current hour. DP is beneficial because it can address the fluctuating power prices by incorporating them into the cost function of the recursive relation.

We inputted the task by hour and hourly energy cost in the form of a table. Since with penalty traditional energy is always more costly than green energy, the former is named "expensive energy", while the latter is "cheap energy." The complete code is in Appendix 4. Figure 3 shows the general logic of the code.

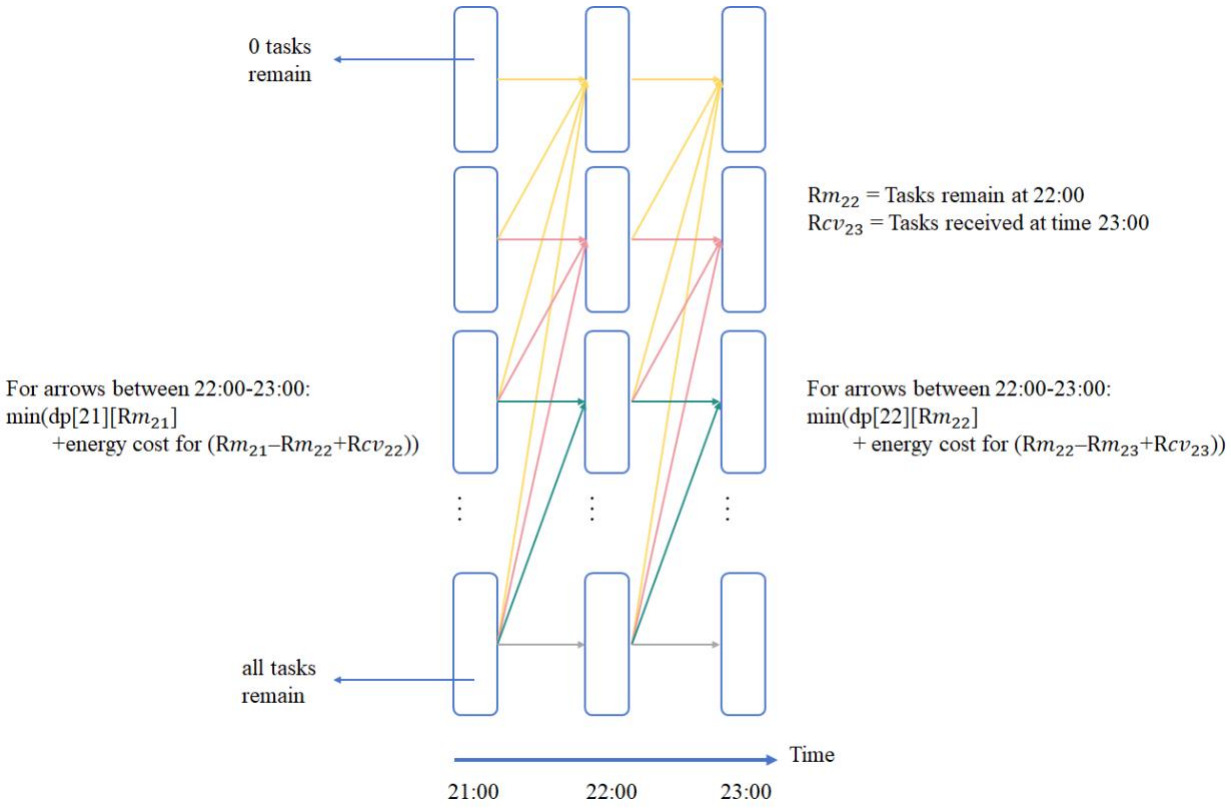

Figure 3. Dynamic Programming Concept Diagram

Then, we construct dynamic programming table dp, which stores the minimal cost to achieve the computational demand. $dp[i][k]$ represents in the ith hour, with tasks worth j more computational demand incomplete, the minimal amount of money spent for the entire process. It is iterated through each hour (from hour 1 to 23). For each hour, it iterates through different levels of computational demand (j) and calculates the minimum cost ($dp[i][k]$). The calculation involves the choice of using green or traditional energy. We also incorporate an inner loop (k) that simplifies the algorithm, by allowing the minimal cost of the current demand level (j) to be transitioned from of a high demand level (where $k > j$) and using the optimal solution from dp_{ik}.

To summarize the procedure in a mathematical way, our model uses a State Transition Equation. The minimum cost $dp[i][k]$ can be calculated using the minimum cost $dp[i-1][k]$ at hour $i-1$ with k kWh of remaining tasks, combined with the energy cost required for the current hour's tasks. The state transition equation is as follows:

$$dp[i][k] = \min_{k>j}(dp[i-1][k] + energy\ cost\ for(k - j + tasks[i]))$$

In which, $tasks[i]$ is the number of tasks (in kWh) in hour i, and $energy\ cost\ for(k - j + tasks[i])$ is the energy cost required to process the current hour's tasks, given the remaining task demand (k kWh at hour $i-1$) and the current hour's demand (j kWh at hour i).

Condition	Total Cost /¥
Condition 1	77469.8
Condition 2	29992.0

Table 4. Total Cost Under Same Computational Demand

Finally, we output the minimum cost to satisfy the computational demand in total. The minimal cost, along with the minimal cost from Condition 1 are presented in Table 4.

6 Dynamic By-Steps Optimization Model

6.1 Delay Penalty Quantification

Since this version of the model presents Objective 2 as energy costs, we planned to also measure delay with delay penalty costs. The next step is to quantify delay penalty. We define it as:

$$penalty = delay\ hours \times delay\ index \times urgency\ index$$

$delay\ index$ is a parameter that can be modified. There is not fixed value for $delay\ index$ because the weight of delay in the optimization model is not fixed. As mentioned in Section 5, our model is a multi-objective model, where we consider two objectives by different priorities. Adding in delay penalty is to balance the two objectives, while delay index places delay penalty to a similar level of magnitude and energy costs, and minor changes alters the weights between the two objectives. For the calculations in Section 6.2, we use:

$$delay\ index = 0.2$$

urgency index identifies the urgency level of the tasks, with higher index for more urgent tasks, which means higher urgency tasks will have greater delay penalty, as already suggested in Assumption 1. However, these indexes can be modified for specific needs. To summarize, we set the penalty to be linear, meaning a certain amount will be charged for every hour of delay. The delay penalty for "medium" priority tasks is different from that of "low" priority tasks. The penalty for "medium" is higher so that the data center will tend to minimize delays for "medium" tasks over "low" tasks for economic benefits.

Therefore, the combined optimized version incorporates quantified version of delay penalty, so it can be combined with energy costs as part of the objective function.

6.2 Modified Model

Based on Assumption 3 and Explanation (2), the data center only knows the number of tasks and the amount of power consumption for the current time period, as shown in Figure 4. After inputting new tasks and tasks not yet completed for the current time period, we input the current time period, which is when the optimization begins, denoted by the purple text in Figure 4. Green energy is prioritized and directly consumed within the current time period to the maximum extent possible, as shown in the solid rectangles in both Figure 4 and Figure 7.

If the green energy is less than the power required to complete the tasks, the remaining power will be supplemented by traditional energy. The computational demand to be satisfied is shown by the lined rectangles in Figure 7. However, the use of traditional energy can be either within the time period or outside of the time period (delayed). If traditional energy outside the time period is used, a delay penalty, explained in Section 6.1 will be incurred.

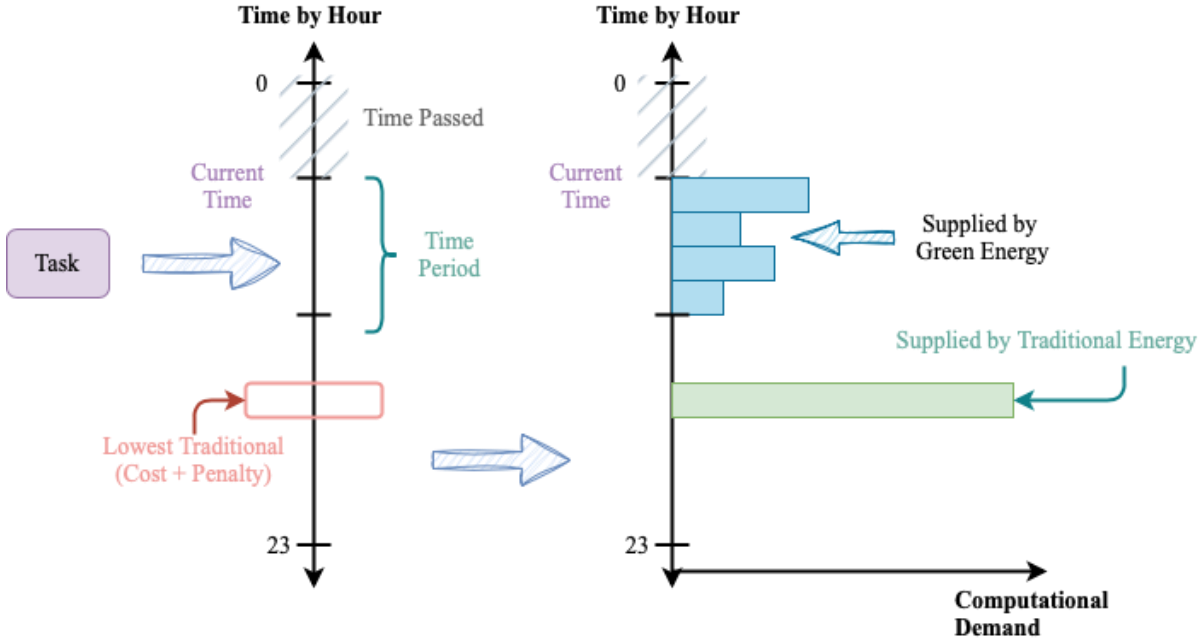

Figure 4. General Flowchart of Modified Model in Processing New Tasks

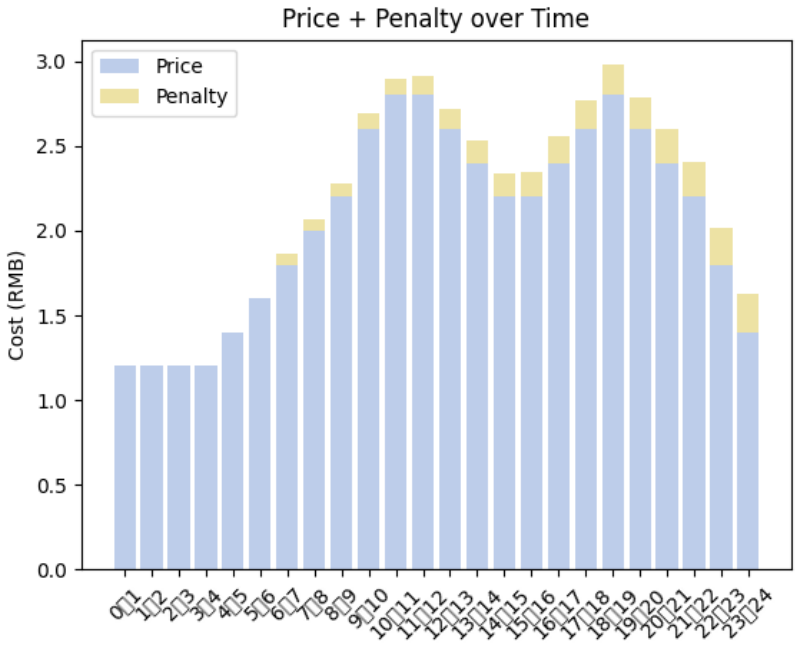

Figure 5. Traditional Energy Price + Penalty of Low Urgency Tasks Given in Time Period 1

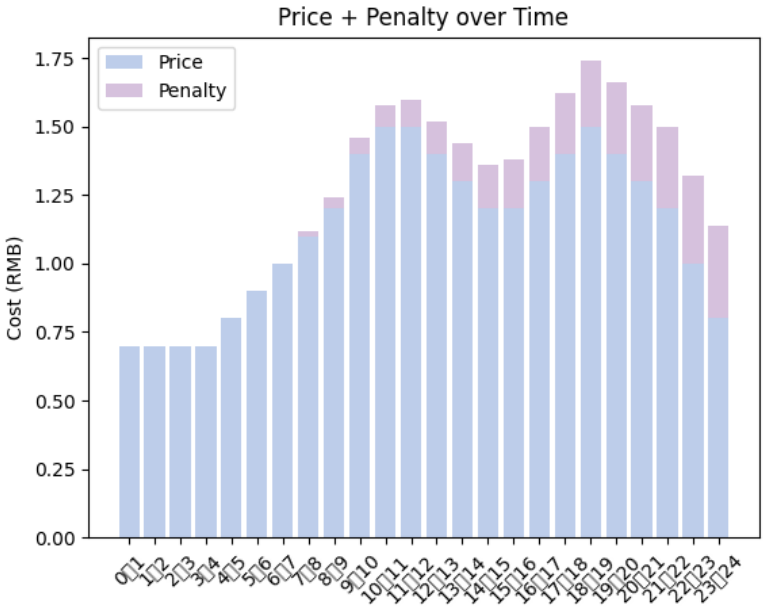

Figure 6. Traditional Energy Price + Penalty of Medium Urgency Tasks Given in Time Period 1

We find the sum of the price of traditional energy (after carbon tax) for future time periods and the penalty for delaying to that hour is the lowest for each time period, from the input starting time to the end of the day (including within and outside the current time period). For a low urgency task given in the first time period (0 to 6 hours), the sum over time period is shown in Figure 5. Similarly, a medium urgency task's sum is presented in Figure 6.

By comparing the sums, we find the cheapest time period for traditional energy, and use the traditional energy at that time to complete the remaining tasks. This is shown as the right part of Figure 7, where all incomplete tasks are stacked up on the cheapest time period. Finally, we calculate the total cost.

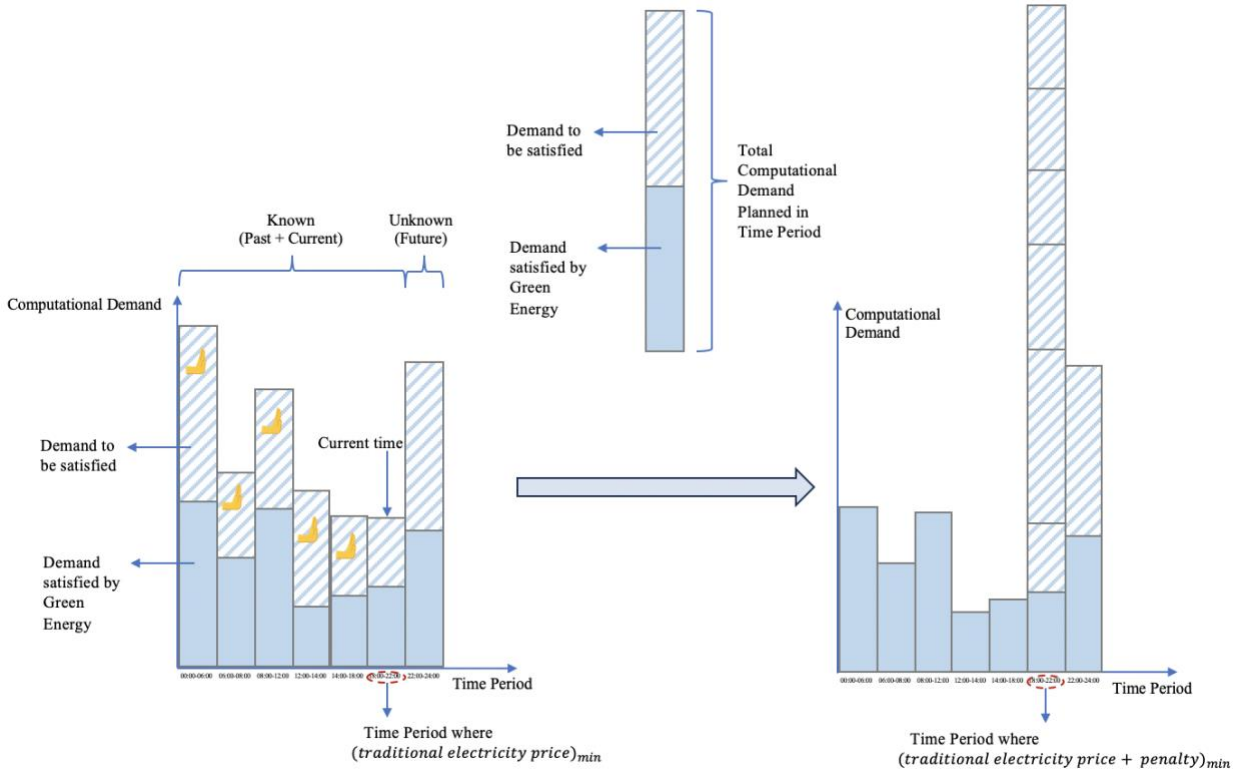

Figure 7. Algorithm Flowchart of Dynamic Environment with Delay Penalty

7 Sensitivity Analysis

7.1 Enhancing Real-time Model

We conduct the sensitivity analysis by introducing variabilities. First, **Green Energy Fluctuation** is implemented. To simulate real-world variability, we added random number between -10% and 10% to the hourly green energy supply. This represents the unpredictable nature of renewable energy sources. Then, we proposed the **Traditional Energy Supply Cap**, which is an upper limit is imposed on the hourly supply of traditional energy to simulate situations with abundant power availability.

Based on these variabilities, we optimized the algorithm to achieve the dual goals of task completion and cost savings. First, similar to the original model, green energy is prioritized due to its lower cost and environmental benefits. Second, remaining tasks are allocated to traditional energy sources, but we consider the hourly supply cap for traditional energy. This means that even the hour with the lowest combined price and penalty might not have enough traditional energy to fulfill the remaining demand to its entirety. Third, the algorithm iteratively allocates

tasks to the next cheapest hour with available traditional energy, continuing until all tasks are assigned. Finally, if all traditional energy sources are exhausted before all tasks are assigned, an energy shortage scenario occurs. This situation demonstrates the limitations of relying solely on real-time decisions and the potential need for longer-term planning or energy storage solutions. The procedure is shown in Figure 8. The complete code is in Appendix 6.

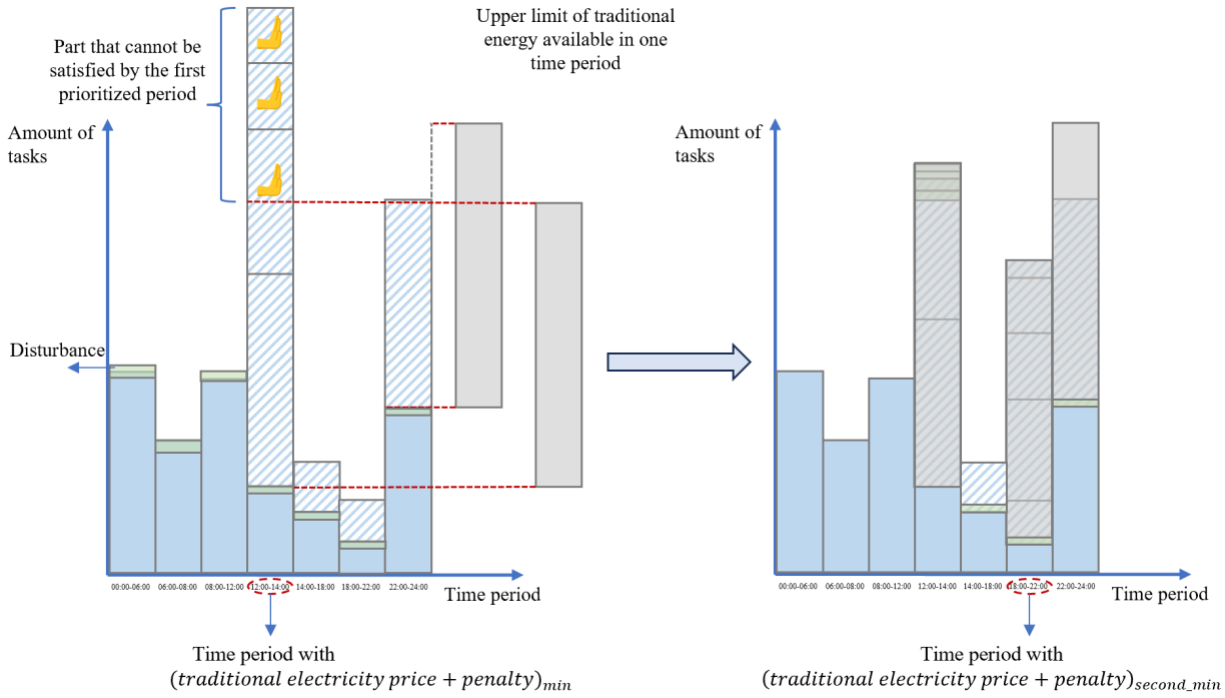

Figure 8. Model Concept Graph with Fluctuation and Traditional Energy Supply Cap

7.2 Enhancing Stability in Real-time Power Scheduling

In Section 7.1, we improved real-time power scheduling. This power scheduling system produces a power scheduling scheme, which is the tasks to complete in each time period and how much clean and traditional energy to consume. To optimize the stability of the model, we enable rapid response and adjustment of the computational load in situations with large fluctuations in power supply. Since the supply of clean energy is relatively unstable, we added a 10% perturbation to the clean energy supply for each time period and then input it into our model again. The model then produces a new scheme that adapts to the new scenario (with perturbation). Subsequently, we propose a “stability" metric. By comparing the similarity between the new scheme and the old scheme, we provide a stability value. The degree of stability is determined by three secondary criteria: (1) task completion, (2) the number of inversions in the task execution sequence, and (3) the difference in resulting costs.

For (1) task completion, we believe that all tasks on the list must be completed, so the output of this criterion is only divided into 100% (all completed) and 0% (not all completed). Throughout the scheme iteration, we need to ensure that the task completion rate is 100%.

For (2) the number of inversions in the task execution sequence, we believe that the smaller this value, the more similar the task execution sequence of the new and old schemes, that is, the less affected by disturbances. The number of inversions I can be shown by the following equation:

$$I(P_1, P_2) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} 1\big(pos(P_1, t_i) > pos(P_1, t_j)\ and\ pos(P_2, t_i) < pos(P_2, t_j)\big)$$

In which, $pos(P_1, t_i)$ represent task t_i's position in sequence P_1, and $pos(P_2, t_i)$ represent task t_i's position in sequence P_2. $1(\cdot)$ is an indicator function that will return 1 when the condition inside the parenthesis is true and return 0 when false.

In terms of the (3) difference in resulting costs, we update the scheme with the vision that the lower the resulting cost (total electricity expenditure), the better. The smaller the difference between the two schemes, the stronger the robustness and adaptability of the original scheme.

Combining these indicators, because 100% completion is the bottom line of our scheme, we simplify the model into a dual-objective model (i.e., cost and sequence similarity). We obtain the "similarity" value between the new scheme and the old scheme by manually weighting and adding them. We use Random Walk Algorithm to continuously iterate and optimize the similarity value of the algorithm. In this way, even if the actual situation encountered by the model is different from the expected situation, the model has sufficient ability to withstand these changes. After the parameters converge, we determine the final model to be put into use. At this time, the model can still exhibit anti-interference and robustness under large power supply differences.

7.3 Model Strengths & Extensions

Based on the sensitivity analysis and the general logic of our model, we summarize the following strengths:

1. **Our model has significant flexibility, allowing it to be suitable for different environment, energy supply and computational demand.** Our model addresses two different explanations of "real-time computational demand." The first one is our model can generate schedule specific to the computational demand and energy costs of the day. It can also cope with unknown total quantity of demand, adapting to new workloads given at each time period

2. **Our model incorporates the strengths of multiple modeling techniques to provide a holistic presentation of the solution.** It stems from the concept of objective functions commonly used in linear programming. However, we noticed the drawbacks of linear programming, being unsuitable for multi-objective optimization. We decided to consider the two different priorities when addressing the two objective functions. For the one which energy cost is prioritized over delay, we used dynamic programming. Finally, we combined the two objectives function by quantifying delay penalty and adding a delay index that can be changed.

3. **Our model introduces two quantifications of variables that significant impact our objectives.** We quantified Carbon Tax to balance green and traditional energy and Delay Penalty to measure the impact of task delayal.

4. **Our model has clear and comparable output.** Instead of having several sets of outputs for each of our objectives, by quantifying the two variables, we allowed one single output, which is the total cost. The total costs include both energy cost, carbon tax and delay penalty, which addresses all parts of consideration. Scheduling can be directly assessed by directly comparing their total cost.

However, there are several possible extensions for the model. The model is currently tested only with a data center of the same level of magnitude. It could be brought to verify whether it can generate schedules for data centers of a larger scale. It can also be extended to balance not only two types of energy, but multiple. For example, green energy can be separated to solar and wind energy, which has significant different supplies at the same hour.

The problem itself and our group made several assumptions that simplifies, also in a way limits, the modeling. For example, we can consider the scenario where energy can be accumulated and stored. The scope of the problem will not be limited to 24 hours. Energy can be stored from the previous day and used on the next day immediately to avoid delayal penalty. Also, it is assumed that tasks can be completely instantaneously. If we consider the time it takes to complete a task, there will be an upper limit for computational demand of each time period. This also means tasks are not discrete; instead, one task can be completed by parts across multiple hours.

8 Conclusion

In addition to being the first model to balance different energy sources, along with providing optimal scheduling, our model achieves the following points:

(1) Our model considers two time-scales: hour and time periods. With tasks introduced by time periods, we can provide scheduling by a smaller scale — hourly.

(2) We quantified the constrain of prioritizing energy costs by introducing Carbon Tax, so the total hourly cost of traditional energy is the sum of its original cost and carbon tax.

(3) We balanced the two objectives by considering them in two different priorities. In Condition 1, we prioritize delay penalty by assigning tasks in a way they are all completed in the time period they are assigned in, then minimize the cost. In Condition 2, we prioritize minimizing cost, through Dynamic Programming.

(4) We combined the two goals by introducing delay penalty as part of the total cost. We maximize green energy use first. By comparing the new total cost of traditional energy, we find the hour from current time to the end of the day where it is the lowest. We propose Dynamic By-Step Optimization, that does dynamic optimization for each time period.

(5) Our model is flexible and stable. Even though we used the data from one data center, our model can cope with changes in perimeters and can be extended to other data centers.

Our results show for our target data center, under Condition 1, where no delay is allowed, the lowest cost is 77469.8 yuan; under Condition 2, where delay is allowed and not penalized, the minimal cost is 29992 yuan; under the combined and dynamic condition, the lowest cost is dependent to real-time inputs. By gradually increasing difficulties and combing conditions, we end up with a multi-objective, dynamic and stable model to optimize scheduling of computational power and electricity over a 24-hour period. We welcome further explorations, such as verifying our model for other data centers and introducing the possibility of energy storage and accumulation. All in all, our model is a valid solution to provide scheduling for data centers, in order to maximizing green energy usage, reducing carbon emissions, and lowering electricity costs.

References

1) Pangarkar, T. (2023, December 19). *Computing Power Statistics: A Boost for Technology*. Market Scoop. https://scoop.market.us/computing-power-statistics/

2) Sullivan, W. (2022, October 25). *Renewable Energy Is Slowing the Rise of Carbon Emissions*. Smithsonian Magazine. https://www.smithsonianmag.com/smart-news/renewable-energy-is-slowing-the-rise-of-carbon-emissions-180980988/

3) University of Wisconsin. (n.d.). *Section B: Comparing Renewable and Non-Renewable Energy Costs - Renewable Energy Education | UWSP*. Www3.Uwsp.edu. https://www3.uwsp.edu/cnr-ap/KEEP/nres635/Pages/Unit2/Section-B-Comparing-Renewable-and-Non-Renewable-Energy-Costs.aspx

4) Gouda, K C & T V, Radhika & Akshatha, M. (2013). Priority based resource allocation model for cloud computing. International Journal of Science, Engineering and Technology Research 2278-7798. 2. 215-219.

5) Bertsimas, D., & Tsitsiklis, J. N. (1997). *Introduction to linear optimization* (Vol. 6, pp. 479-530). Belmont, MA: Athena scientific.

6) Rust, J. (2008). Dynamic programming. *The new Palgrave dictionary of economics*, *1*, 8.

7) International Energy Agency. (2024). *IEA - The global energy authority*. Iea.org. https://www.iea.org/

8) World Bank. (2024). *Carbon Pricing Dashboard | Up-to-date overview of carbon pricing initiatives*. Carbonpricingdashboard.worldbank.org. https://carbonpricingdashboard.worldbank.org/

Appendix

[1] https://github.com/SinoHero/When-Watts-Meet-Bits-The-Power-Computing-Collaborative-Scheduling/blob/main/Tasks.csv

[2] https://raw.githubusercontent.com/SinoHero/When-Watts-Meet-Bits-The-Power-Computing-Collaborative-Scheduling/refs/heads/main/Costs.csv

[3] https://github.com/SinoHero/When-Watts-Meet-Bits-The-Power-Computing-Collaborative-Scheduling/blob/main/NoDelay.py

[4] https://raw.githubusercontent.com/SinoHero/When-Watts-Meet-Bits-The-Power-Computing-Collaborative-Scheduling/refs/heads/main/JustDelayv2.py

[5] https://raw.githubusercontent.com/SinoHero/When-Watts-Meet-Bits-The-Power-Computing-Collaborative-Scheduling/refs/heads/main/real_time.py

[6] https://raw.githubusercontent.com/SinoHero/When-Watts-Meet-Bits-The-Power-Computing-Collaborative-Scheduling/refs/heads/main/optimization_of_real_time.py