December 2018

Cost-Effective Load Scheduling for Hybrid Renewable Energy Systems

Avinash Shashikala Rajendra
University of Wisconsin-Milwaukee

COST-EFFICIENT LOAD SCHEDULING FOR HYBRID RENEWABLE ENERGY SYSTEMS

by

Avinash Rajendra

A Thesis Submitted in

Partial Fulfillment of the

Requirements for the Degree of

Master of Science

in Computer Science

at

The University of Wisconsin-Milwaukee

December 2018

ABSTRACT

COST-EFFICIENT LOAD SCHEDULING
FOR HYBRID RENEWABLE ENERGY
SYSTEMS

by

Avinash Rajendra

The University of Wisconsin-Milwaukee, 2018
Under the Supervision of Professor Jun Zhang

Hybrid renewable energy systems offer great promise for the future. However, some lingering concerns regarding stability and cost efficiency still exist. If a private party installs the system and maintains full control, the party may itself alleviate some of these problems by wisely optimizing the benefits offered by the system. One of the ways to do so is to develop a schedule for their load such that the cost incurred is minimized; this is done by maximally utilizing the renewable sources of energy before using the backup options of more conventional energy sources. Creating such a schedule involves considering several factors, such as solar energy available and the quantity of load that may be flexibly scheduled as opposed to fixed demands. This work presents a unique and innovate method – dynamic programming – to solve this problem. This is modeled in a mathematical context, one of optimal control, and then implemented using MATLAB. Care is taken to generate a realistic model that serves as a starting point for further research while idealizing some components for simplicity.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I want to thank my thesis advisor, Professor Jun Zhang, for spending much time with me going over all aspects of the thesis, from topic selection to conclusion of the project a year later. His class *Introduction to Machine Learning* in Fall 2017 was where I was first introduced to him, and I understood soon afterwards that his vast wisdom and knowledge of algorithms and mathematical concepts would be invaluable to my work. My thorough understanding of dynamic programming and machine learning is a result of his clear and detailed explanations.

Special gratitude also goes to Professor Adel Nasiri for not only proposing the subject of this thesis, but also for providing me the opportunity to work on his research projects. The monetary benefits of performing in this role from the onset of my graduate education were supplemented by the experience I gained in several areas related to computer science. The Graduate Research Assistantship that I earned for Fall 2018 by virtue of my work on these projects was a particularly handsome reward. Thanks also to Nasim Yahyasoltani for providing guidance on the research projects along with additional advice for my thesis.

Chapter 1 – Introduction

Solar power is a primary source of energy for the future. Generating the energy, of course, costs nothing at all, so the financial concerns are limited to harvesting and storing the energy. Although these costs are not necessarily insignificant – the costs of labor and equipment can add up to a high amount that might make the whole project impractical – the benefits of utilizing solar energy are considered highly lucrative. However, as with any source of energy, solar power has its downsides.

Solar energy is unstable, which means that the Sun to often unable to accommodate demand for electricity at all times of the day. Cloud cover and precipitation may significantly impact the quantity of solar power that can be harvested, as does geographical latitude of the location utilizing energy from the Sun. The need to supply energy at such possible downtimes for solar power suggests that other, more reliable sources of energy must be available for deployment as well. On the other hand, if the Sun blazes on to the PV cells at a time when demand of energy is low, there would be great wastage of precious energy. An energy-storage mechanism thus merits strong consideration. The system that results from these reflections is a hybrid one that contains a renewable source of energy, an energy-storage mechanism, and one or more traditional sources of energy.

This thesis describes a hybrid system that consists of several forms of energy, namely grid, generator, direct solar power (from PV cells), and stored solar power (from battery); from which electrical demands, of which some are fixed and unchangeable, may be met. The cost and availability of each form of energy is different and may be dependent on the time of day, week, and year. Using solar energy as the most preferable option and utilizing several types of data

along with the technique of dynamic programming, a load schedule that minimizes cost for the consumer is developed. The process and technique described may be utilized at a variety of sites, like family home, school, and commercial property. They may also be applied at any location around the world at any time of the year. As an example, this thesis uses the application of single-family residence in Milwaukee, WI, USA and focuses on the warm month of July.

Contributions of This Thesis

This thesis applies dynamic programming for the first time as the method to solve a problem that has been a popular subject in industry and academia. The project involves using in a practical situation an algorithm often associated with mathematical contexts. Existing literature, which is briefly explored in Chapter 2, contains plethora of works on optimization of hybrid renewable energy systems but few, if any, that apply dynamic programming or related techniques to such a topic. Although some parts of this thesis are idealized and simplified, the project proffers a strong basis for more intricate additions and modifications.

Chapter 2 – Related Works

An abundance of literature exists on the topic of optimization of renewable energy sources. A few of them will be concisely explained here to provide context to the current thesis.

[1] discusses a computer program that was developed to calculate the most efficient energy source to meet a required load in rural South Africa at any time of the day. The available energy sources were solar and wind, so the program logically considered, among other factors, time of day and average wind speed at the location in question. The description of this program suggests that it required manual execution and entering of parameters. Because the project was completed before the 21st Century, it did not incorporate modern tools such as new algorithms and machine learning.

[2] explains a method using linear programming, a specific type of mathematical optimization that minimizes a linear function under linear constraints, to develop an optimized schedule of charging and discharging a battery in a system that can cover a load with solar power, battery power, or a distribution feeder. The paper also contains cost analyses that highlight, in turn, the economic benefits of adopting the best schedule when the solar-energy infrastructure already exists; the effectiveness of the battery in reducing demand from the grid even when the costs of installing the solar equipment are prohibitive; and the situations where solar-energy utilization is practically advantageous. While this work was innovative in using a linear function for optimization in this context, the linear model might not have captured all the details contained in the energy system.

[3] and [4] both describe projects that suggest installation of a hybrid renewable energy system for areas in need of progress. [3] concentrates on proposing a system for a rural school in

Morocco. The suggested system is equipped with a hybrid renewable energy system consisting of solar energy, wind energy, a battery, and a diesel generator. Using HOMER Pro (Hybrid Optimization Model for Electrical Renewable), which is a simulation software developed by the U.S. National Renewable Energy Laboratory (NREL), the authors analyze three combinations of energy sources to determine the best strategy for covering the electrical load of the school while maintaining cost efficiency and considering the unpredictability of wind and solar energy. [4] explores a possible system for an underdeveloped island in Bangladesh. This system does not include wind energy; however, it does add a generator fueled by biogas, which is an economical source of energy for this specific location because of the large number of cattle residing there. As in [3], this project considers solar energy, a battery, and a diesel generator. Also, as in [3], the authors of [4] use the HOMER simulation program to develop an optimized system architecture. These works both put emphasis on cost optimization, but using an existing program developed by a third party instead of creating a new approach themselves.

[5] uses a probabilistic approach to study the performance of a hybrid solar-wind energy system that is backed up by a battery and generator. Connection to an electric grid is available as an additional option to the system. This system is installed at Vasavi College of Engineering in Hyderabad, India. The study is conducted using the average wind speed at an attractive location for harvesting wind energy and ten years' worth of prior data on solar radiation to predict the solar power generation. The prediction is made with the quadratic equation $P = Ax^2 + Bx + C$, where x is the solar radiation; P is the power generation; and A, B, and C are coefficients derived from measured data. The average wind speed and predicted solar power generation act as the probabilistic factors for this project. Information on annual energy production by this hybrid system is collected over two years. As usual, a cost-benefit analysis is presented and considers

lifetimes of the equipment. It concludes that if a reduction in cost of renewable-energy equipment is assumed, the hybrid system is indeed an economically superior choice to a more conventional option. It also states that solar and wind energies act as complements to one another. This project also shines a spotlight on optimization, but on a hybrid system that has already been installed. Flexibility in modifying the components of the system for study purposes is thus limited.

Chapter 3 – Problem Formulation

Chapter 3.1 – Components of the System

Energy Sources: As explained earlier, the system contains four different sources of energy – solar energy, battery, grid, and generator. The first of these is the one most preferred to meet the demand. If, after completely doing so, excess solar energy is available, the battery (which initially has no charge) is charged so that it can be used as the backup to solar energy if needed in the future. In other words, if solar energy is unable to fully meet the demand, the battery is discharged. Some unavoidable loss of energy occurs when charging the battery with solar energy. The percentage of solar energy that charges the battery compared to the quantity of solar energy that is provided to the battery is known as round-trip efficiency. If both solar energy and battery combined are unable to fully meet the demand, grid or generator is chosen depending on the cost of each at the hour the energy needs to be supplied.

An example of realistic solar-energy output is 1 kWh per day with a PV panel rated 250 W. Of course, the actual output depends on the rating of the panel, the size of the panel, the geographic location where the panel is installed, precipitation, and other factors [6]. The round-trip efficiency of a battery is determined by the age and constitution of the battery along with other details, but ranges from 80% to 95% for a typical battery [7].

Costs: Energy can be bought from the grid or generator for a certain cost if both solar energy and battery cannot completely meet the load demand for the hour. Depending on the hour of day, the grid or generator is cheaper than the other and the more economical option is accordingly selected.

Loads: There are two types of loads – fixed and variable. Fixed loads cannot be changed in timing or quantity. On the other hand, variable loads can be changed in timing or hourly quantity but must remain constant for the day. A variation is considered where the variable load for the day is within a range, but the exact value is randomly determined.

An example of a fixed load is lighting. The lights must always be kept on at certain times, sometimes even throughout the day. If ten bulbs rated 100 W are used for ten hours on one day, the energy consumption by the lighting is 10,000 Wh or 10 kWh [8]. An example of a variable load is a washing machine. This can be operated at a time of the user's choosing. A typical washing machine consumes 255 watts per hour of use. This translates to 255 Wh or 0.255 kWh [9].

Figure 1 shows a diagram of the components of the system.

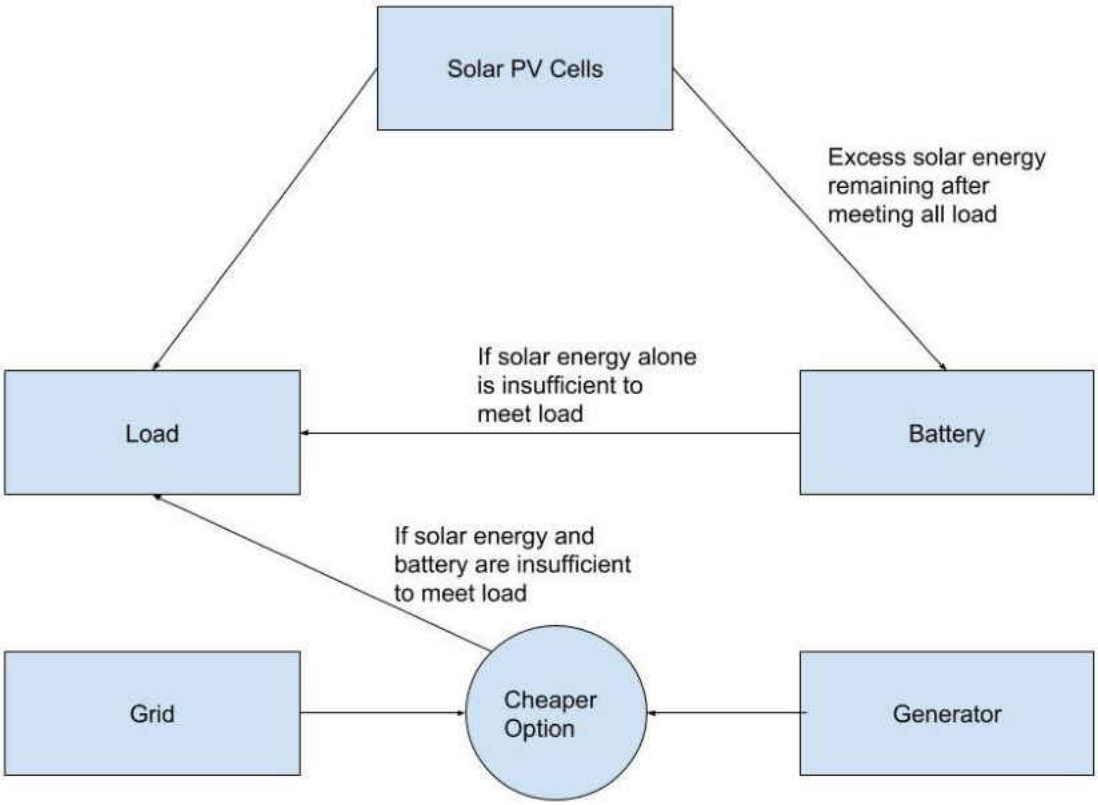

Figure 1: Components of the System

Chapter 3.2 – Mathematical Description

Chapter 3.2.1 – The Model

The mathematical model for this problem is a state-space representation that contains inputs, a control, a state, and a cost.

Inputs: There are two inputs to the system – solar energy available and fixed-load schedule. Neither is controllable but must be considered while developing the optimal-control strategy.

Control: The control is the variable-load schedule. This is completely controllable and needs to be determined such that the cost incurred after adopting the total-load schedule – the schedule for the fixed and variable loads combined – is the lowest possible.

State: The state of the system is given by the amount of battery charge available at the beginning of any hour. This depends on both an uncontrollable factor – the solar energy available at the previous hour – and a controllable factor – the demand at the previous hour.

Cost: The cost is the price of energy bought from the grid or generator. The cost thus depends on the total load as well as the available solar energy and energy in battery.

Chapter 3.2.2 – Notations and Definitions

t: The time, indicated in discrete, whole-number values that represent the hour of the time span being studied

x(t): The state of the system, which is the energy in the battery, at time t

s(t): One input to the system, which is the solar energy available, at time t

l(t): Another input to the system, which is the fixed-load schedule, at time t

u(t): The control, which is the variable-load schedule, at time t

a(t, u(t)): The amount of energy to be purchased from the generator or grid at time t as a result of covering u(t) amount of variable load

g_1(t): The generator cost at time t

g_2(t): The grid cost at time t

c(t, u(t)): The total cost incurred by the system at time t for covering u(t) amount of variable load

Chapter 3.2.3 – Equations

The problem that has been described thus far may be viewed as one of optimal control. Here, the optimal-control strategy is to obtain the cheapest-possible load schedule for a given day. The following equations mathematically model this problem of optimal control.

State Equation: $x(t+1) = \max(x(t) + s(t) - l(t) - u(t), 0)$

The state equation models the energy available in the battery (x(t + 1)) for the next hour (t + 1). Depending on which is higher, it is either zero or the difference between the energy available and the total load. The energy available is the sum of solar energy and the energy in the battery (x(t) + s(t)). The total load is the sum of fixed and variable loads (l(t) + u(t)). Zero is the lowest output of this equation because energy available in the battery can never be negative.

Cost Function: $c(t, u(t)) = \min\big(g_1(t) * |a\big(t, u(t)\big)|, g_2(t) * |a\big(t, u(t)\big)|\big)$,

where $a(t, u(t)) = \min(x(t) + s(t) - l(t) - u(t), 0)$

The cost function calculates the cost of a certain quantity of variable load (u(t)) at a certain time (t). The min operator chooses the cheaper option between grid ($g_2(t)$) and generator ($g_1(t)$). a(t, u(t)) is the amount of energy to be bought from either of those two sources. If solar energy alone or solar energy combined with battery were able to cover all the load, the amount of energy to be bought will be zero. Otherwise, it is the difference between the load that was covered by solar and battery energies (x(t) + s(t)) and the total load to be covered (l(t) + u(t)). If a(t, u(t)) is not zero, it is negative, indicating that some load remains to be covered. Because a(t, u(t)) is non-positive and both $g_1(t)$ and $g_2(t)$ are non-negative, the absolute value of a(t, u(t)) is inserted into

the cost function before being multiplied by $g_1(t)$ and $g_2(t)$ (separately) to obtain two non-negative values, out of which the minimum is chosen.

The min and max operators found in the state equation and cost function make this problem highly non-linear. Straightforward analytic techniques, such as linear programming, would be difficult, if not impossible, to execute on this model. Therefore, a more complex approach is necessary.

Chapter 4 – Approach to Solution

Chapter 4.1 – Dynamic Programming

Chapter 4.1.1 – Theory

Dynamic programming is an algorithm commonly applied to optimal-control problems. It is similar to the divide-and-conquer method in that both approaches solve a problem by first solving subproblems recursively and then joining the results. However, those problems that are conducive to dynamic programming contain overlapping subproblems – the subproblems themselves contain subsubproblems. This means that divide-and-conquer, while still a theoretically valid algorithm for this problem, may become highly inefficient because the same subsubproblems may be solved repeatedly. Dynamic programming, on the other hand, saves solutions to these subproblems in a table so that they need not be computed more than once. Dynamic programming can be also contrasted to greedy algorithms. While the former first optimally solves subproblems before constructing an optimal solution to the complete problem, the latter makes the apparent best choice before solving subproblems. [10]

A dynamic-programming algorithm consists of four key steps:

1) Conceive of optimal-solution structure.

2) Define optimal value to each subproblem recursively.

3) Compute optimal value; often starting from smallest subproblem.

4) Create optimal solution from solutions to subproblems generated previously.

Dynamic programming is essentially defined by the first three steps. [10]

Saving the solution to each subproblem after it is calculated just once is crucial to the viability of dynamic programming as an algorithm. The saved solutions to the subproblems may simply be referred to whenever the subproblems need to be solved more than once. While saving the solutions consume memory, the process saves much time. This situation is an example of time-memory tradeoff. The increase in runtime efficiency is potentially great enough to reduce an exponential-time solution to a polynomial-time one. For dynamic programming to run in polynomial time, the number of distinct subproblems must be polynomial in input size and each subproblem can be solved in polynomial time. [10]

Dynamic programming may be implemented in two equivalent ways. Top-down with memorization starts with recursion on the whole problem, but then saves the results in a table to each subproblem beginning with the smallest one. Whenever a certain subproblem is encountered, the table is first checked if the subproblem has been solved previously. If so, that solution is returned and repeated computation of the solution to a subproblem is avoided. Otherwise, the solution to the subproblem is calculated normally. The bottom-up method involves sorting subproblems by size and solving them in ascending order. When solving a subproblem, all smaller subproblems are assumed to have been already solved and their solutions saved. Each subproblem is thus solved only once. Both approaches usually lead to the same asymptotic running times. [10]

Two features are characteristic of dynamic-programming problems: optimal substructure and overlapping problems. [10]

Optimal Substructure

If a problem displays optimal substructure, the full optimal solution contains optimal solutions to smaller subproblems. Finding such a substructure often involves four steps.

1) The initial, whole problem needs to be solved by making a choice that leaves subproblems to be solved.

2) For a given problem, the choice that results in the optimal solution is known.

3) This choice leads to a specific set of subproblems with certain qualities.

4) Solutions to the subproblems of the complete optimal solution are themselves optimal. [10]

A problem that exhibits optimal substructure adheres to the Principle of Optimality, which states that “An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.” [11]

Optimal substructure varies across the problem domain in two different ways.

1) The number of subproblems used by the optimal solution.

2) The number of choices available to select the subproblem for usage in the optimal solution. [10]

Optimal substructure is often utilized in dynamic programming by first finding optimal solutions to subproblems and then finding the complete optimal solution. This involves choosing the subproblems to use the optimal solution. The total cost of the solution is usually the sum of the subproblem costs and the cost of the choice itself. [10]

A Bellman equation uses the nature of optimal substructure that is inherent to a problem able to be solved by dynamic programming to describe the necessary condition for optimality

[12]. The value of the equation is the combination of the cost resulting from a certain choice and the costs obtained from the subproblems derived from making that choice. Some simple Bellman equations may be solved analytically; most, however, need to be solved numerically, especially if the optimal-control strategy contains nonlinear elements. [13]

Overlapping Subproblems

Overlapping subproblems are present when a recursive algorithm repeatedly arrives at the same subproblems. This quality distinguishes a problem that can be solved by dynamic programming from one that can be handled by divide-and-conquer. Overlapping subproblems generally means that the space of subproblems is rather small. In other words, the number of distinct subproblems is low compared to the input size. [10]

Chapter 4.1.2 – Application to Thesis Problem

Dynamic programming is quite a suitable approach to solving the thesis problem. It can be modeled by the following Bellman equation:

$$v(t,k) = \min_{u}\big(c(t,u(t)) + v(t-1,k-u(t))\big),$$

where t is a certain hour of the day, k is the quantity of variable load to cover during the time interval [0, t], v is the minimum cost of k, u is the control defined in Chapter 3.2.3, and c is the cost function defined in the same chapter. This equation fits the standard model for Bellman equations, except that the cost function implicitly contains the state term x(t) – energy in the battery at hour t, which is a term in a(t, u(t)) as defined in Chapter 3.2.3 – that is dependent on the solution to the subproblem v(t – 1, any). Typically, the cost function is independent from solutions to subproblems.

The existence of Bellman equation implies that this problem exhibits optimal substructure. This can also be proven according to the four steps described in Chapter 4.1.1 (note: example values given in the remainder of Chapter 4.1.2 will assume 12 kWh of variable load per day, which is the primary case).

1) The full problem has n be 24. Make a choice for the 24^{th} hour, like 1 kWh. u(24) is thus 1. The total variable load to cover during the day is 12 kWh. This corresponds to v(24, 12). Then, the largest subproblem to be solved is v(23, 11).

2) Assume u(24) is known such that c(24, u(24)) is optimal. Let u(24) be 1 for this purpose.

3) The optimal u(24) results in the subproblem v(23, 11). In fact, this is a superset of even smaller subproblems.

4) v(23, 11) and the smaller subproblems within it are themselves optimal. To prove so, assume that v(24, 12) is optimal but v(23, 11) is not. Then, a better solution for v(23, 11) must exist, making v(24, 12) better as well. But, this contradicts the assumption that v(24, 12) was already optimal.

The existence of the Bellman equation also shows the presence of overlapping subproblems. This can be evidenced in the fact that v(24, 12) is a superset of smaller subproblems like v(23, 11), which is itself a superset of even smaller subproblems.

Because the cost function contains the nonlinear min and max operators, and the cost function is part of the Bellman equation describing the optimal-control strategy, the Bellman equation cannot be solved analytically. However, it is friendly to a numerical approach, specifically one with top-down memorization because x(t), energy in the battery, will be known for hour t only after the problem has been solved for all previous hours. The Bellman equation is

implemented as a recursive algorithm, so the first function to be called is v(24, 12), but the table is filled in a bottom-up fashion, starting from v(1, 12).

Chapter 4.2 – Implementation

Chapter 4.2.1 – Data for Each Element of Mathematical Equations

The following list compiles the sources and/or basis for the data supplied for each part of the state equation and cost function.

t: The time is supplied in discrete values for every hour of the day

x(t): The battery contains no energy in the beginning and is supplied with excess energy from the Sun if such excess is available. The battery is assumed to have a round-trip efficiency of 50%, which is on the lower end of the possible values [7].

s(t): The hourly values for solar energy are supplied by the System Advisor Model (SAM) program, explained in Chapter 4.2.2.

l(t): Fixed loads are specified to be 0.5 kWh each at the following hours of the day: 0, 4, 8, 12, 16, and 20.

u(t): The variable load is the output generated for every hour by the dynamic-programming algorithm implemented with MATLAB, explained in Chapter 4.2.3. The primary case considered is where the total variable load for the day stays constant at 12 kWh. For the variation where the total variable load for the day is determined randomly but remains within a range, the minimum and maximum quantities are specified as 9 kWh to 15 kWh, respectively.

a(t, u(t)): The amount of energy to be purchased from the grid or generator is derived from x(t), s(t), l(t), and u(t).

$g_1(t)$: The generator cost is given as \$0.15/kWh at all times of the week [14].

$g_2(t)$: The grid costs are given in the following table. They are stated such that the grid is competitive in price with the generator. The units are cents per kilowatt-hour.

Table 1: Grid Costs

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Sunday	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
Monday	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	17	17	17	17	13	13	13
Tuesday	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	17	17	17	17	13	13	13
Wednesday	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	17	17	17	17	13	13	13
Thursday	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	17	17	17	17	13	13	13
Friday	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	17	17	17	17	13	13	13
Saturday	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10

$c(t, u(t))$: Using MATLAB, the cost is calculated for every hour from $g_1(t)$, $g_2(t)$, and $a(t)$.

Chapter 4.2.2 – System Advisor Model

The System Advisor Model (SAM) is "techno-economic computer model designed to facilitate decision making for people involved in the renewable energy industry" [15]. It is a sophisticated program that allows a variety of professionals, including researchers, engineers, project managers, and financial analysts to develop realistic models of renewable energy systems [15]. These models consider several factors like basic components of the system (solar only, wind only, solar-and-wind hybrid, etc.), location of the system, and type of battery to be installed.

For the current project, SAM is used to obtain a series of values of solar energy that would be available to meet the load requirements. To do this, a complete system needs to be modeled. The model preferred for this case is the *Photovoltaic (detailed)* option (this one allows more settings to be modified by the user than the simpler *Photovoltaic (PVWatts)* choice).

Within the *Photovoltaic (detailed)* option, the *Residential (distributed)* selection is made because the application of current project resembles that of a single-family home.

The chosen model contains many more factors than the ones being considered in the current application; so, those components that are not within its scope or do not affect the goal of acquiring the solar-energy values, like snowfall and lifetime of battery, are kept at default. The important parameters to set are the *Location and Resource* and the *Electric Load*.

The weather data are generated by a file that contains real recordings from past years or simulations for a typical meteorological year (TMY) at a certain location. To select a file, under the parameter *Location and Resource* and under the section *NREL National Solar Radiation Database (NSRDB)*, the button for *TMY or Single-year for Americas and Asia* is clicked. The location is then entered and searched. Milwaukee, WI is selected for this case as this project is being conducted in that city. The first option yielded by the search, *milwaukee_WI_psm_satellite_60_tmy*, is chosen. As the name of the file suggests, it contains the TMY simulations that will be a general depiction of the weather in Milwaukee and thus will be better suited for this project, which is intended to be valid for any year. This file will now appear in the subsection *Files in* Library within the section *Solar Resource Library*. Clicking on the file will allow it and some of its details to populate a few of the fields in *Solar Resource Library*. The process for selection of weather data is now complete.

Now, the *Electric Load* section must be completed. This section accounts for the total load consumption by the building so that the model can calculate the solar energy available to meet this demand. Entering realistic values for the current situation suffices to produce satisfactory results. For the model, under the *Building Characteristics* subsection, the *Floor area* is specified as 1,000 sq. ft., the *Year built* is given as 1975, the *Number of stories* is declared as

1, and the *Number of occupants* is also declared as 1. All the options are checked under the *Electrical Appliances* subsection. Under the *Temperature Settings* subsection, the *Heating setpoint* and *Heating setback point* are set to 75° F and the *Cooling point and Cooling setup point* are set to 82° F. Under the *Monthly Load Data,* the values submitted are 800 kWh for Jan. to Mar. and Oct. to Dec., 700 kWh for Apr. and Sep., and 600 kWh for May to Aug. The premise behind these relative energy values is that the heating system will be heavily utilized during the colder months.

After these steps are finished, the *Simulate* button is clicked. The *Data Tables* tab is selected, the *Hourly Data* section is expanded, and the option for *Electricity from system to load (year 1 hourly) (kWh)* is chosen as the only column to display. Then, the data is saved by clicking the *Save as CSV* button and choosing a desirable directory. If the time range of the data to be examined is smaller than the whole dataset, which is true in this case as the desired time range is the month of July, the dataset is trimmed using a program like Excel to contain only the desired range. The solar-energy values have now been obtained and can be used to solve the problem with another application.

Figure 2 shows the screen on SAM where for every hour, the quantity of solar energy applied to the load is shown. Because the SAM model has all solar energy directed to the load with no battery enabled, these values serve, in effect, as those of solar energy available to the whole system at each hour of the year for this thesis project.

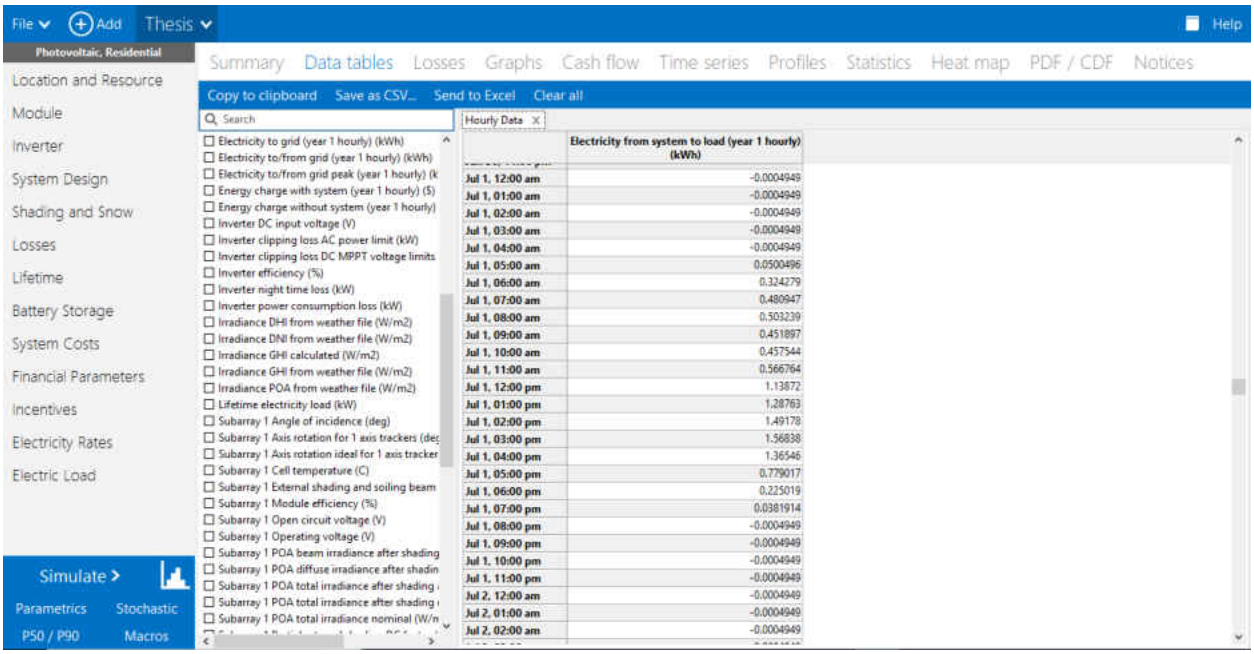

Figure 2: Example Screenshot of SAM

Chapter 4.2.3 – MATLAB

The problem is solved by means of a computer program written in MATLAB, which is a software and computer language created for scientific and mathematical computation. This program applies the top-down-memoization version of dynamic programming that was explored in Chapter 4.1.1 by utilizing typical computer-language data types and concepts like arrays, for loops, if-then statements, and functions.

To calculate the total minimum cost for a given day, a function known as `recursive_portion` is created. This function essentially implements the Bellman equation

stated in Chapter 4.1.2. Parameters to this function include n and k, as defined in the Bellman equation, as well as the energy in the battery available at the beginning of the day (same as quantity available after previous day; 0 if first day). Initially, n is 24 and k is 12 (for the primary case – this case will henceforth be assumed for the remainder of Chapter 4.2.3). Then, a choice is made for u(t = n). This choice ranges in ascending order from 0 to 12 in specified increments (also known as step sizes), so the first u(t = n) will always be 0. The function proceeds to call itself in a recursive way with n – 1 (e.g. if n = 24, 24 – 1 = 23) and k – u(t = n) (e.g. if n = 24, 12 – 0 = 12) as the parameters. This process will continue until n = 1. At this point, the recursion stops momentarily and the cheapest cost possible for the remaining load of u(t = 1) (e.g. 12) is calculated. This value, along with the energy available in the battery for u(t = 2) (e.g. 0) after possibly having used the battery for u(t = 1), is saved in a table and subsequently returned to the instance of `recursive_portion` that has n = 2 and a corresponding variable-load choice of u(t = 2) (e.g. 0). This instance now calculates the cheapest cost possible for the current choice of u(t = 2). Then, this instance makes the next choice available for u(t = 2) (e.g. 1 if increment = 1) and again calls itself with parameters n – 1 and k – u(t = 2) (e.g. 11), which triggers a repetition of the procedure described for t = 1 but for the quantity of u(t = 1) resulting from the new choice of u(t = 2). In this manner, the cheapest cost possible for each valid combination of n (1 – 24 in integer values) and k (0 – 12 in specified increments) will eventually be tabulated. Whenever an instance of `recursive_portion` is called with a certain combination of n and k for which the minimum cost has already been calculated, that value will simply be retrieved from the table and returned to the previous instance of `recursive_portion`, thereby saving significant computation time.

The aforementioned process may be repeated for multiple days with ease. The cheapest cost may be calculated for several days by simply adding together the cheapest cost over each day in that time span. One important point is that this sum is much distinguished from the minimum cost computed over several days, which would result from applying dynamic programming over that combined time frame.

Chapter 5 – Results and Analysis

Chapter 5.1 – Results

Primary Case – Constant Quantity of Variable Load for Day

As mentioned before, the amount of variable load that can be assigned to any hour ranges from 0 to the total remaining variable load to be assigned (12 kWh initially). Ideally, this is a true spectrum, out of which any value can be considered valid. However, because this solution is numerical in nature, the range consists of increments of a size that is determined by the user. As the increments become smaller, the program takes a much greater amount of time to complete, thus rendering extremely small increments impractical, if not impossible. So, to determine the best practical increment and to observe the correlation between increment size and minimum cost, the increments considered here are 1 kWh, 0.5 kWh, and 0.25 kWh. Table 2 tabulates the total minimum cost and the running time of the program using each of these increments for the 31 days in July. Figure 3 shows the information on Table 2 on a bar graph for better visual comparison. On the bar graph, the total minimum cost is on the x-axis and the running time (shown on the graph as the average of the two extreme values listed on the table) is on the y-axis to glean light on the tradeoff between decrease in cost and increase in running time.

Table 2: Total Minimum Cost and Running Time for Each Increment (Primary Case)

Increment	Total Minimum Cost	Running Time
1 kWh	$22.53	0.6 – 0.9 seconds
0.5 kWh	$21.57	1.8 – 1.95 seconds
0.25 kWh	$21.31	6.8 – 8.4 seconds

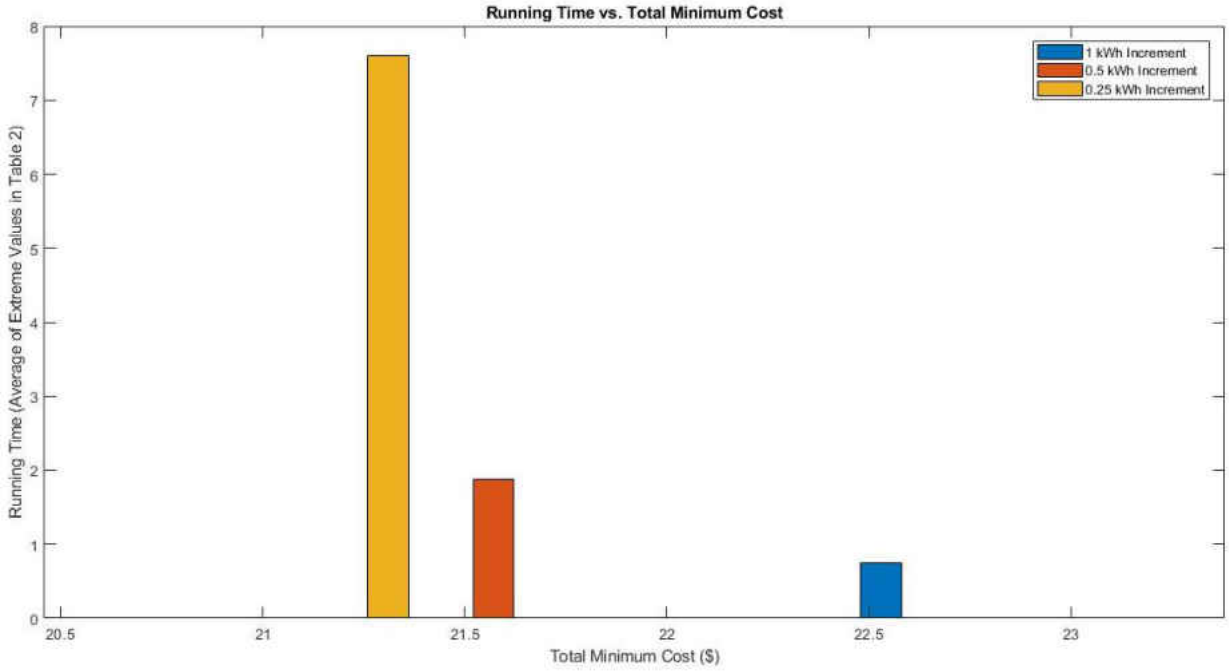

Figure 3: Running Time vs. Total Minimum Cost

Table 3 displays the hourly variable load schedule for the same duration in increments of 1 kWh. The unit is kWh. Each row represents a day in July. Each column represents an hour of the day.

Table 3: Variable Load Schedule in Increments of 1 kWh

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1st	0	0	0	0	0	0	0	0	0	1	1	1	1	2	2	2	1	1	0	0	0	0	0	0
2nd	0	0	0	0	0	0	1	1	0	1	1	1	0	1	1	1	1	1	0	0	0	1	1	0
3rd	0	0	0	0	0	1	1	1	0	1	1	1	0	1	1	1	1	1	1	0	0	0	0	0
4th	0	0	0	0	0	0	0	1	0	1	1	1	0	1	2	2	1	1	1	0	0	0	0	0
5th	0	0	0	0	0	0	0	0	0	1	1	1	1	2	2	2	1	1	0	0	0	0	0	0
6th	0	0	0	0	0	0	0	0	0	1	1	1	1	2	2	2	1	1	0	0	0	0	0	0
7th	1	0	0	0	0	0	0	1	0	1	1	1	0	1	1	1	1	1	1	1	0	0	0	0
8th	0	0	0	0	0	0	1	1	0	1	1	1	0	1	1	1	1	1	1	1	0	0	0	0
9th	0	0	0	0	0	0	0	1	0	1	1	1	0	1	2	2	1	1	0	0	0	1	0	0
10th	0	0	0	0	0	0	0	0	0	0	1	2	1	2	2	2	1	1	0	0	0	0	0	0
11th	0	0	0	0	0	0	0	1	0	1	1	1	0	1	2	2	1	1	0	0	0	1	0	0
12th	0	0	0	0	0	0	0	1	0	1	1	1	0	2	2	2	1	1	0	0	0	0	0	0
13th	0	0	0	0	0	0	1	1	0	1	1	1	0	1	1	2	1	1	0	0	0	1	0	0
14th	0	0	0	0	0	0	0	1	0	1	1	1	0	1	2	2	1	1	1	0	0	0	0	0
15th	0	0	0	0	0	0	0	0	0	0	1	1	1	2	2	2	1	1	1	0	0	0	0	0
16th	0	0	0	0	0	0	0	1	0	1	1	2	1	2	2	1	0	1	0	0	0	0	0	0
17th	0	0	0	0	0	0	0	0	0	0	1	2	1	2	2	2	1	1	0	0	0	0	0	0
18th	0	0	0	0	0	0	0	1	0	1	1	1	0	1	2	2	1	1	1	0	0	0	0	0
19th	0	0	0	0	0	0	0	1	0	0	1	1	0	1	1	1	0	1	0	0	0	2	1	2
20th	0	0	0	0	0	0	0	1	0	1	1	1	1	2	2	1	1	1	0	0	0	0	0	0
21st	0	0	0	0	0	0	0	1	0	1	1	1	0	1	2	2	1	1	1	0	0	0	0	0
22nd	0	0	0	0	0	0	0	1	0	1	1	1	0	2	2	2	1	1	0	0	0	0	0	0
23rd	0	0	0	0	0	0	0	1	0	1	1	1	1	1	2	2	1	1	0	0	0	0	0	0
24th	0	0	0	0	0	0	0	0	0	1	1	1	1	2	2	2	1	1	0	0	0	0	0	0
25th	0	0	0	0	0	0	1	1	0	1	1	1	0	1	1	1	0	1	0	0	0	3	0	0
26th	0	0	0	0	0	0	0	0	0	1	1	1	1	2	2	2	1	1	0	0	0	0	0	0
27th	0	0	0	0	0	0	0	0	0	1	1	1	1	2	2	2	1	1	0	0	0	0	0	0
28th	0	0	0	0	0	0	0	0	0	0	1	2	1	2	2	2	1	1	0	0	0	0	0	0
29th	0	0	0	0	0	0	0	1	0	1	1	1	1	2	2	1	1	1	0	0	0	0	0	0
30th	0	0	0	0	0	0	0	0	0	0	1	2	1	2	2	2	1	1	0	0	0	0	0	0
31st	0	0	0	0	0	0	0	1	0	1	1	2	1	2	2	1	0	1	0	0	0	0	0	0

Table 4a displays the load schedule (in kWh) for the first twelve hours of the day in increments of 0.5 kWh. Table 4b displays the same data for the second twelve hours of the day. As before, the unit is kWh, each row represents a day in July, and each column represents an hour of the day.

Table 4a: Variable Load Schedule for First Twelve Hours in Increments of 0.5 kWh

	0	1	2	3	4	5	6	7	8	9	10	11
1st	0	0	0	0	0	0	0.5	0.5	0	0.5	0.5	1
2nd	2	0	0	0	0	0.5	0.5	1	0.5	0.5	0.5	1
3rd	4	0.5	0	0	0	0.5	0.5	0.5	0	0.5	0.5	0.5
4th	0	0	0	0	0	0.5	0.5	0.5	0	0.5	1	0.5
5th	0	0	0	0	0	0.5	0.5	0.5	0	0.5	0.5	1
6th	0	0	0	0	0	0	0.5	0.5	0	0.5	0.5	0.5
7th	0	0	0	0	0	0	0	1.5	0	1	2	0.5
8th	2.5	0	0	0	0	0.5	0.5	0.5	0.5	0.5	0.5	0.5
9th	0	0	0	0	0	0.5	0.5	0.5	0.5	0.5	0.5	0.5
10th	0	0	0	0	0	0	0.5	0.5	0	0.5	0.5	1.5
11th	2.5	0	0	0	0	0	0.5	0.5	0	0.5	0.5	0.5
12th	1	0	0	0	0	0	2.5	0.5	0	0.5	0.5	0.5
13th	0.5	0	0	0	0	2.5	0.5	0.5	0	0.5	0.5	0.5
14th	0	0	0	0	0	1.5	0.5	0.5	0	0.5	0.5	0.5
15th	0	0	0	0	0	0	0	0.5	0	0.5	0.5	1.5
16th	0	0	0	0	0	0.5	0.5	1	0.5	0.5	1	1.5
17th	0	0	0	0	0	0	0.5	0.5	0	0.5	0.5	1.5
18th	4	0	0	0	0	0	0.5	0.5	0	0.5	0.5	0.5
19th	4	0.5	0.5	0.5	0	0.5	1	1	0	0.5	0.5	0.5
20th	0.5	0	0	0	0	0.5	0.5	0.5	0	0.5	0.5	0.5
21st	0	0	0	0	0	1	0.5	1	1.5	0.5	0.5	0.5
22nd	0	0	0	0	0	0.5	0.5	0.5	0.5	0.5	0.5	0.5
23rd	0	0	0	0	0	0.5	0.5	0.5	0.5	1	0.5	0.5
24th	0	0	0	0	0	0	0.5	0.5	0	0.5	0.5	1
25th	1	0	0	0	0	2	1	0.5	0.5	0.5	0.5	0.5
26th	0	0	0	0	0	0	0.5	0.5	0	0.5	0.5	1
27th	0	0	0	0	0	0	0.5	0.5	0	0.5	0.5	1
28th	0	0	0	0	0	0	0	0.5	0	0.5	1	1.5
29th	0	0	0	0	0	0	0.5	0.5	0	0.5	1	1.5
30th	0	0	0	0	0	0	0	0.5	0	0.5	0.5	1.5
31st	0	0	0	0	0	0	0.5	0.5	0	0.5	1	1.5

Table 4b: Variable Load Schedule for Second Twelve Hours in Increments of 0.5 kWh

	12	13	14	15	16	17	18	19	20	21	22	23
1st	1	1.5	1.5	2	1	1	0.5	0.5	0	0	0	0
2nd	0	0.5	1	1.5	0.5	0.5	0.5	0	0	1	0	0
3rd	0	0.5	0.5	1	1	1	0.5	0	0	0	0	0
4th	0	1	1.5	1.5	1	1	0.5	0	0	1.5	0.5	0
5th	1	1.5	1.5	2	1	1	0.5	0	0	0	0	0
6th	1	1.5	2	1.5	0.5	0.5	0.5	0	0	1	0.5	0.5
7th	0.5	1	0.5	1	1	1	1	0.5	0.5	0	0	0
8th	0	0.5	1	1	1	1	1	0.5	0	0	0	0
9th	0.5	1.5	1.5	1.5	1	1	0.5	0	0	0.5	0.5	0
10th	1	1.5	2	1.5	1	0.5	0.5	0	0	0.5	0	0
11th	0	1	1.5	1.5	1	1	0.5	0	0	0.5	0	0
12th	0	1.5	1.5	1.5	1	0.5	0.5	0	0	0	0	0
13th	0	1	1.5	2	0.5	1	0.5	0	0	0	0	0
14th	0.5	1.5	1.5	1.5	1	1	0.5	0.5	0	0	0	0
15th	1	1.5	2	2	1	1	0.5	0	0	0	0	0
16th	1.5	1.5	1.5	1	0	0.5	0.5	0	0	0	0	0
17th	1	2	2	2	0.5	0.5	0.5	0	0	0	0	0
18th	0	0.5	1.5	1.5	0.5	1	0.5	0	0	0	0	0
19th	0	0.5	0.5	0.5	0	0.5	0.5	0	0	0	0	0
20th	0.5	1.5	1.5	1	1	1	0.5	0	0	0.5	0.5	0.5
21st	0	1	1.5	1.5	1	1	0.5	0	0	0	0	0
22nd	0.5	1.5	1.5	1.5	1	1.5	1	0	0	0	0	0
23rd	0.5	1.5	1.5	2	1	1	0.5	0	0	0	0	0
24th	1	2	2	1.5	1	0.5	0.5	0	0	0.5	0	0
25th	0	1.5	1	1	1	0.5	0.5	0	0	0	0	0
26th	1	1.5	2	2	1	1	0.5	0	0	0	0	0
27th	1	2	2	2	1	0.5	0.5	0	0	0	0	0
28th	1	1.5	1.5	2	1	1	0.5	0	0	0	0	0
29th	1	1.5	2	1.5	1	0.5	0.5	0	0	0	0	0
30th	1	2	2	2	1	0.5	0.5	0	0	0	0	0
31st	1.5	2	1.5	0.5	0	0.5	0.5	0	0	1.5	0	0

Table 5a displays the load schedule (in kWh) for the first twelve hours of the day in increments of 0.25 kWh. Table 5b displays the same data for the second twelve hours of the day. Once again, the unit is kWh, each row represents a day in July, and each column represents an hour of the day.

Table 5a: Variable Load Schedule for First Twelve Hours in Increments of 0.25 kWh

	0	1	2	3	4	5	6	7	8	9	10	11
1st	0	0	0	0	0	0.25	0.5	0.5	0.25	0.75	0.75	0.75
2nd	1.75	0.25	0.25	0.25	0	0.25	0.5	1	0.5	0.5	1	0.75
3rd	0	0	0	0	0	0.25	0.5	0.5	0	0.5	0.5	0.5
4th	0	0	0	0	0	0.25	0.25	0.5	0	0.5	0.5	0.5
5th	0	0	0	0	0	0.25	0.5	0.5	0	0.5	0.5	0.75
6th	1.25	0	0	0	0	0.25	0.5	0.75	0.25	0.5	0.75	0.5
7th	0.75	0.5	0.5	1	0.5	0.5	0	0.5	0.5	0.75	0.75	0.5
8th	0.25	0	0	0	0	0.5	0.25	1.5	0.75	1	0.5	0.75
9th	0	0	0	0	0	0.25	0.5	0.5	0	0.5	0.5	0.5
10th	0	0	0	0	0	0.25	0.5	0.5	0	0.5	0.75	1.5
11th	1.75	0.25	0.25	0.25	0.75	0.25	0.5	1	0	0.5	0.5	0.5
12th	1.75	0.25	0.25	0.25	0.75	0.25	0.25	0.75	0.25	0.75	0.5	0.5
13th	0	0	0	0	0	0.25	0.5	0.5	0	0.5	0.5	0.5
14th	0	0	0	0	0	0.25	1.25	0.5	0.25	0.5	0.5	0.75
15th	0	0	0	0	0	0	0.25	0.5	0	0.5	0.75	1.5
16th	1.75	0.25	0.25	0.25	0	0.25	0.25	0.75	0.25	0.5	0.75	1.5
17th	0.25	0	0	0	0	0.25	0.25	0.5	0	0.5	0.75	1.5
18th	1.75	0.25	0.25	0.25	0.75	0.25	1	0.25	0	1	0.5	0.5
19th	4	0.5	0.5	0.5	0	0.5	1	1.25	0.25	0.25	0.5	0.5
20th	1.25	0	0	0	0	0.25	0.5	0.5	0.25	0.5	0.75	0.5
21st	0.75	0.5	0.5	0	0	0.25	0.25	0.25	0.25	0.5	0.5	0.75
22nd	0.5	0	0	0	0	1.25	0.25	0.75	0.5	0.5	0.5	0.5
23rd	1.25	0	0	0	0	0.25	0.25	0.75	0.5	0.75	0.5	0.75
24th	0	0	0	0	0	0.25	0.25	0.5	0	0.5	0.5	1.5
25th	1.75	0.25	0.25	0.25	0.75	0.75	1	0.5	0.5	0.75	0.5	0.5
26th	0	0	0	0	0	0.25	0.25	0.5	0	0.5	0.5	1.25
27th	0	0	0	0	0	0.25	0.5	0.5	0	0.5	0.5	0.75
28th	0	0	0	0	0	0	0.25	0.5	0	0.5	1.25	1.5
29th	0	0	0	0	0	0	0.25	0.5	0.25	0.5	1	1.5
30th	0	0	0	0	0	0	0.25	0.5	0	0.5	0.75	1.25
31st	1.25	0	0	0	0	0.25	0.25	0.75	0.25	0.5	1	1.75

Table 5b: Variable Load Schedule for Second Twelve Hours in Increments of 0.25 kWh

	12	13	14	15	16	17	18	19	20	21	22	23
1st	0.75	1.5	1.5	2	1	1	0.25	0.25	0	0	0	0
2nd	0	0.5	1	1.25	0.5	0.5	0.25	0	0	0.75	0.25	0
3rd	0	0.5	0.5	1	0.75	1	0.25	0	0	5.25	0	0
4th	0	1	1.25	1.25	1	1	0.25	0	0	3.5	0.25	0
5th	0.75	1.25	1.5	1.5	1	1	0.25	0	0	1.75	0	0
6th	1	1.5	1.75	1.5	0.5	0.75	0.25	0	0	0	0	0
7th	0.25	0.75	0.5	1	0.75	0.75	0.75	0.25	0.25	0	0	0
8th	0	0.5	1	1	1	1	0.75	0.25	0.5	0.25	0.25	0
9th	0.25	1.25	1.5	1.5	1	0.75	0.25	0	0	2.5	0.25	0
10th	1	1.75	1.75	1.5	1	0.75	0.25	0	0	0	0	0
11th	0	0.75	1.25	1.5	1	0.75	0.25	0	0	0	0	0
12th	0	1.5	1.25	1.25	0.75	0.5	0.25	0	0	0	0	0
13th	0	0.75	1.25	1.5	0.5	0.75	0.25	0	0	4.25	0	0
14th	0.5	1.5	1.75	1.5	1	1.25	0.25	0.25	0	0	0	0
15th	1.25	1.75	1.75	2	0.75	0.75	0.25	0	0	0	0	0
16th	1.25	1.25	1.5	0.5	0	0.5	0.25	0	0	0	0	0
17th	1.25	1.75	2	1.75	0.5	0.5	0.25	0	0	0	0	0
18th	0.25	0.75	1.25	1.5	0.5	0.75	0.25	0	0	0	0	0
19th	0	0.25	0.5	0.5	0	0.5	0.25	0	0	0.25	0	0
20th	0.75	1.25	1.75	1.75	1	0.75	0.25	0	0	0	0	0
21st	0	1.25	1.75	1.5	1	1.25	0.25	0	0.5	0	0	0
22nd	0.75	1.25	1.5	1.5	1	1	0.25	0	0	0	0	0
23rd	0.5	1.25	1.5	1.75	1	0.75	0.25	0	0	0	0	0
24th	1.25	2	2	1.5	0.75	0.75	0.25	0	0	0	0	0
25th	0.25	0.75	0.75	1	0.25	0.5	0.25	0	0	0.5	0	0
26th	1	1.5	2	1.75	1.25	0.75	0.25	0	0	0.25	0	0
27th	1.25	1.75	2.25	2	0.75	0.75	0.25	0	0	0	0	0
28th	1	1.5	1.75	1.75	1	0.75	0.25	0	0	0	0	0
29th	1	2	1.75	1.25	1	0.5	0.25	0	0.25	0	0	0
30th	1.25	1.75	1.75	2	1	0.75	0.25	0	0	0	0	0
31st	1.25	2	1.25	0.5	0	0.75	0.25	0	0	0	0	0

Figure 4 shows the optimal variable-load schedule with each increment.

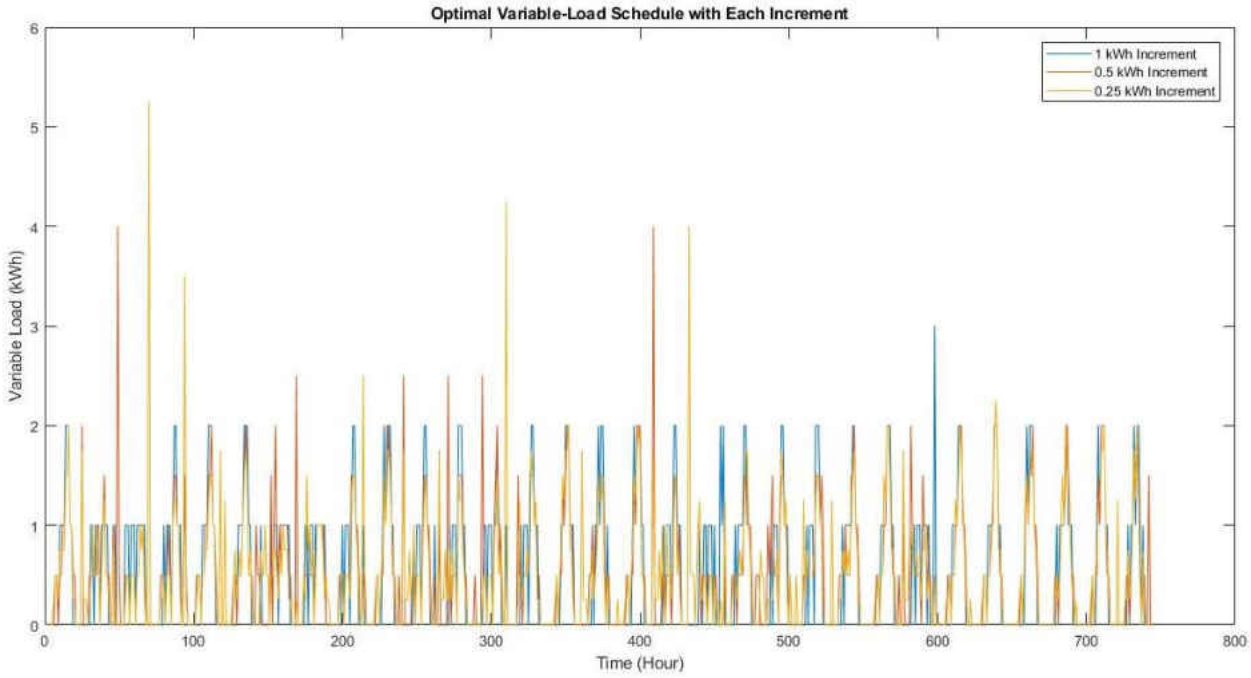

Figure 4: Optimal Variable-Load Schedule with Each Increment

Table 6a shows for the increment of 1 kWh the energy in the battery at the beginning of hours 0 to 11 of each day of the month. Table 6b shows the same data at the beginning of hours 12 to 23 of each day of the month. Similarly, Tables 7a and 7b show the respective data for the increment of 0.5 kWh and Tables 8a and 8b do so for the increment of 0.25 kWh. For all the tables, the unit is kWh, each row represents a day in July, and each column represents an hour of the day. Figure 5 shows the energy in the battery with each increment.

Table 6a: Energy in Battery at Beginning of First Twelve Hours (1 kWh Increment)

	0	1	2	3	4	5	6	7	8	9	10	11
1st	0	0	0	0	0	0.025	0.18715	0.4276	0.4292	0	0	0
2nd	0	0	0	0	0	0.0196	0	0	0.0016	0	0	0
3rd	0	0	0	0	0	0	0	0	0	0	0	0
4th	0	0	0	0	0	0.01945	0.11755	0	0	0	0	0
5th	0	0	0	0	0	0.0219	0.1767	0.4108	0.3958	0	0	0
6th	0	0	0	0	0	0.0226	0.1785	0.4126	0.3976	0	0	0
7th	0	0	0	0	0	0	0	0	0	0	0	0
8th	0	0	0	0	0	0.01425	0	0	0.0016	0	0	0
9th	0	0	0	0	0	0.02035	0.1709	0	0.0016	0	0	0
10th	0	0	0	0	0	0.01755	0.15765	0.39175	0.37675	0.5919	0.1505	0
11th	0	0	0	0	0	0	0.0587	0	0	0	0	0
12th	0	0	0	0	0	0	0.0436	0	0	0	0	0
13th	0	0	0	0	0	0.0132	0	0	0	0	0	0
14th	0	0	0	0	0	0.0128	0.1538	0	0	0	0	0
15th	0	0	0	0	0	0.00995	0.1431	0.38355	0.38515	0.6111	0.3332	0.55335
16th	0	0	0	0	0	0.0084	0.11525	0	0.0016	0	0	0
17th	0	0	0	0	0	0.00705	0.1301	0.3642	0.3492	0.56435	0.11345	0
18th	0	0	0	0	0	0	0.0284	0	0	0	0	0
19th	0	0	0	0	0	0	0	0	0	0.02385	0	0
20th	0	0	0	0	0	0.0036	0.06525	0	0	0	0	0
21st	0	0	0	0	0	0.00295	0.0723	0	0	0	0	0
22nd	0	0	0	0	0	0.00265	0.12455	0	0.0016	0	0	0
23rd	0	0	0	0	0	0.00285	0.1272	0	0.0016	0	0	0
24th	0	0	0	0	0	0.00135	0.111	0.3451	0.3301	0	0	0.00465
25th	0	0	0	0	0	0.00065	0	0	0	0	0	0
26th	0	0	0	0	0	0.0006	0.11485	0.34895	0.33395	0	0	0.02
27th	0	0	0	0	0	5.00E-05	0.11525	0.34935	0.33435	0	0	0
28th	0	0	0	0	0	0	0.1058	0.3399	0.3249	0.61225	0.7087	0.1789
29th	0	0	0	0	0	0	0.03155	0	0.0016	0	0	0.09595
30th	0	0	0	0	0	0	0.1078	0.34825	0.34985	0.5758	0.2846	0
31st	0	0	0	0	0	0	0.0969	0	0	0	0	0

Table 6b: Energy in Battery at Beginning of Second Twelve Hours (1 kWh Increment)

	12	13	14	15	16	17	18	19	20	21	22	23
1st	0	0	0	0	0	0	0.1125	0.1316	0	0	0	0
2nd	0	0	0	0.00065	0	0	0.0758	0.08305	0	0	0	0
3rd	0	0	0	0	0	0	0	0.0208	0	0	0	0
4th	0	0	0	0	0	0	0	0.01935	0	0	0	0
5th	0	0	0	0	0	0	0.1277	0.14665	0	0	0	0
6th	0	0	0	0	0	0	0.11	0.12525	0	0	0	0
7th	0	0	0	0	0	0	0	0	0	0	0	0
8th	0	0	0	0	0	0	0	0	0	0	0	0
9th	0.0766	0.1032	0	0	0	0	0.1203	0.13585	0	0	0	0
10th	0	0	0	0	0	0	0.0782	0.08395	0	0	0	0
11th	0	0	0	0	0	0	0.1135	0.12705	0	0	0	0
12th	0	0	0	0	0	0	0.06895	0.07315	0	0	0	0
13th	0	0	0.0497	0	0	0	0.10975	0.1311	0	0	0	0
14th	0.1044	0.16015	0	0	0	0	0	0.01195	0	0	0	0
15th	0.61705	0.32755	0.18495	0.03885	0	0	0	0.01935	0	0	0	0
16th	0.03095	0	0	0	0	0	0.07705	0.08125	0	0	0	0
17th	0.0277	0	0	0	0	0	0.0947	0.1025	0	0	0	0
18th	0	0	0	0	0	0	0	0	0	0	0	0
19th	0	0	0	0	0	0	0.09975	0.09975	0	0	0	0
20th	0	0	0	0	0	0	0.10465	0.10465	0	0	0	0
21st	0	0	0	0	0	0	0	0	0	0	0	0
22nd	0.06725	0	0	0	0	0	0.10845	0.10845	0	0	0	0
23rd	0	0.1095	0	0	0	0	0.1091	0.1091	0	0	0	0
24th	0	0	0	0	0	0	0.10275	0.10275	0	0	0	0
25th	0	0	0	0	0.0095	0	0.09595	0.09595	0	0	0	0
26th	0	0	0	0	0	0	0.10145	0.10145	0	0	0	0
27th	0	0	0	0	0	0	0.0909	0.0909	0	0	0	0
28th	0.2176	0	0	0	0	0	0.095	0.095	0	0	0	0
29th	0.02315	0	0	0.02545	0	0	0.00995	0.00995	0	0	0	0
30th	0.0422	0	0	0	0	0	0.08785	0.08785	0	0	0	0
31st	0.11225	0	0	0	0	0	0.0917	0.0917	0	0	0	0

Table 7a: Energy in Battery at Beginning of First Twelve Hours (0.5 kWh Increment)

	0	1	2	3	4	5	6	7	8	9	10	11
1st	0	0	0	0	0	0.025	0	0	0.0016	0	0	0
2nd	0	0	0	0	0	0	0	0	0	0	0	0
3rd	0	0	0	0	0	0	0	0	0	0	0	0
4th	0	0	0	0	0	0	0	0	0	0	0	0
5th	0	0	0	0	0	0	0	0	0	0	0	0
6th	0	0	0	0	0	0.0226	0	0	0	0	0	0
7th	0	0	0	0	0	0	0	0	0	0	0	0
8th	0	0	0	0	0	0	0	0	0	0	0	0
9th	0	0	0	0	0	0	0	0	0	0	0	0
10th	0	0	0	0	0	0.01755	0	0	0	0	0.0293	0
11th	0	0	0	0	0	0	0	0	0	0	0	0
12th	0	0	0	0	0	0	0	0	0	0	0	0
13th	0	0	0	0	0	0	0	0	0	0	0	0
14th	0	0	0	0	0	0	0	0	0	0	0	0
15th	0	0	0	0	0	0.00995	0.1431	0.124	0.1256	0.0775	0.18855	0.12885
16th	0	0	0	0	0	0	0	0	0	0	0	0
17th	0	0	0	0	0	0.00705	0	0	0	0	0.02455	0
18th	0	0	0	0	0	0	0	0	0	0	0	0
19th	0	0	0	0	0	0	0	0	0	0	0	0
20th	0	0	0	0	0	0	0	0	0	0	0	0
21st	0	0	0	0	0	0	0	0	0	0	0	0
22nd	0	0	0	0	0	0	0	0	0	0	0	0
23rd	0	0	0	0	0	0	0	0	0	0	0	0
24th	0	0	0	0	0	0.00135	0	0	0	0	0	0.00465
25th	0	0	0	0	0	0	0	0	0	0	0	0
26th	0	0	0	0	0	0.0006	0	0	0	0	0	0.02
27th	0	0	0	0	0	5.00E-05	0	0	0	0	0	0
28th	0	0	0	0	0	0	0.1058	0.074	0.059	0.09635	0.1928	0.163
29th	0	0	0	0	0	0	0	0	0.0016	0	0	0
30th	0	0	0	0	0	0	0.1078	0.0887	0.0903	0.0422	0.1466	0
31st	0	0	0	0	0	0	0	0	0	0	0	0

Table 7b: Energy in Battery at Beginning of Second Twelve Hours (0.5 kWh Increment)

	12	13	14	15	16	17	18	19	20	21	22	23
1st	0	0	0	0	0	0	0	0	0	0	0	0
2nd	0	0	0	0	0	0	0	0.00725	0	0	0	0
3rd	0	0	0	0	0	0	0	0.0208	0	0	0	0
4th	0	0	0	0	0	0	0	0.01935	0	0	0	0
5th	0	0	0	0	0	0	0	0.01895	0	0	0	0
6th	0	0	0	0	0	0.04555	0	0.01525	0	0	0	0
7th	0	0	0	0	0	0	0	0	0	0	0	0
8th	0	0	0	0	0	0	0	0	0	0	0	0
9th	0	0	0	0	0	0	0	0.01555	0	0	0	0
10th	0	0.0455	0	0.00025	0	0.0617	0	0.00575	0	0	0	0
11th	0	0	0	0	0	0	0	0.01355	0	0	0	0
12th	0	0	0	0	0	0.01495	0	0.0042	0	0	0	0
13th	0	0	0	0	0	0	0	0.02135	0	0	0	0
14th	0	0	0	0	0	0	0	0	0	0	0	0
15th	0.19255	0.2978	0.1552	0.0091	0	0	0	0.01935	0	0	0	0
16th	0	0	0	0	0	0	0	0.0042	0	0	0	0
17th	0.0277	0	0	0	0	0	0	0.0078	0	0	0	0
18th	0	0	0	0	0	0	0	0	0	0	0	0
19th	0	0	0	0	0	0	0	0	0	0	0	0
20th	0	0	0	0	0	0	0	0	0	0	0	0
21st	0	0	0	0	0	0	0	0	0	0	0	0
22nd	0	0	0	0	0	0	0	0	0	0	0	0
23rd	0	0	0	0	0	0	0	0	0	0	0	0
24th	0	0	0	0	0	0.0462	0	0	0	0	0	0
25th	0	0	0	0	0	0	0	0	0	0	0	0
26th	0	0	0	0	0	0	0	0	0	0	0	0
27th	0	0	0	0	0	0.0635	0	0	0	0	0	0
28th	0.2017	0.2316	0.3146	0	0	0	0	0	0	0	0	0
29th	0	0.0085	0	0	0	0	0	0	0	0	0	0
30th	0.0422	0	0	0	0	0.1052	0	0	0	0	0	0
31st	0	0	0	0	0	0.0571	0	0	0	0	0	0

Table 8a: Energy in Battery at Beginning of First Twelve Hours (0.25 kWh Increment)

	0	1	2	3	4	5	6	7	8	9	10	11
1st	0	0	0	0	0	0	0	0	0	0	0	0
2nd	0	0	0	0	0	0	0	0	0	0	0	0
3rd	0	0	0	0	0	0	0	0	0	0	0	0
4th	0	0	0	0	0	0	0	0	0	0	0	0
5th	0	0	0	0	0	0	0	0	0	0	0	0
6th	0	0	0	0	0	0.00995	0.0181	0	0.0016	0	0	0
7th	0	0	0	0	0	0	0	0	0	0	0	0
8th	0	0	0	0	0	0	0	0	0	0	0	0
9th	0	0	0	0	0	0	0	0	0	0	0	0
10th	0	0	0	0	0	0	0	0	0	0	0	0
11th	0	0	0	0	0	0	0	0	0	0	0	0
12th	0	0	0	0	0	0	0	0	0	0	0	0
13th	0	0	0	0	0	0	0	0	0	0	0	0
14th	0	0	0	0	0	0	0	0	0	0	0	0
15th	0	0	0	0	0	0	0	0	0	0	0	0
16th	0	0	0	0	0	0	0	0	0	0	0	0
17th	0	0	0	0	0	0	0	0	0	0	0	0
18th	0	0	0	0	0	0.00E+00	0	0	0	0	0	0
19th	0	0	0	0	0	0	0	0	0	0.03735	0	0
20th	0	0	0	0	0	0	0	0	0	0	0	0
21st	0	0	0	0	0	0	0	0	0.0016	0	0	0.0447
22nd	0	0	0	0	0	0	0	0	0	0	0	0
23rd	0	0	0	0	0	0	0	0	0	0	0	0
24th	0	0	0	0	0	0	0	0	0	0	0	0
25th	0	0	0	0	0	0	0	0	0	0	0	0
26th	0	0	0	0	0	0	0	0	0	0	0	0
27th	0	0	0	0	0	0	0	0	0	0	0	0
28th	0	0	0	0	0	0.00995	0.0181	0	0.0016	0	0	0
29th	0	0	0	0	0	0	0	0	0	0	0	0
30th	0	0	0	0	0	0	0	0	0	0	0	0
31st	0	0	0	0	0	0	0	0	0	0	0	0

Table 8b: Energy in Battery at Beginning of Second Twelve Hours (0.25 kWh Increment)

	12	13	14	15	16	17	18	19	20	21	22	23
1st	0	0	0	0	0	0	0	0	0	0	0	0
2nd	0	0	0	0	0	0	0	0.00725	0	0	0	0
3rd	0	0	0	0	0	0	0.0068	0.0276	0	0	0	0
4th	0	0	0	0.0047	0	0	0.0019	0.02125	0	0	0	0
5th	0	0.00475	0	0.00015	0	0	0.0027	0.02165	0	0	0	0
6th	0	0	0	0	0	0	0	0.01525	0	0	0	0
7th	0	0	0	0	0	0	0	0	0	0	0	0
8th	0	0	0	0	0	0	0	0	0	0	0	0
9th	0	0	0	0	0	0.02075	0.01135	0.0269	0	0	0	0
10th	0	0	0	0.00025	0	0	0	0.00575	0	0	0	0
11th	0	0	0	0	0	0.0033	0	0.01355	0	0	0	0
12th	0	0	0	0	0	0.01495	0	0.0042	0	0	0	0
13th	0	0	0	0	0	0	0	0.02135	0	0	0	0
14th	0	0	0	0	0	0	0	0	0	0	0	0
15th	0	0	0.0537	0	0.0446	0.04585	0.01685	0.0362	0	0	0	0
16th	0	0	0	0	0	0	0	0.0042	0	0	0	0
17th	0	0	0	0	0	0	0	0.0078	0	0	0	0
18th	0	0	0	0	0	0	0	0	0	0	0	0
19th	0	0	0	0	0	0	0	0	0	0	0	0
20th	0	0	0	0	0	0	0	0	0	0	0	0
21st	0	0	0	0	0	0	0	0	0	0	0	0
22nd	0	0	0	0	0	0	0	0	0	0	0	0
23rd	0	0	0	0	0	0.01495	0	0	0	0	0	0
24th	0	0	0	0	0	0	0	0	0	0	0	0
25th	0	0	0	0	0	0	0	0	0	0	0	0
26th	0	0	0	0	0	0.00785	0	0	0	0	0	0
27th	0	0	0	0	0	0	0	0	0	0	0	0
28th	0.0387	0.0686	0	0	0	0	0	0	0	0	0	0
29th	0	0	0	0	0	0	0	0	0	0	0	0
30th	0	0	0.05095	0	0	0	0	0	0	0	0	0
31st	0	0	0	0	0	0	0	0	0	0	0	0

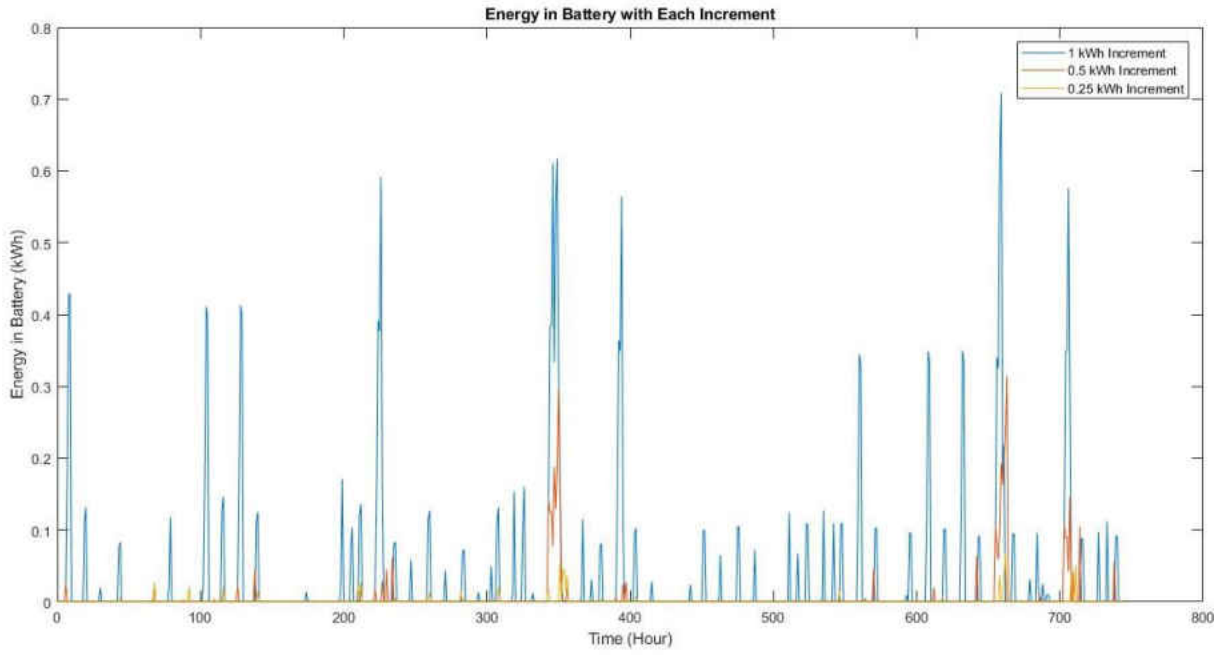

Figure 5: Energy in Battery with Each Increment

Variation – Variable Load for Day is Randomly Determined within Range

The process used to develop the variable load schedule and obtain the cheapest cost is quite similar to that used for the previous case. The key difference is that instead of the total variable load invariably summing to 12 kWh each day, it adds up to a random quantity within the range 9 kWh to 15 kWh.

One point of note is that the values that can be chosen within this range are restricted to multiples of the increment – multiples of 1, 0.5, or 0.25. All three options are studied and the results are displayed in Table 6.

Another notable mention is that the element of randomness in the quantity of variable load to meet on a certain day will likely cause the minimum cost and optimal load schedule to vary for every run of the program; however, because the variable load is randomly determined 31

times (for every day of the month of July), the randomness results in a reasonably uniform distribution and thus a single run of the program sufficiently represents typical values.

Running time is omitted from this variation because the primary case offers better relative perspective on this metric.

Table 9: Total Minimum Cost for Each Increment (Variation)

Increment	Total Minimum Cost
1 kWh	$23.97
0.5 kWh	$22.40
0.25 kWh	$21.01

Chapter 5.2 – Analysis

Primary Case – Constant Quantity of Variable Load for Day

A randomized practice to develop the load schedule is introduced to gain perspective into the significance of the reduction in cost offered by the dynamic-programming approach. This new procedure, used for comparative purposes, sets a random quantity of variable load for each hour of the day. This algorithm pays no heed to efficiency concerns; the only limitation placed on this method is that as before, the quantity of variable load for each day needs to total 12 kWh. A variable load schedule is thus developed for the whole month of July and the respective cost is obtained.

Table 10 shows the cost of the randomized algorithm as well as the cost improvements of dynamic programming over that process.

Table 10: Comparison Between Randomized Algorithm and Dynamic Programming

Algorithm/Increment	**Cost**	**Improvement over Randomized Algorithm**
Randomized/NA	$28.55	N/A
Dynamic Programming/1 kWh	$22.53	21.1% ($6.02)
Dynamic Programming/0.5 kWh	$21.57	24.4% ($6.98)
Dynamic Programming/0.25 kWh	$21.31	25.4% ($7.24)

The differences shown by Table 10 are beyond marginal and prove that dynamic programming offers considerable savings.

The tables and plot showing energy in the battery clearly have the smallest values when the increment size is also smallest, proving that smaller increment sizes allow more solar energy to be directed to the load before charging the battery. However, the additional precision that is provided by increments of 0.25 kWh comes with a significant tradeoff in increased running time and only a marginal improvement in the total cost, as shown in Table 11.

Table 11 documents the tradeoff between smaller increments/cost improvements and increased running times.

Table 11: Tradeoff Between Smaller Increments/Cost Improvements and Increased Running Times (Primary Case)

Increment	**Cost Improvement over Previous Increment**	**Running Time Increase over Previous Increment**
1 kWh	N/A	N/A
0.5 kWh	4.3% ($0.94)	147% (~1 s)
0.25 kWh	1.2% ($0.26)	311% (~ 5 s – 6 s)

Table 11 shows that 0.5 kWh offers a nice balance between running time and cost efficiency. For this reason, it is the best one to use for this problem.

Variation – Variable Load for Day is Randomly Determined within Range

The relationship between increment size and minimum cost carries over from the primary case to this variation. Table 12 shows the relationship between smaller increment sizes and improvements in total minimum cost. Running time is omitted from this variation as a topic of focus because the primary case offers more suitable grounds (without randomness) for that study.

Table 12: Relationship Between Smaller Increments and Cost Improvements (Variation)

Increment	**Cost Improvement over Previous Increment**
1 kWh	N/A
0.5 kWh	6.5% ($1.57)
0.25 kWh	6.2% ($1.39)

The improvement obtained by the 0.25-kWh increment over the 0.5-kWh increment is higher than that obtained by using the 0.25 kWh for the primary case and is comparable to the improvement obtained by the 0.5-kWh increment over the 1-kWh increment in this variation, but because of the discussion regarding running time in the primary case coupled with the fact that the reductions in cost will vary in this variation due to randomness, the best increment size still appears to be 0.5 kWh for this problem.

Chapter 6 – Conclusion and Future Work

This project has proven that dynamic programming is a robust and innovative way to solve the problem of cost optimization with hybrid renewable energy systems. As shown, it leads to a reduction in cost compared to a randomized approach to developing a load schedule. The technique described is also quite flexible, as the process remains similar regardless of details such as type of building, time of year, or geographic location. The procedure may be easily adjusted by substituting the relevant parts.

There are many avenues of expansion to this project. The following is a non-exhaustive list of possible ideas.

1) A comparison could be made between the savings that result from using dynamic programming for the month of July and those that may be obtained from using the same algorithm on a cold month, such as December. During such times of the year, the savings may or may not be as significant as those during warmer months. Similarly, the type of building and geographic location could be varied to develop a more general understanding of the efficiency of the approach described in this thesis.

2) Rather than requiring that the variable load remain constant or within a range for the whole day, this constraint could instead be applied to a week or month. Making this change could result in a cheaper cost, but may also appear to be less realistic, as operating electrical devices on such a tight schedule for long periods of time may be unfeasible. Also, dynamic programming tends to get inefficient as the time frame is lengthened.

3) Instead of considering a month as the window of time over which the cost must be optimized, the scope could be widened by focusing on several months or even years.

4) Instead of always allowing variable loads to be split into smaller quantities, introduce a restriction that certain loads may not be divided. In reality, this limitation would correspond to a certain appliance or other electrical device that necessarily consumes a minimum quantity of energy when in operation.

5) The battery for this project was a simple bank of energy that only considered a round-trip efficiency. This device could be made more complex and realistic, such as by introducing lifetime and a maximum capacity, especially if this technique is conducted for several months or years instead of only one month. In the same vein, the efficiency and lifetime of solar PV cells may also be factored into the problem.

6) Snow cover on the solar PV cells may detract from the amount of energy available to the system. This may be added as a factor for colder months.

7) Wind energy could be included as an additional source of renewable energy. This would require understanding of the details of its production and storage.

8) Other techniques and algorithms, especially machine learning, may be implemented to perhaps improve on the cost savings offered by dynamic programming.

References

[1] N. M. Ijumba, V. T. Raphalalani and P. Reddy, "Optimised Application of Renewable Energy Sources in Rural," in *IEEE Africon. 5th Africon Conference in Africa*, Cape Town, South Africa, 1999.

[2] P. M. Corrigan and G. T. Heydt, "Optimized Dispatch of a Residential Solar Energy System," in *39th North American Power Symposium*, Las Cruces, NM, United States, 2007.

[3] M. Lamnadi, M. Trihi and A. Boulezhar, "Study of a hybrid renewable energy system for a rural school in Tagzirt, Morocco," in *International Renewable and Sustainable Energy Conference (IRSEC)*, Marrakech, Morocco, 2016.

[4] S. Salehin, A. K. M. S. Islam, R. Hoque, M. Rahman, A. Hoque and E. Manna, "Optimized Model of a Solar PV-Biogas-Diesel Hybrid Energy System for Adorsho Char Island, Bangladesh," in *3rd International Conference on the Developments in Renewable Energy Technology (ICDRET)*, Dhaka, Bangladesh, 2014.

[5] J. Reddy and D. Reddy, "Probabilistic performance assessment of a roof top wind, solar photo voltaic hybrid energy system," in *Annual Symposium Reliability and Maintainability*, Los Angeles, CA, United States, 2004.

[6] B. Zientara, "How much electricity does a solar panel produce?," Solar Power Rocks, [Online]. Available: https://www.solarpowerrocks.com/solar-basics/how-much-electricity-does-a-solar-panel-produce/. [Accessed 2nd Dec. 2018].

[7] Victron Energy, "12,8 Volt Lithium-Iron-Phosphate Batteries," [Online]. Available: https://www.victronenergy.com/upload/documents/Datasheet-12,8-Volt-lithium-iron-phosphate-batteries-EN.pdf. [Accessed 2nd Dec. 2018].

[8] A. Thomas, "kW vs. kWh: How much energy is my lighting using?," Regency Lighting, 16th Jun. 2016. [Online]. Available: https://insights.regencylighting.com/kw-vs-kwh-how-much-energy-is-my-lighting-using. [Accessed 2nd Dec. 2018].

[9] IGS Energy, "How Much Electricity Do My Home Appliances Use?," IGS Energy, [Online]. Available: https://www.igsenergy.com/energy-resource-library/electricity-articles/how-much-electricity-do-my-home-appliances-use/. [Accessed 2nd Dec. 2018].

[10] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, "Dynamic Programming," in *Introduction to Algorithms*, Third ed., Cambridge, MA, United States, MIT Press, 2009, pp. 359-413.

[11] R. E. Bellman and S. Dreyfus, "The Principle of Optimality," in *Applied Dynamic Programming*, Princeton, NJ, United States, Princeton University Press, 1962, p. 15.

[12] A. K. Dixit, "Dynamic Programming," in *Optimization in Economic Theory*, Second ed., New York, NY, United States, Oxford University Press, 1990, pp. 162-180.

[13] S. Caban, "When the Bellman Equation Cannot Be Solved Analytically," University of Vienna, Vienna, Austria, 2010.

[14] A. Nasiri, Discussion Participant, *Thesis Topic.* [Discussion]. May 2018.

[15] N. Blair, N. DiOrio, J. Freeman, P. Gilman, S. Janzou, T. Neises and M. Wagner, "System Advisor Model (SAM) General Description (Version 2017.9.5)," National Renewable Energy Laboratory, Golden, CO, United States, 2017.