|  |  |
| --- | --- |
| **МИНОБРНАУКИ РОССИИ**  федеральное государственное бюджетное образовательное  учреждение высшего образования  **«Национальный исследовательский университет «МЭИ»** | |
| **Институт** | Электроэнергетики |
| **Кафедра** | РЗиАЭ |

**ЛАБОРАТОРНАЯ РАБОТА №3**

по курсу: «Методы решения задач оптимизации»

Динамическое программирование

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(тема работы)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
| Выполнил:  Студент: Груздов А.Г.  (Ф.И.О.)  Группа \_Э-13м-21\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ г. | Проверил:  Руководитель Рогозинников Е.И., Рыжков А.К.  (Ф.И.О.)  Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  «отлично», «хорошо», «удовлетворительно», «неуд.»  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ г. |

Москва 2022 г.

Содержание

[1 Теоретическая часть 3](#_Toc103833784)

[2 Практическая часть 6](#_Toc103833785)

[Заключение 9](#_Toc103833786)

# Теоретическая часть

**Цель работы:** получение практических навыков работы с методом решения задач нелинейного программирования с количеством неизвестных больше трех.

**Исходные данные:**

Таблица – Исходные данные задачи динамического программирования

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Вариант** | **capacity** | **initCharge** | **priceSchedule** | **loadSchedule** | **constan Load** | **targetCharge** |
| 5 | 15000 | 3200 | 1.5, 1.5, 1.5, 1.5,  1.5, 1.5, 2, 3, 5,  5, 5, 4.5, 3, 3, 3,  3, 4.5, 5, 7, 9,  11, 12, 8, 4 | 480, 320, 320, 360, 360,  360,420, 920,1200, 720,  680, 720,800,820,960,  1200, 1380, 1380, 1520,  1800, 1920, 1920, 1640,  1020 | 170 | 2000 |

**Формулировка задачи:**

Персональный энергоблок (ПЭБ) имеет в своём составе аккумуляторную батарею ёмкости 15000 Вт·ч и уровнем заряда 3200 Вт·ч. Цена на электроэнергию в тчении дня изменяется согласно почасовому графику, представленному в таблице 2, нагрузка потребителя, подключенного к ПЭБ, изменяется согласно почасовому графику, представленному в таблице 3, также подключен потребитель с постоянной нагрузкой 170 Вт·ч. ПЭБ способен каждый час либо заряжать свою аккумуляторную батарею (от 1 до 4 кВт·ч), покупая электроэнергию из сети, либо разряжать (от 1 до 4 Вт·ч) – продавая излишки электроэнергии в сеть, либо не производить торговых операцй вовсе. Необходимо спланировать график торговых операций на следующий день, имея перечисленную информацию, на каждый час так, чтобы суммарное вознаграждение к концу дня было максимальным, а оставшийся заряд аккумуляторной батареи был выше значения 2000 Вт·ч.

Таблица 2 – Почасовой график изменения цены на э/э

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 1,2 | 1,4 | 1,5 | 1,6 | 1,7 | 1,8 | 2 | 3 | 4 | 5 | 5 | 4,5 |
| 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| 3 | 3 | 2,7 | 3,2 | 4,5 | 5 | 7 | 9 | 10 | 12 | 8 | 3 |

Таблица 3 – Почасовой график изменения нагрузки потребителя

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 480 | 320 | 320 | 360 | 360 | 360 | 420 | 920 | 1200 | 720 | 680 | 720 |
| 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| 800 | 820 | 960 | 1200 | 1380 | 1380 | 1520 | 1800 | 1920 | 1920 | 1640 | 1020 |

**Динамическое программирование** – способ решения сложных задач путём разбиения их на более простые подзадачи. Задача динамического программирования – это нахождение оптимального решения или стратегии для процесса, который развернут во времени и имеет непрерывно изменяющиеся параметры или ситуации. Ключевая идея в динамическом программировании достаточно проста. Как правило, чтобы решить поставленную задачу, требуется решить отдельные части задачи (подзадачи), после чего объединить решения подзадач в одно общее решение. Часто многие из этих подзадач одинаковы. Подход динамического программирования состоит в том, чтобы решить каждую подзадачу только один раз, сократив тем самым количество вычислений. Это особенно полезно в случаях, когда число повторяющихся подзадач экспоненциально велико.

**Метод динамического программирования сверху** — это простое запоминание результатов решения тех подзадач, которые могут повторно встретиться в дальнейшем. **Динамическое программирование снизу** включает в себя переформулирование сложной задачи в виде рекурсивной последовательности более простых подзадач.

Можно сказать, что динамическое программирование пользуется следующими **свойствами задачи**:

- перекрывающиеся подзадачи – означают подзадачи, которые используются для решения некоторого количества задач (не одной) большего размера (то есть мы несколько раз проделываем одно и то же). Ярким примером является вычисление последовательности Фибоначчи;

- оптимальная подструктура – означает, что оптимальное решение подзадач меньшего размера может быть использовано для решения исходной задачи;

- возможность запоминания решения часто встречающихся подзадач.

# Практическая часть

**Ход работы:**

Метод применим к задачам с оптимальной подструктурой, выглядящим как набор перекрывающихся подзадач, сложность которых чуть меньше исходной.

Для решения данной задачи использовалось такое понятие, как хэш-таблица. **Хеш-таблица** — это структура данных, реализующая интерфейс ассоциативного массива, а именно, она позволяет хранить пары (ключ, значение) и выполнять три операции: операцию добавления новой пары, операцию поиска и операцию удаления пары по ключу. Для лучшего понимания представлена хеш-таблица для первого часа.

Таблица 4 – Хеш-таблица для первого часа

|  |  |  |
| --- | --- | --- |
| **xi** | **f1(xi)** | **g1(xi)** |
| -4000 | 6 | -1450 |
| -3000 | 4,5 | -450 |
| -2000 | 3 | 550 |
| -1000 | 1,5 | 1550 |
| 0 | 0 | 2550 |
| 1000 | -1,5 | 3550 |
| 2000 | -3 | 4550 |
| 3000 | -4,5 | 5550 |
| 4000 | -6 | 6550 |

Где xi в таблице – значения торговых операций, f1(xi) – значения целевой функции (цены), получаемые на данный час, g1(xi) – значения остаточного заряда на батарее.

Значение целевой функции в общем виде будет рассчитываться по следующей формуле:

,

где n – номер часа;

p – стоимость одного кВт⋅ч электроэнергии (priceSchedul);

Расчет значения остаточного заряда батареи в общем виде:

,

где с – постоянная нагрузка (constantLoad);

t – переменная нагрузка (loadSchedule);

Решение данной задачи можно описать как простой перебор, но с фильтрацией возможных вариантов на каждой из итераций перебора.

Перебор осуществляется с учётом следующих правил:

* Торговая операция может быть использована в качестве варианта, если заряд аккумуляторной батареи после применения данной операции не превышает значение максимальной ёмкости АБ;
* Торговая операция может быть использована в качестве варианта, если заряд аккумуляторной батареи после применения данной операции не меньше значения нагрузки следующего часа.

Очевидно, что метод стандартного перебора не подходит для данной задачи, из-за огромного числа всевозможных вариантов (максимум 924 вариантов). Поэтому на каждом шаге применяется инструмент фильтрации полученных вариантов.

Фильтрация осуществляется с учётом следующего правила:

Если величина ёмкости АБ на конец часа одного варианта не превышает величину ёмкости АБ другого варианта при этом значение функции прибыли первого варианта меньше, либо равно значению функции прибыли второго варианта, то первый вариант можно отбросить, поскольку он заведомо является хуже второго.

Пример такой фильтрации приведён ниже. На таблице 5 представлены все допустимые варианты торговых операций для седьмого часа. Среди них выделены красным и жёлтым цветом варианты, которые были отфильтрованы. Желтым и зелёным с ёмкостью АБ на конец часа равной 7300 Вт·ч. Таких вариантов всего 8 штук. Не выделенные и выделены зелёным цветом представлены все допустимые варианты после фильтрации. Из 8 исходных вариантов был выбран 1 единственный с наибольшим значением функции прибыли.

Таблица 5 – Количество вариантов торговых операций для седьмой часа с фильтрацией и без

|  |
| --- |
| hashTable['h7;8300;4000,4000,-4000,-4000,4000,4000,1000,1000'] = {float} -17000.0 |
| hashTable['h7;9300;4000,4000,-4000,-4000,4000,4000,1000,2000'] = {float} -20000.0 |
| hashTable['h7;10300;4000,4000,-4000,-4000,4000,4000,1000,3000'] = {float} -23000.0 |
| hashTable['h7;11300;4000,4000,-4000,-4000,4000,4000,1000,4000'] = {float} -26000.0 |
| hashTable['h7;6300;4000,4000,-4000,-4000,4000,4000,1000,-1000'] = {float} -11000.0 |
| hashTable['h7;5300;4000,4000,-4000,-4000,4000,4000,1000,-2000'] = {float} -8000.0 |
| hashTable['h7;4300;4000,4000,-4000,-4000,4000,4000,1000,-3000'] = {float} -5000.0 |
| hashTable['h7;3300;4000,4000,-4000,-4000,4000,4000,1000,-4000'] = {float} -2000.0 |
| hashTable['h7;7300;4000,4000,-4000,-4000,4000,4000,1000,0'] = {float} -14000.0 |
| hashTable['h7;9300;4000,4000,-4000,-4000,4000,4000,2000,1000'] = {float} -19000.0 |
| hashTable['h7;10300;4000,4000,-4000,-4000,4000,4000,2000,2000'] = {float} -22000.0 |
| hashTable['h7;11300;4000,4000,-4000,-4000,4000,4000,2000,3000'] = {float} -25000.0 |
| hashTable['h7;12300;4000,4000,-4000,-4000,4000,4000,2000,4000'] = {float} -28000.0 |
| hashTable['h7;7300;4000,4000,-4000,-4000,4000,4000,2000,-1000'] = {float} -13000.0 |
| hashTable['h7;6300;4000,4000,-4000,-4000,4000,4000,2000,-2000'] = {float} -10000.0 |
| hashTable['h7;5300;4000,4000,-4000,-4000,4000,4000,2000,-3000'] = {float} -7000.0 |
| hashTable['h7;4300;4000,4000,-4000,-4000,4000,4000,2000,-4000'] = {float} -4000.0 |
| hashTable['h7;8300;4000,4000,-4000,-4000,4000,4000,2000,0'] = {float} -16000.0 |
| hashTable['h7;10300;4000,4000,-4000,-4000,4000,4000,3000,1000'] = {float} -21000.0 |
| hashTable['h7;11300;4000,4000,-4000,-4000,4000,4000,3000,2000'] = {float} -24000.0 |
| hashTable['h7;12300;4000,4000,-4000,-4000,4000,4000,3000,3000'] = {float} -27000.0 |
| hashTable['h7;13300;4000,4000,-4000,-4000,4000,4000,3000,4000'] = {float} -30000.0 |
| hashTable['h7;8300;4000,4000,-4000,-4000,4000,4000,3000,-1000'] = {float} -15000.0 |
| hashTable['h7;7300;4000,4000,-4000,-4000,4000,4000,3000,-2000'] = {float} -12000.0 |
| hashTable['h7;6300;4000,4000,-4000,-4000,4000,4000,3000,-3000'] = {float} -9000.0 |
| hashTable['h7;5300;4000,4000,-4000,-4000,4000,4000,3000,-4000'] = {float} -6000.0 |
| hashTable['h7;9300;4000,4000,-4000,-4000,4000,4000,3000,0'] = {float} -18000.0 |
| hashTable['h7;11300;4000,4000,-4000,-4000,4000,4000,4000,1000'] = {float} -23000.0 |
| hashTable['h7;12300;4000,4000,-4000,-4000,4000,4000,4000,2000'] = {float} -26000.0 |
| hashTable['h7;13300;4000,4000,-4000,-4000,4000,4000,4000,3000'] = {float} -29000.0 |
| hashTable['h7;14300;4000,4000,-4000,-4000,4000,4000,4000,4000'] = {float} -32000.0 |
| hashTable['h7;9300;4000,4000,-4000,-4000,4000,4000,4000,-1000'] = {float} -17000.0 |
| hashTable['h7;8300;4000,4000,-4000,-4000,4000,4000,4000,-2000'] = {float} -14000.0 |
| hashTable['h7;7300;4000,4000,-4000,-4000,4000,4000,4000,-3000'] = {float} -11000.0 |
| hashTable['h7;6300;4000,4000,-4000,-4000,4000,4000,4000,-4000'] = {float} -8000.0 |
| hashTable['h7;10300;4000,4000,-4000,-4000,4000,4000,4000,0'] = {float} -20000.0 |
| hashTable['h7;6300;4000,4000,-4000,-4000,4000,4000,-1000,1000'] = {float} -13000.0 |
| hashTable['h7;7300;4000,4000,-4000,-4000,4000,4000,-1000,2000'] = {float} -16000.0 |
| hashTable['h7;8300;4000,4000,-4000,-4000,4000,4000,-1000,3000'] = {float} -19000.0 |
| hashTable['h7;9300;4000,4000,-4000,-4000,4000,4000,-1000,4000'] = {float} -22000.0 |
| hashTable['h7;4300;4000,4000,-4000,-4000,4000,4000,-1000,-1000'] = {float} -7000.0 |
| hashTable['h7;3300;4000,4000,-4000,-4000,4000,4000,-1000,-2000'] = {float} -4000.0 |
| hashTable['h7;2300;4000,4000,-4000,-4000,4000,4000,-1000,-3000'] = {float} -1000.0 |
| hashTable['h7;5300;4000,4000,-4000,-4000,4000,4000,-1000,0'] = {float} -10000.0 |
| hashTable['h7;5300;4000,4000,-4000,-4000,4000,4000,-2000,1000'] = {float} -11000.0 |
| hashTable['h7;6300;4000,4000,-4000,-4000,4000,4000,-2000,2000'] = {float} -14000.0 |
| hashTable['h7;7300;4000,4000,-4000,-4000,4000,4000,-2000,3000'] = {float} -17000.0 |
| hashTable['h7;8300;4000,4000,-4000,-4000,4000,4000,-2000,4000'] = {float} -20000.0 |
| hashTable['h7;3300;4000,4000,-4000,-4000,4000,4000,-2000,-1000'] = {float} -5000.0 |
| hashTable['h7;2300;4000,4000,-4000,-4000,4000,4000,-2000,-2000'] = {float} -2000.0 |
| hashTable['h7;4300;4000,4000,-4000,-4000,4000,4000,-2000,0'] = {float} -8000.0 |
| hashTable['h7;4300;4000,4000,-4000,-4000,4000,4000,-3000,1000'] = {float} -9000.0 |
| hashTable['h7;5300;4000,4000,-4000,-4000,4000,4000,-3000,2000'] = {float} -12000.0 |
| hashTable['h7;6300;4000,4000,-4000,-4000,4000,4000,-3000,3000'] = {float} -15000.0 |
| hashTable['h7;7300;4000,4000,-4000,-4000,4000,4000,-3000,4000'] = {float} -18000.0 |
| hashTable['h7;2300;4000,4000,-4000,-4000,4000,4000,-3000,-1000'] = {float} -3000.0 |
| hashTable['h7;3300;4000,4000,-4000,-4000,4000,4000,-3000,0'] = {float} -6000.0 |
| hashTable['h7;3300;4000,4000,-4000,-4000,4000,4000,-4000,1000'] = {float} -7000.0 |
| hashTable['h7;4300;4000,4000,-4000,-4000,4000,4000,-4000,2000'] = {float} -10000.0 |
| hashTable['h7;5300;4000,4000,-4000,-4000,4000,4000,-4000,3000'] = {float} -13000.0 |
| hashTable['h7;6300;4000,4000,-4000,-4000,4000,4000,-4000,4000'] = {float} -16000.0 |
| hashTable['h7;2300;4000,4000,-4000,-4000,4000,4000,-4000,0'] = {float} -4000.0 |
| hashTable['h7;7300;4000,4000,-4000,-4000,4000,4000,0,1000'] = {float} -15000.0 |
| hashTable['h7;8300;4000,4000,-4000,-4000,4000,4000,0,2000'] = {float} -18000.0 |
| hashTable['h7;9300;4000,4000,-4000,-4000,4000,4000,0,3000'] = {float} -21000.0 |
| hashTable['h7;10300;4000,4000,-4000,-4000,4000,4000,0,4000'] = {float} -24000.0 |
| hashTable['h7;5300;4000,4000,-4000,-4000,4000,4000,0,-1000'] = {float} -9000.0 |
| hashTable['h7;4300;4000,4000,-4000,-4000,4000,4000,0,-2000'] = {float} -6000.0 |
| hashTable['h7;3300;4000,4000,-4000,-4000,4000,4000,0,-3000'] = {float} -3000.0 |
| hashTable['h7;2300;4000,4000,-4000,-4000,4000,4000,0,-4000'] = {float} 0.0 |

Результат работы алгоритма представлен на рисунке 2 “Заряд батареи с учётом торговых операций”. Можно отметить, что результат работы можно объяснить логически следующим образом: в часы максимальной цены на э/э необходимо продавать электроэнергию в максимально возможном объёме, а в часы минимальных цен на э/э необходимо закупать максимально возможное количество э/э.

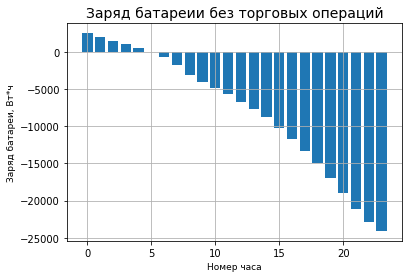


Рисунок – Гистограммы заряда батареи без торговых операций

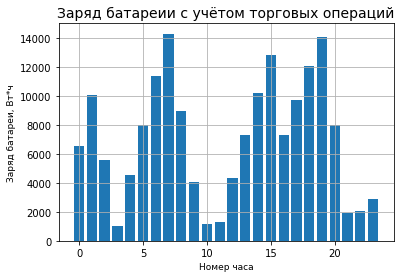


Рисунок – Гистограммы заряда батареи с учётом торговых операций

На рисунке 3 представлен почасовой график функции прибыли. По итогам 24 часов значение функции прибыли равно -100500.0 у.е.

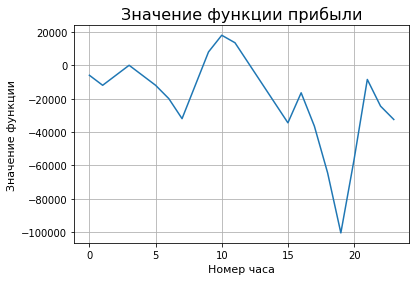


Рисунок – Значение функции прибыли

Заключение

По графику видно, что без применения торговых операций заряд батареи в течение дня просто распродастся меньше чем за 5 часов, что не является оптимальным решением в реальной энергосистеме. С применением торговых операций возможно спланировать такой график на следующий день, при котором значение целевой функции будет максимально возможным, а значение уровня заряда батареи не будет меньше нуля.