Team 11623

2021 HiMCM
Problem A: Storing the Sun

Table of Contents

Summary

Our fundamental approach to this problem was to write a program that calculated the optimal battery and solar panel configuration for any location in the United States. We decided to optimize for solar panels as well as batteries because having many solar panels reduces the number of batteries needed during dawn and dusk, making it impossible to optimize for one without the other. To start, our first step was to find and compile the immense amount of data needed to be able to account for that much variability of input. To choose any location in the US to construct this theoretical house, we needed databases for thousands of locations with accurate data to be able to approximate the user's input to a general region. The specific data required was solar intensity measurements, to determine the output of our solar panels, as well as temperature data to determine when heating the house was necessary–the largest draw of electricity in any household with electric heating. Once we had our data (over 17 GB) for a sufficient number of locations across the United States, the databases needed to be parsed and compressed into smaller files with only the imperative information for our simulation. The program we wrote to sort through this data was able to write files with only the metric we needed for only the locations we deemed necessary to accurately measure every climate zone across the country (over 5000 unique locations for the temperature data). The next step was the simulation. This program would take input from the user and output the cheapest configuration for the types and number of batteries as well as number of solar panels that would allow for the house to maintain 90% electrical uptime. The user inputs an exact longitude and latitude coordinate they would like their house to be theoretically constructed at. The program then immediately calculates the nearest location where data was collected for both solar and temperature data using the Haversine formula, an equation used to find the distance between two points on a sphere. With the data for this location, the program then simulates over 10 years of usage using random data points throughout the years of collected data to approximate the changes in seasons, weather, length of days, and other factors that vary throughout the year. The program tests every needed combination of different types and amounts of batteries as well as solar panels in ascending order of price until it finds one that passes our 90% uptime threshold. Each combination is tested using a Monte-Carlo simulation which calculates the net power every hour, fills and drains the batteries accordingly, and if the configuration ever runs out of power in any given hour it is counted as a failure. If the amount of failed hours in the simulation exceeds 10% of the total simulated hours then the next configuration is tested. Batteries are filled every hour by calculating the output of the solar panels using solar intensity data from the databases for that hour in the year, filling efficiency of the batteries, and efficiency of the panels themselves. The draining method in the program is calculated using energy drawn in that hour according to a base load and whether the electric heater is running as well as peak outputs and capacities of the batteries filled during the daylight hours. All of these calculations are done in parallel on the CPU to decrease computation time, and the output is the optimal configuration of batteries and solar panels *for any location in the entire country* to be able to survive with operational electricity with 90% electrical uptime.

Understanding the Problem

First Steps

After reading the problem, our first step was to determine the function of the model. First, we needed to know the goal for the model, and second, what the inputs and output parameters would be. In Problem 1.C, it asks for “the best battery storage option,” which we interpreted to mean the cheapest reasonable solution capable of providing power to the home. This allowed us to determine our metric for success:

A solution is deemed “best” if and only if:
1. It is able to power the home above a certain threshold of time, i.e. 90% of the day
2. It is a reasonable solution for a domestic environment, i.e. there are no noxious fumes
3. It is the cheapest solution capable of meeting criteria 1 and 2

Brainstorming

Our next step was to brainstorm the parameters and other questions for our model. Some of our ideas can be found in Table 1.

- How many people will be living in the home - user input
- Where is the home geographically - randomized by program based on point database
- How much sunlight does the home get - based on randomly chosen location and probability of cloud cover
- What appliances do the people in the home use - make a list? Just average energy usage per person?
- How much energy per day/week/month/year does an average person use?
- What type of solar panels would be used?
- How much energy do appliances use and how much does that cost?
- How often do solar panels break / fail?
- How often do batteries break / fail?
- What other battery options are on the market?
- How often does the national grid break down?
- What does “best” mean in relation to an energy / solar system?
- How do different weather conditions affect energy collected by the panels?
- What’s the average cloud coverage of the area?
- How many hours of sunlight does the location receive?
- What is the strength of that sunlight?
- Average location data?

Table 1: Early Questions

Analysis

The most basic simplification of a problem such as this is a series of inputs that give an output. From what we understand, the user should be able to give specifications of the household (number of residents, etc.) as well as how many years they expect to retain residence and receive the most optimal configuration of batteries and solar panels that will maintain a given threshold uptime. The model used to carry out these calculations would take into consideration the user inputs, values acquired from research, and data from our chosen databases to choose the cheapest successful model. Research was conducted throughout the process as knowledge gaps arose primarily utilizing the internet and public databases. Some research included new real world solar panels and batteries to supplement our options, java methods and examples, and certain math concepts which would be useful in the production of our model.

Assumptions

Our next step was to figure out our assumptions. Here is our final list:

1. The house can be built anywhere in the U.S.
    - Justification: The data was given was in US dollars
2. The best battery system will be the cheapest combination that supplies all needed power while retaining a sufficient uptime
    - Justification: One of the benefits of the grid is its reliability, so this solution must be able to supply electricity for a large percent of the time.
3. Weather conditions (diffuse horizontal irradiance, air temperatures) from 15 years of sampled data in the databases are representative of general future trends
4. All batteries can be connected to one another regardless of voltage or other characteristics
5. The solar panels, batteries, and any additional hardware are installed outside of the home
    - Justification: Most real-world solar applications house batteries, inverters, and other hardware are installed on the exterior of a home
6. The use of a heating system is dependent on air temperature
    - If the temperature drops below a certain threshold in a given day, the heat will be turned on for the day using average heating runtimes for the United States
7. Solar panels are horizontal on the ground (tangent to the Earth, with the Earth being treated as a perfect sphere)
8. Battery Efficiency is applied during the charging phase
    - For example, charging at 100w for one hour at 95% efficiency fills a battery by 95Wh, but a 100Wh battery can discharge the full 100Wh.

Detailed Model

Initial Thoughts

This is a description of the entire model and all of its stages, which will be broken down into sub-models later. First, we decided on the final inputs, outputs, and data requirements:

External Data	Input Parameters	Outputs
- Sunlight Intensity by Hour by Location - Minimum Temperature by Location - Other battery options - Which Solar Panel	- Location - Duration of Stay - Number of Household Members	- Number of Each Battery - Number of Solar Panels - Cost for the Stay Duration

Table 2. Program Requirements and Goals.

External Data

Before we could build our model, we had to collect more data on sun intensity, temperature, and battery and solar panel options. For solar intensity, we used the National Solar Radiation Database and found 873 locations across the United States with sufficient hourly solar intensity information from 1991-2005 totaling 357MB of data. We then organized this data by location using longitude and latitude coordinates. For the minimum temperature, we found a user 3danim8 who had compiled data from the National Oceanic and Atmospheric Administration (NOAA), and we found that 5,931 locations across the US had at least 15 years of minimum daily temperature data, totaling 17.7GB.

For batteries, we found batteries made by LG and BYD which we included in our possibilities, and decided to discard the Trojan batteries, as that type of battery can produce toxic gases while charging, something not suitable for domestic use.

For solar panels, we looked at a variety of panels, as can be seen in Table 3. We decided to go with the Longi #LR6-60-HPB-350M, which cost $0.60 per watt, has an area of 1.82 m^2, and an efficiency of 19.2%.

Type of Solar Panel	Price per watt
Renogy	1.25
LONGI #LR6-60-HPB-350M	0.6
Canadian Solar	0.71
Silfab Solar	0.76
LONGI #LR4-60-HPH-360M	0.72
Hanwha Q Cell	0.7
JA Solar Technology	0.73

Table 3. Solar Panels and their cost efficiencies.

Energy

We deemed it necessary to optimize for solar panels as well as batteries because the energy production and storage are linked in a way that one cannot be solved without the other. Consider Table 4. The red lines show when, on a given day, the solar panels begin and cease to produce a surplus of energy compared to baseline usage. In the first example, there are many solar panels, and so there is a shorter time when the batteries must supply electricity to meet baseline load (shown by the areas outside the red lines). This means that fewer batteries are needed, but more solar panels, both of which factor into the cost. For the second example, fewer solar panels are used, but there is a longer period of time when the batteries must supply the electricity, saving cost on panels but increasing the cost of the batteries. Thus, to minimize cost, both variables must be optimized.

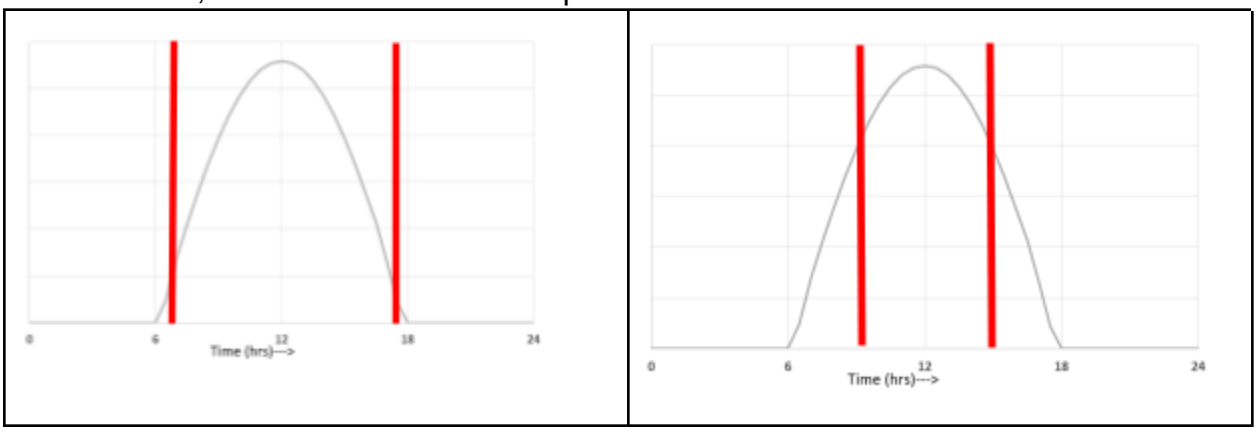


Table 4. Two examples of times of a surplus of solar energy.

Energy Production

The energy calculations performed by the simulation are much more complicated than an output from our solar panels and a usage rate of the house. Since the simulation will run on an hour-by-hour basis, we have to know the output of our solar panels varies in that time. This can be given by the equation:

$$P_{output} = P_{sun} * A_{panel} * \eta_{panel}$$

Equation 1. Solar panel power production, where P is power, A is surface area, and η is efficiency.

In order to use this equation, the first piece of information needed is sunlight data collected in a relevant unit of measurement. After many hours of searching through public databases we landed on one collected by the National Renewable Energy Laboratory. This database provided years and years worth of hourly solar radiation intensity. The solar radiation data was used to calculate production of the solar panels in our model, which gave us the energy that could be collected by the panels and used to charge the batteries. The solar radiation was collected as watt hours per meter squared per hour ($Wh/m^2/hr$) for every hour at every location. These values were then multiplied by efficiency and amount of the chosen solar panels as well as by the area of the solar panel configuration. With all these values found in research, read from databases, and given in the problem, the final output of watt hours for any given hour in the year available to use in the house and be stored in batteries could be defined.

Energy Usage

Energy usage depends on many factors, from the size of the house to the number of people. We split energy usage into two categories - baseline and heating. The baseline load runs all the time, and the heating load only runs when it is cold. We found that 46% of the energy usage for a single family home comes from heating, so we used Equation 2 for the hourly energy consumption of a 1600 sq. ft. house:

$$(1426.0 - (96 * (6 - numPeople))) / 730.0 * 0.54$$

Equation 2. Hourly energy usage by a 1600 sq. ft. house by number of people.

For energy due to heating, we decided to have a household turn on the heat if the minimum temperature during the day dropped below 5C. We also found that on modern high-efficiency heating systems, peak power draw is about 10.5 KW, and the systems run for about 80% of the time. This allows us to vary heating usage by location, as colder climates will use heat for more of the year.

Cost Calculations

To calculate the cost, use Equation 3. It sums the cost of all of the solar panels and batteries to find the cost to install the system.

$$C = \Sigma C_{Solar\ Panels} + \Sigma C_{Batteries}$$

Equation 3. Total cost (C) for batteries and solar panels.

Batteries

Along with the question posed, we were given a table of five common batteries for solar energy storage. These batteries were meant to provide us with an idea of the options available for our needs, but they were nowhere near a full list of possibilities. To increase the accuracy of our model, we researched general types of batteries, and more specific solar storage solutions. To understand the possible options, we look at the five examples provided. The list of five batteries can be split into two categories based on chemical composition. The first two items, the Deka and Trojan brand batteries, are of the lead-acid type. The basic construction of these batteries can be explained as suspending lead plates in tubs of sulfuric acid. The lead plates serve as the cells of the battery, and their reaction with the acid surrounding them is what produces electricity. The sulfuric acid is an electrolyte in this system, and in the process of this chemical reaction, the sulfur escapes the acid and comes in direct contact with the lead plates. Eventually, the acid solution is depleted of its sulfate content, and the battery is discharged and requires recharging, where sulfates are forced back into the acid solution. This battery technology was one of the first developed, and works well for many applications. When it comes to our specific problem, however, lead-acid batteries are not an optimal fit. In the construction of this house, we are looking for 100% of our electricity consumption to be obtained from solar energy. Along with this requirement, we would want to minimize the cost of the operation by only purchasing the least amount necessary of batteries and solar panels. For the Trojan Flooded-Lead Acid battery, toxic gases can be released during the charging cycle, so we eliminated this battery from our possibilities. The LG Chem RESU 16 H Prime also beat the Tesla Powerwall+ on every statistic while costing the same, so we eliminated the Powerwall+ from our simulation as well. Lithium batteries provide a different type of technology that is much more conducive to our situation. This newer technology allows for faster collecting and dispensing of energy, and also has the capability of very deep cycles, in some instances nearly up to 100%. Lithium batteries have two main subcategories, lithium iron phosphate (LFP) and nickel manganese cobalt oxide (NMC). NMC batteries are capable of deep cycles, but fall short in regards to rapid recharging and discharging. Specifically for our off-the-grid needs, we need fast recharging times as we are solely reliant on collecting energy during sunlight hours to carry us through the night. NMC batteries are limited in their charging rates, and do not absorb all possible sunlight, and this is where LPF batteries have an advantage. It is essential that we collect as much sunlight energy as is available, so utilizing some LPF type batteries would be the best option for our purposes.

Setup

For the setup for our solar array, the panels are put flat on level ground in an area that has sunlight from sunrise to sunset with no obstacles.

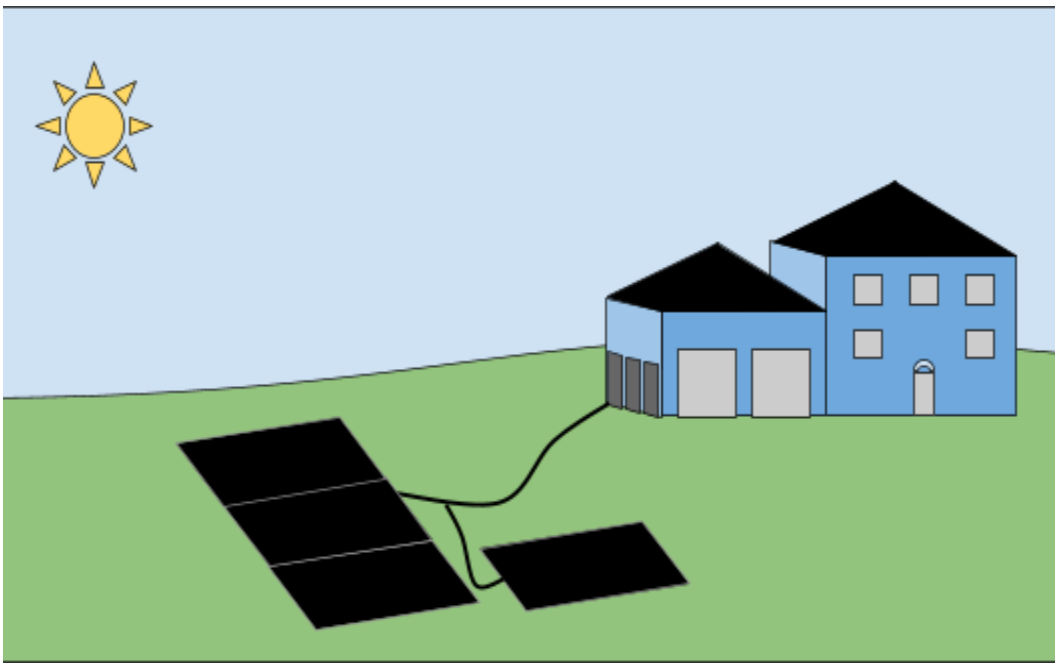
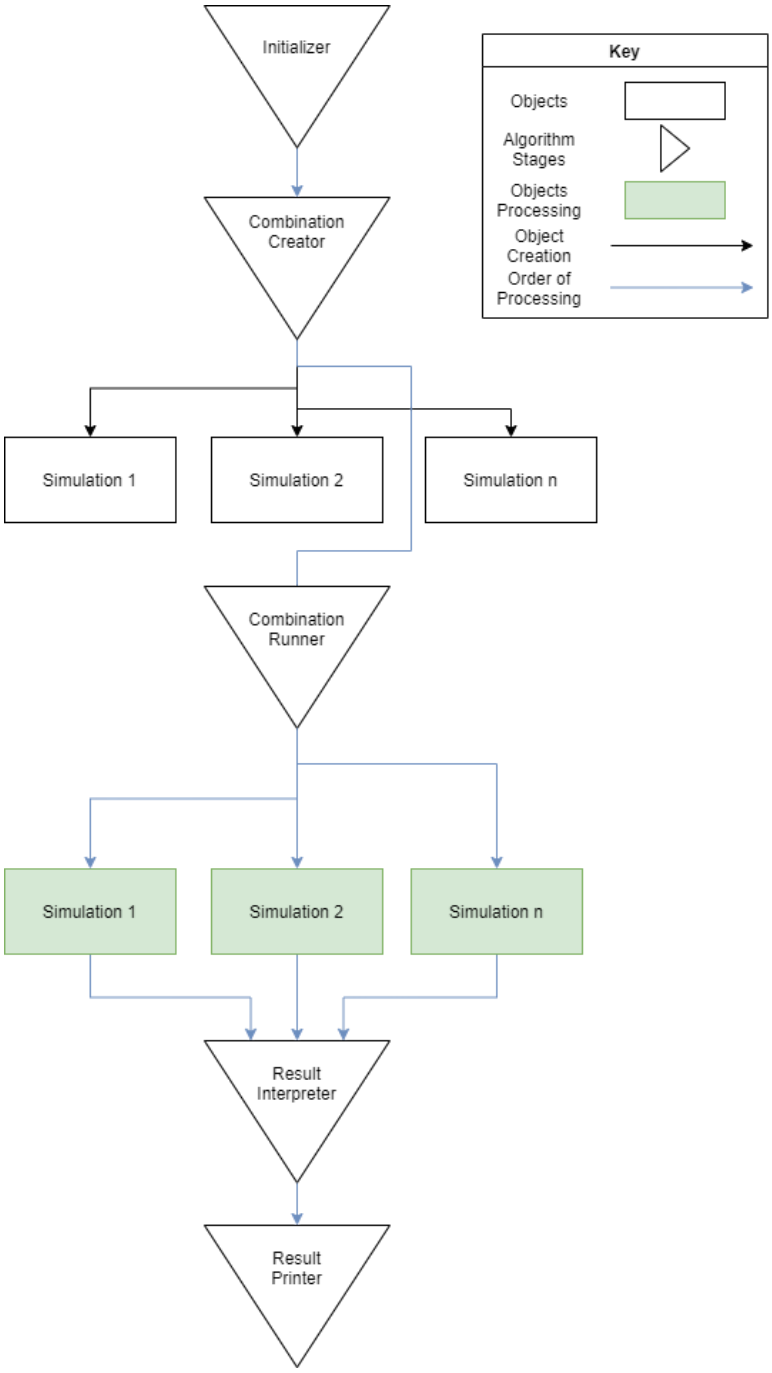


Figure 1. Our solar setup.

Diagram



We then made a diagram of all of the stages our algorithm would need to go through. Since our model is dynamic with respect to the probabilities, we decided the best method to test the probabilities was to use a Monte-Carlo Method - we would randomly simulate the world for a long period of time to see how often one particular combination of batteries failed. This would be repeated for every combination of batteries until we found the cheapest option. The algorithm was implemented in Java. Figure 2 shows the outline of the program's stages.

Figure 2. Diagram of the flow of the entire model.

Data Processor

The first step in running our model was to process all of the data and compile it into an easy-to-access format. To do so, we read through all of the .csv files we downloaded from the databases and extracted the data we needed, separated it by location, and wrote it to new files separated by location.

Initializer

The first step in the program is to initialize everything and make sure all of the data is collected. A diagram of the initializer's functions is found in Figure 3.

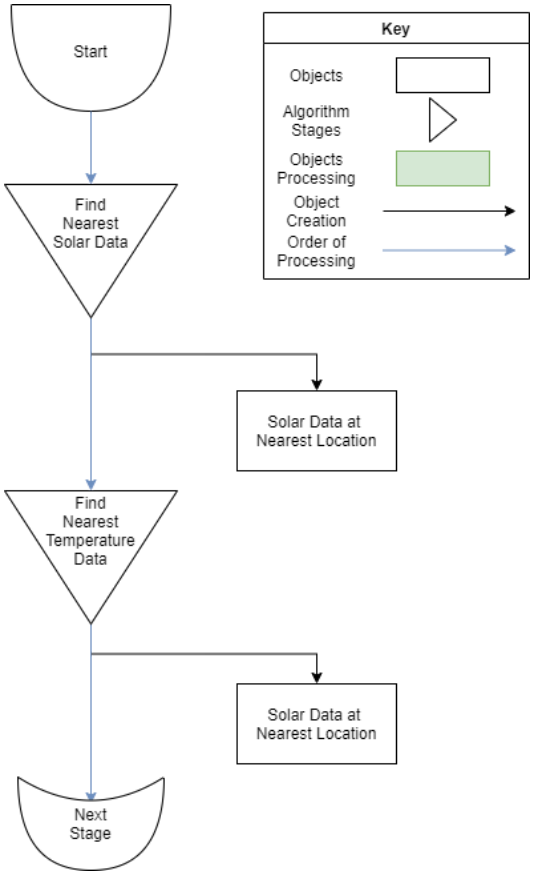


Figure 3. Initializer stage steps.

Nearest Data

In order to find the nearest data point for each of the solar and temperature databases, we need to be able to compute the distance between the user-selected point and the points for which we have data. However, since these are points on a sphere, the standard cartesian distance formula will not work, as we need distance ‘as the crow flies.’ The formula for this distance is the Haversine Formula, which can be seen in Figure 4 and Equation 4.

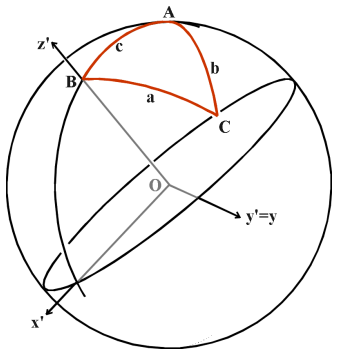


Figure 4. Diagram of distances on a sphere.

$$d = 2r \sin^{-1}\left(\sqrt{\sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\psi_2 - \psi_1}{2}\right)}\right)$$

Equation 4. Haversine formula, where Φ is latitude, r is radius of the Earth, and ψ is longitude

After the files for the nearest data are found, the combination creator creates many possibilities of battery configurations in increasing order of cost, so the lowest cost is among the first ones to be computed.

Combination Creator

The next stage in the program is the combination creator. After analyzing the batteries, we figured out that Discover batteries are necessary for the max output and DEKA batteries are necessary for capacity. This reduced the number of possibilities enormously. The combination creator then runs through all of the possibilities of DEKA batteries and solar panels and creates a Simulator object for each one, with the number of Discover batteries set at the minimum required to meet peak output. A diagram of the Combination Creator can be found in Figure 5.

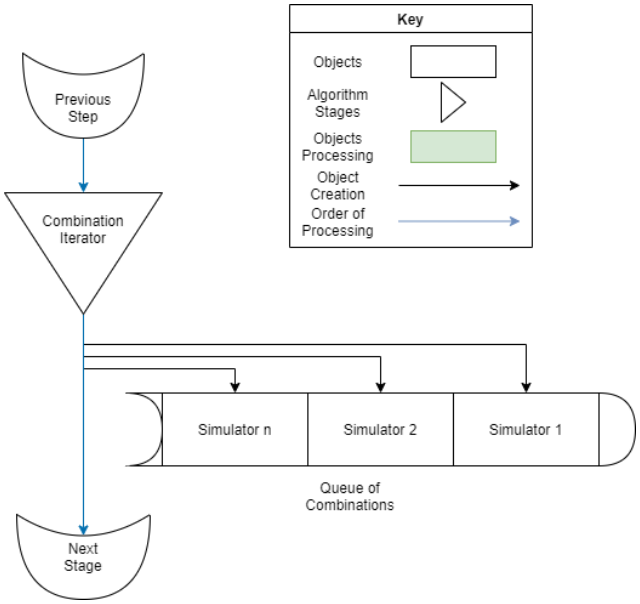


Figure 5. Combination Creator

The Combination Creator's main component is the iterator, which works through the possibilities of batteries and solar panels in increasing cost order. It does this through the use of nested "for" loops, which iterate through every possibility up to a maximum value, and then increase the maximum value when every possibility up to that point has been tested. An example output with three for loops looks like this:

000 001 010 011 100 101 110 111 200 201 202 210 211 212 220 221 222

This is tested to an absolute maximum above the number of items needed to be successful.

This algorithm which tries every possibility has a poor computational complexity, as for whatever absolute maximum 'n' is set for the number of each item, it has to iterate through $n^{\text{number of items}}$ possibilities. Due to the fact that each simulation is a Monte-Carlo simulation, which are computationally intensive, we decided to do a low-precision pass through with a greater probability of false positives and negatives for each simulation, and then a longer, high-precision "check" simulation for all possibilities of batteries that pass the initial round to ensure that each one is not a 'fluke.' This saves enormously on computational time while only increasing the probability of a missed cheaper successful set of batteries by a very small amount.

As they are created, the Simulator objects are stored in a queue. The Queue data structure is an ordered list, with the order determined by the "First In, First Out (FIFO)" rule. This ensures that the cheapest possibilities are stored and processed first as to not waste any computational time later on searching through unnecessary possibilities.

Combination Runner

The Combination Runner stage takes each Simulator object and runs it to compute the result, which is both an uptime percentage and a boolean value of whether it passes the uptime threshold. By iterating through the queue in order, it computes the cheapest possibilities first and the more expensive ones later, meaning that among the first few solutions computed is the cheapest overall solution.

However, the large computational time for each simulation meant that running through them one-by-one was impractical. Because each Simulator object can be run independently, we decided to implement multithreading into our program, as can be seen in Figure 6.

```
public class Simulator extends Thread {
```

Figure 6. Implementation of Multithreading in Java.

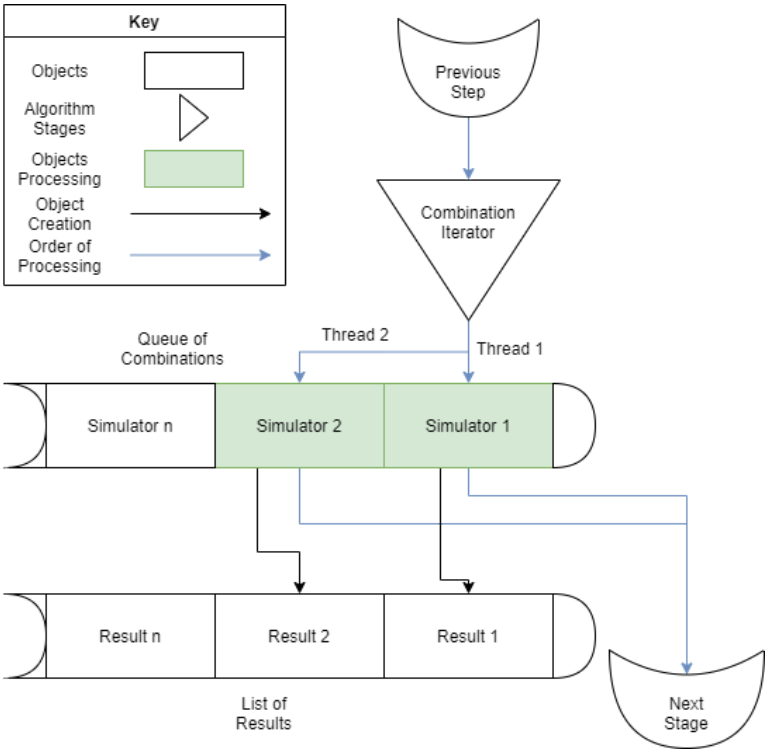


Figure 7. Combination Runner Diagram

On an 8 core 16 thread processor, the fastest we had available, we ran 10 simultaneous computations, which decreased our computational time by a factor of almost 10 (running multiple threads on modern CPUs can decrease the clock speed on each thread by a small amount, which is why the improvement is less than perfect scaling). Implementation of multithreading comes with one problem: dealing with “out-of-order execution.” Modern CPUs have multiple cores to allow for simultaneous

execution of multiple programs - this is why it is possible to have a few programs open at the same time without anything slowing down. On these CPUs, different physical cores run at different clock speeds at given points in time depending on temperature, physical design, and other factors. This means that the first thread started may not be the first one to finish: they will be in roughly the same order, but *not exact*. This means that even though we run the cheapest simulation first, it might not be the first to finish. While more complicated applications use precise thread-synchronization algorithms, ours simply waits for the first ten threads that return a successful value and picks the cheapest one. This makes it very likely that among those ten first solutions one is the cheapest. (This out-of-order multithreading is especially relevant with Apple's M1 series and Intel's Alder Lake processors, as each have distinct performance and efficiency cores that run at different speeds, so threads cannot be assumed to be roughly synchronized.)

Simulator

Each simulator object runs a Monte-Carlo Simulation in order to determine the portion of the time there is sufficient electricity to meet both peak demand and continuous demand. A diagram of a simulator is found in Figure 8.

When a simulator is run, it calculates the starting cost of buying the batteries and solar panels. It then iterates hour by hour through the specified number of years. Each hour, it calculates the panel production and power use for that hour of that day in the year. It gathers the panel production data by selecting a random year in the file for solar intensity at the nearest location and getting the data for that hour in that year. For example, if the simulation is currently simulating 11:00 AM on March 15, it will pick a random year 1991-2005 and get the solar intensity from 11:00 AM on March 15 of that year. Next, the power draw is calculated. If that day's minimum temperature is over 5C, it just uses the baseline load. If it is under 5C, the continuous power draw is set to the baseline load + (10.5KWh * 0.8), as the heaters use 10.5 KW of power and run 80% of the time. The peak load is set to the baseline load + 10.5 KW, as 10.5 KW is being drawn, just not for the full time. After this, the net power production is calculated. If it is positive, the panels are producing more than the house is using, and the batteries can charge. The charging algorithm is explained with Figure 9. If the power production is less than the power use, the batteries are discharging, and the discharging algorithm uses the same concepts as the charging algorithm.

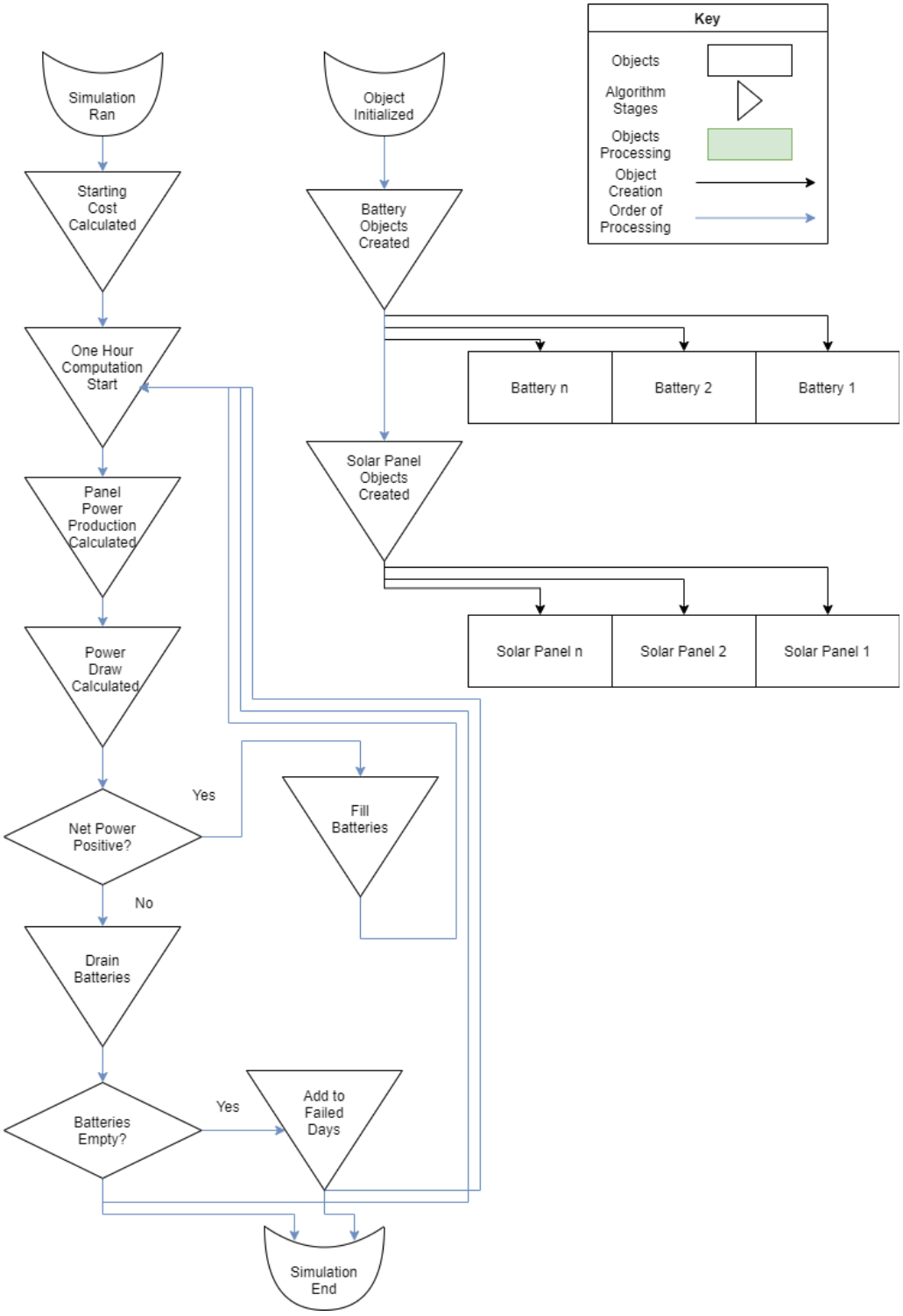


Figure 8. Simulator Diagram.

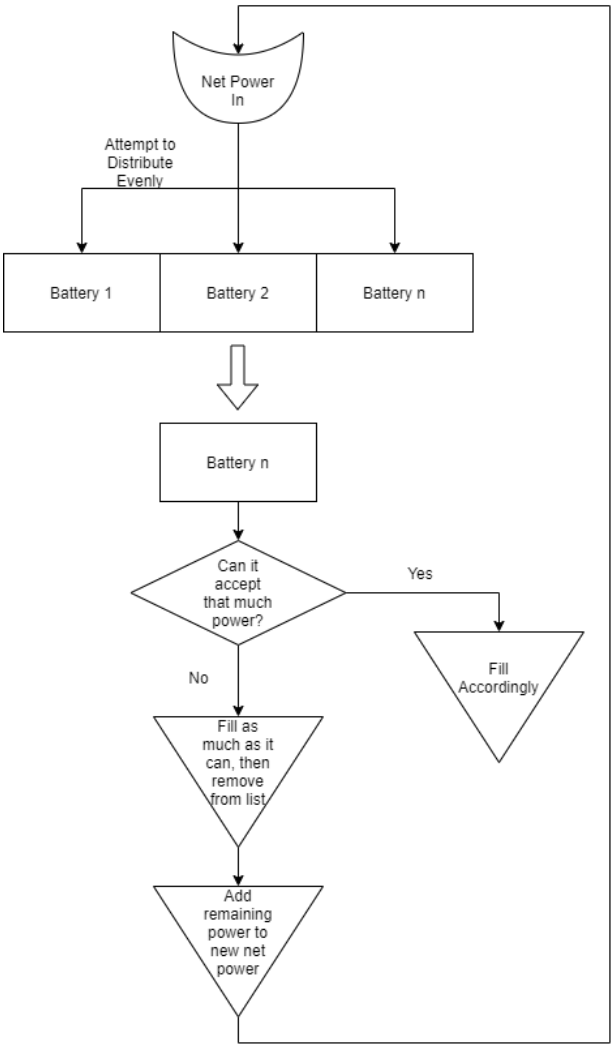


Figure 9. Battery Charging Algorithm

The algorithm used to charge batteries is a simplification of the real-world process. It attempts to distribute all incoming power across all of the batteries evenly. Any time it cannot push that much power into a battery, it performs the same process on the excess, with the batteries that cannot accept any more power removed. If all of the batteries are saturated either in their charge rate or their capacity, the remaining energy is wasted. The same process happens when discharging, but if the batteries run out or cannot provide enough power to meet instantaneous load, a failure is added to the count. This is implemented using recursion within our program.

Result Interpreter

As the simulators finish, each one creates a result that has all of the relevant information about the simulation in it. The result interpreter then looks at the first few simulators that finish with a success result and selects the cheapest one, printing the combination of batteries and the number of solar panels.

Results

Figure 10 shows a sample of results for New York, NY, Bismarck, ND, Miami, FL, and Los Angeles, CA.

Figure 10. A sampling of results.

Interpretation

At first, this chart seems to be erratic, with New York costing over ten times that of Florida to install a system with 90% uptime. However, this is correct, as in Miami and Los Angeles, a heating system is not needed to maintain 90% uptime, and the heating system uses about 10X the electricity as the rest of the house combined at its maximum. This means that this system can be implemented in warmer climates for a much more reasonable price than in colder climates.

Strengths and Weaknesses

Our model had many strengths, but there are a few gaps in our model, some which could be rectified and others that do not work well with our choice of model.

Strengths of our model:

- We had a large amount of user customization for our model. This allowed us to find the best battery combination for a number of different scenarios.
- We used as much real world data as could be found in our model. Specifically for weather data, we used many data points to be able to choose any location in the United States and have a reliable input of how much energy would be available to collect.
- Our model automatically chooses the cheapest possible option. It tests different combinations of batteries, starting with least expensive configurations and moving to more expensive ones, so the first working solution will be the cheapest.

Weaknesses of our model:

- We limited the types of solar panels that were used. The choice of solar panel affects how much electricity is generated and having only chosen one type reduces the customization.
- We also limited the amount of batteries. We used the given batteries as well as 2 supplemental batteries we researched. Our model will return the optimal combination of the seven batteries used, but there may be an alternative option that was not considered.
- We do not check if the location requested is in the United States. We assumed the house is being built in the United States as that was the type currency given and all the data we gathered is in the U.S. In our program we do not check if the coordinates imputed are in the U.S. Therefore it will use the weather conditions closest in distance for data that we have but it may not be accurate.
- The weather conditions in our simulation are predictions based on past data. We do not know if the amount of sunlight to convert to electricity we calculated is accurate.

Concrete Problem

Advantages/Disadvantages:

Disadvantages

- The largest disadvantage for using concrete as a battery is its extremely low capacity. The highest capacity measured in the given article is 0.007kWh/m^2 whereas the lowest of our given batteries is 1.18kWh. One would need approximately 168.57m^2 worth of concrete to match one small battery, as the paper measured battery size in terms of *square* meters.

Amount of concrete to match capacity:

$$0.007kWh/m^2 \bullet x\,m^2 = 1.18kWh$$

$$x = 168.57\,m^2$$

- It takes up a lot of space. 168.57m^2 covers a wide area and that is the smallest amount. To reasonably power a house you would need much more.
- It could be used for multiple purposes which introduces another disadvantage. If being used for something other than a battery it introduces many other variables that could affect the efficiency and other factors of its battery use.

Advantages

- Concrete is widely accessible. If you need more batteries you can easily add more concrete.
- While it physically takes up much more space it can be put into unused space. This unused space gives it the possibility to be multi-purpose such as being part of the structure of a house. It is especially helpful for large buildings like skyscrapers who already use lots of concrete.

Incorporation into a home:

If incorporating cement into our off the grid home, we would use it as a supplement. We would plan when building the house to make part of the structure usable as a battery. It is less efficient to use concrete rather than batteries so we would use as much as we could in the house and then subtract the total capacity of the concrete from the needed capacity to support the house properly. This could be applied to any other house that was not already built but the amount of batteries added to the concrete would change according to the house.

Incorporation into our model:

To incorporate cement into our model there are various unknowns that we would need to find. First we would need to know how much concrete we would need for a 1600 square meter house and use that for the capacity. We would need to calculate the price of altering the concrete. We would need the concrete to build the foundation already so the cost is irrelevant but if the price of altering the concrete exceeds the price of simply having batteries we would default back to not using concrete. We would also find the instantaneous power rating and the continuous power rating. This is another

metric that may cause an outage more often than the batteries. We would also need to adjust our code to account for this new variable but otherwise with these statistics we could incorporate concrete as a battery.

Newspaper Article

If you ever find yourself with a need to create an off-the-grid solar-dependent house, look no further than our battery configuration modelling system. This model was created to address the problem with instituting a fully solar-powered home, isolated from the national power grid. Most applications of solar panels are in urban areas where the houses are still connected to the grid, and they just are boosted by their own solar arrays. These applications pose much less risk of power outages, they are in fact meant to provide a backup or protection for when the national grid is down. In our remote housing situation, solar power would be our only source of electricity. Because of this, batteries are an integral part of the system. Solar batteries would be used to store excess sunlight energy collected during the day, and that power could be drawn at night when there is no direct energy from sunlight. In our model, we look to find the optimal configuration of batteries. “Optimal” in this sense would be the configuration which stores and provides the necessary amount of power for the house. Another aspect of this optimality is the reliability of the system. The efficacy of the model would be measured in terms of its uptime, where if it is able to keep the house running over 90% of the time, including all-electric heating, it is considered a successful configuration. To find these configurations, we looked at multiple battery options to allow for extensive possibilities. Another aspect of our model was the possibility of the user choosing their specific location anywhere in the US. The original problem states that the home was to be built in a remote location, so the model allows for the house to be built anywhere in the US in any remote location the user may desire. Another user input is the number of people living in the house. The number of people in the house affects the amount of electricity and energy needed, therefore the number of batteries needed and their configuration. Our model takes into account many different variables and therefore returns a personalized answer to the user based on their specific inputs. An individual can choose any location in the US to build their house, and choose the number of people that will be living in the house, and the program will give a unique and specific configuration of batteries that should be used for optimal results.

References

Archives. NSRDB. (n.d.). Retrieved November 11, 2021, from https://nsrdb.nrel.gov/data-sets/archives.html

Battery-box h10.0 from BYD (Huizhou) battery: Specs, prices and Reviews. SolarReviews. (n.d.). Retrieved November 11, 2021, from https://www.solarreviews.com/manufacturers/byd-(huizhou)-battery/battery-storage/bHub99945batteryboxh100.

BufferedReader class in Java. Bufferedreader class in Java. (n.d.). Retrieved November 11, 2021, from https://www.tutorialspoint.com/bufferedreader-class-in-java.

Cheap solar panels: 2021 guide to affordable solar panels. Neumeister, K. (2021, October 20). EcoWatch. Retrieved November 11, 2021, from https://www.ecowatch.com/cheap-solar-panels-2655296944.html.

Concrete Network. (2021, July 26). Concrete Prices 2021 - How Much Does Concrete Cost? Retrieved November 12, 2021, from https://www.concretenetwork.com/concrete-prices.html

Distances on Earth 2: The Haversine Formula. Peterson, D. (2021, March 22). The Math Doctors. Retrieved November 11, 2021, from https://www.themathdoctors.org/distances-on-earth-2-the-haversine-formula/.

Electricity usage calculator: Estimate your kwh energy usage. ComparePower. (2020, April 14). Retrieved November 11, 2021, from https://comparepower.com/kwh-electricity-energy-usage-calculator/.

Solar Panel Degradation and The Lifespan of Solar Panels. Gambone, S. (n.d.). Retrieved November 12, 2021, from https://www.paradisesolarenergy.com/blog/solar-panel-degradation-and-the-lifespan-of-solar-panels

Global Climate Data Set 1 – daily temperatures. Black, 3danim8 (aka K). (2017, June 15). Go To DataBlends.us! (3danim8's Blog retired in March 2018). Retrieved November 11, 2021, from https://3danim8.wordpress.com/2017/06/12/global-climate-data-set-1-daily-temperatures/.

Going off-grid in the 2020s: Updated battery choices for today's power needs. Pickerel, K. (2021, October 26). Solar Power World. Retrieved November 11, 2021, from https://www.solarpowerworldonline.com/2021/04/off-grid-batteries-for-todays-power-needs/.

Haversine formula to find distance between two points on a sphere. GeeksforGeeks. (2021, May 12). Retrieved November 11, 2021, from https://www.geeksforgeeks.org/haversine-formula-to-find-distance-between-two-points-on-a-sphere/.

How many solar panels do you need: Panel size and output factors. SunPower. (2021, September 9). SunPower. Retrieved November 11, 2021, from https://us.sunpower.

com/how-many-solar-Panels-do-you-need-panel-size-and-output-factors.

Solar energy. Hurley, S. (2021, October 19). Retrieved November 14, 2021, from https://explainingscience.org/2019/03/09/solar-energy/

Java String contains() Method. Java string contains() method. (n.d.). Retrieved November 11, 2021, from https://www.w3schools.com/java/ref_string_contains.asp#:~:text=Java%20String%20contains%20%28%29%20Method%201%20Definition%20and,found%20%20the%0ava.lang%20package.%204%20Technical%20Details.

Lead acid battery downsides & maintenance. PowerTech Systems - PowerTech Systems. (n.d.). Retrieved November 12, 2021, from https://www.powertechsystems.eu/home/tech-corner/lead-acid-battery-downsides/.

Longi Solar LR4-60HPB-350M 350W mono solar panel. Solaris. (n.d.). Retrieved November 11, 2021, from https://www.solaris-shop.com/longi-solar-lr4-60hpb-350m-350w-mono-solar-panel/.

Multithreading in java - all you need to know about Java multithreading. Great Learning Team. (2021, September 20). GreatLearning Blog: Free Resources what Matters to shape your Career! Retrieved November 13, 2021, from https://www.mygreatlearning.com/blog/multithreading-in-java/.

Silicon Valley Power. (n.d.). Retrieved November 12, 2021, from https://www.siliconvalleypower.com/residents/save-energy/appliance-energy-use-chart

Solar performance and efficiency. Energy.gov. (n.d.). Retrieved November 11, 2021, from https://www.energy.gov/eere/solar/solar-performance-and-efficiency.

The LG Chem Battery: Is it right for your home? Zientara, B. (2020, October 10). Solar Reviews. Retrieved November 11, 2021, from https://www.solarreviews.com/blog/lg-chem-battery-review.

U.S. Energy Information Administration - EIA - independent statistics and analysis. Use of energy in homes - U.S. Energy Information Administration (EIA). (n.d.). Retrieved November 11, 2021, from https://www.eia.gov/energyexplained/use-of-energy/homes.php.

Usa Map with 50 States, Printable, Blank Map, No Text. (2009). Free US and World Maps. Retrieved November 14, 2021, from https://www.freeusandworldmaps.com/html/USAandCanada/USPrintable.html.