

## Динамическое программирование для эффективного обучения: максимизация результатов через оптимальное распределение времени

Данил Александрович Гордеев<sup>✉</sup>

Московский финансово-юридический университет МФЮА, Москва, Россия

<sup>✉</sup>[dan.gor220@yandex.ru](mailto:dan.gor220@yandex.ru)

**Аннотация.** В статье представлен метод динамического программирования, позволяющий решить задачу оптимального распределения времени между занятиями с целью максимизации уровня освоения материала. При этом предполагается, что уровень освоения занятия пропорционален затраченному времени на него, и скорость освоения следующего занятия обратно пропорциональна уровню освоения предыдущего. Сформулирована постановка задачи, описана математическая модель динамического программирования, определено рекуррентное соотношение Беллмана и представлен программный код модели динамического программирования, реализующий функции условной и безусловной оптимизации. Последовательный вызов функций при заданных аргументах позволил получить управляющие последовательности, переводящие управляемую систему в состояние, при котором результат управления максимален.

**Ключевые слова и фразы:** динамическое программирование, принцип оптимальности, оптимизация, программный код модели динамического программирования, уравнение Беллмана

Для цитирования: Гордеев Д. А. Динамическое программирование для эффективного обучения: максимизация результатов через оптимальное распределение времени // рукопись направлена в журнал *Программные системы: теория и приложения* 07.11.2023

## Введение

Смысловой аспект информации, отражающий соотношение формы сообщения и содержания, образует семантическую информацию. Основными свойствами семантической информации являются тиражируемость и невзаимозаменяемость [1, 2]. Методика освоения обучающимися семантической информации напрямую связана с этими свойствами. Тиражируемость подразумевает, что семантическая информация может использоваться многократно для передачи одного и того же значения. В обучении это означает, что информация может быть изучена несколько раз в различных контекстах, что обеспечит лучшее понимание материала. Невзаимозаменяемость информации означает, что значение не может быть заменено другим аналогичным, что в обучении соответствует невозможности замены ключевых терминов и понятий.

Качество освоения обучающимися информации во многом зависит от времени, выделенного на получение знания. Эффективное распределение этого времени позволяет максимизировать результат от обучения. Подобное распределение является задачей оптимизации, решение которой возможно при помощи метода динамического программирования. Основная идея заключается в том, что задачи могут быть разбиты на более мелкие подзадачи, а ответ на каждую подзадачу может быть сохранен, чтобы использовать его для решения более крупных задач. Такие операции называются многошаговыми. На каждом шаге оптимизируется управление только этого шага. Вместе с тем, на каждом шаге управление выбирается с учетом последствий [3]. В основе метода динамического программирования лежит принцип оптимальности, сформулированный Ричардом Беллманом, согласно которому, каково бы ни было начальное состояние на любом шаге и управление, выбранное на этом шаге, последующие управления должны выбираться оптимальными относительно состояния, к которому придет система в конце данного шага [4].

Статья состоит из двух разделов. Первый раздел включает в себя постановку задачи динамического программирования, определение условий и ограничений, целевой функции, уравнений состояния и уравнения Беллмана, а также описание этапов условной и безусловной оптимизации. Во втором разделе представлен программный код модели динамического программирования, который определяет функции условной и безусловной оптимизации, а также демонстрирует пример работы алгоритма с заданными параметрами.

## 1. Постановка задачи

Задача динамического программирования для максимизации результатов обучения заключается в распределении времени  $T_0$  между  $n$  занятиями  $L_1, L_2, \dots, L_n$ . Выделенное занятию  $L_n$  время  $t_k$  гарантирует уровень освоения  $f_k(t_k)$  ( $k = 1, 2, \dots, n$ ). При этом предполагается, что:

- (1) Уровень освоения, полученный на разных занятиях, выражается в одинаковых единицах;
- (2) Общий уровень освоения равен сумме уровней освоения, полученных от распределения всего времени по всем занятиям;
- (3) Уровень освоения занятия пропорционален затраченному времени на него;
- (4) Скорость освоения следующего занятия обратно пропорциональна уровню освоения предыдущего.

Общий уровень освоения выразим целевой функцией, имеющей аддитивную форму:

$$(1) \quad Z = \sum_{k=1}^n f_k(t_k)$$

Переменные  $t_k$  должны удовлетворять условиям:

$$(2) \quad (t_1 + t_2 + \dots + t_n) - Q = T_0; \quad t_k > 0 \quad (k = 1, \dots, n),$$

где  $Q$  – дополнительное время, полученное в соответствии с условием 4, согласно которому более высокий уровень освоения текущего занятия уменьшает время, необходимое для освоения следующего. Получение дополнительного времени представлено следующей формулой:

$$(3) \quad Q = (t_k - t_{min}) * q; \quad Q < 0 \Rightarrow Q = 0,$$

где  $t_{min}$  – время освоения занятия на минимальном уровне, а  $q$  – заданный коэффициент прироста времени.

Требуется определить переменные  $t_1, t_2, \dots, t_n$ , которые удовлетворяют ограничениям 2 и обращают в максимум целевую функцию 1.

Распределение времени между  $n$  занятиями можно интерпретировать как  $n$ -шаговый процесс. За номер  $k$ -го шага примем номер занятия, которому выделяется время  $t_k$ . На первом шаге выделяется время  $t_1$ , для первого занятия, на втором шаге – время  $t_2$  для второго занятия, и так далее. Переменные  $t_k$  ( $k = 1, \dots, n$ ) можно рассматривать как управляющие переменные.

Начальное состояние системы характеризуется величиной  $T_0$  времени, подлежащего распределению. После выделения  $t_1$  остается  $T_1 = T_0 - t_1 + Q$  времени и так далее. Величины  $T_0, T_1, \dots, T_n$ , характеризующие остаток времени после распределения на предшествующих шагах,

будем рассматривать как параметры состояния. Уравнениями состояния служат равенства:

$$(4) \quad T_k = T_{k-1} - t_k + Q \quad (k = 1, \dots, n)$$

Максимальный уровень освоения зависит от того, сколько времени осталось от предыдущих  $k - 1$  шагов, то есть от величины  $T_{k-1}$ . Обозначим его через  $Z_k(T_{k-1})$ . Таким образом,  $Z_1(T_0) = Z_{max}$ , то есть  $Z_1(T_0)$  представляет суммарный максимальный уровень освоения за  $n$  шагов (уровень освоения, полученный при оптимальном распределении времени  $T_0$  между  $n$  занятиями).

Допустимое значение  $t_k$  можно выбирать из условия  $0 < t_k \leq T_{k-1}$ . Принцип оптимальности в данном случае означает, что, выделив величину  $t_k$  и получив от  $k$ -го занятия уровень освоения  $f_k(t_k)$ , следует распределить оставшееся время  $T_k = T_{k-1} - t_k + Q$  самым выгодным образом и получить от занятий  $L_{k+1}, \dots, L_n$  максимальный уровень освоения  $Z_{k+1}(T_k)$ . Если оставшегося времени  $T_k$  недостаточно для освоения занятия на более высоком уровне, то время округляется в меньшую сторону, и выбирается подходящий уровень освоения для данного значения времени. Величину  $t_k$  следует определять из условия максимизации суммы  $f_k(t_k) + Z_{k+1}(T_k)$ . Таким образом получаем уравнение Беллмана:

$$(5) \quad Z_k(T_{k-1}) = \max_{0 < t_k \leq T_{k-1}} \{f_k(t_k) + Z_{k+1}(T_k)\}$$

После выполнения всех вычислений заканчивается основной этап условной оптимизации и начинается этап безусловной оптимизации. На данном этапе, зная функцию  $Z_1(T_0)$ , по заданному значению  $T_0$  определяем  $Z_{max} = Z_1(T_0)$ . Далее, обращаемся к последовательности  $t_k(T_{k-1})$ , которую проходим от начала к концу процесса. Выделяем  $t_1 = t_1(T_0)$  первому занятию; тогда для распределения остается  $T_1 = T_0 - t_1 + Q$ . По этой величине определяем оптимальное количество времени  $t_2 = t_2(T_1)$ , выделяемых второму занятию. Снова находим  $T_2 = T_1 - t_2 + Q$ , после чего определяем  $t_3$ , и так далее, пока не будет определено искомое оптимальное управление  $(t_1, t_2, \dots, t_n)$ .

## 2. Программный код

Модель динамического программирования может быть представлена в виде программного кода, который реализует функции *conditional Optimization* и *unconditionalOptimization*, предназначенные для условной и безусловной оптимизации. Функция условной оптимизации принимает входные параметры в виде двумерного массива *data* и коэффициента прироста времени  $q$ . Первый элемент массива *data* представляет собой

массив, содержащий время, которое должно быть распределено на каждом шаге. Остальные элементы представляют собой массивы данных с различными уровнями освоения занятий в зависимости от заданного времени.

Основная задача функции условной оптимизации заключается в поиске условных максимумов для каждого шага на основе предоставленных данных и условий. Это достигается путем итераций для каждого шага и оценки условного максимума для каждого временного значения, учитывая условия на предыдущих шагах. Результатом работы функции является объект *conditionalMaximums*, содержащий информацию о условных максимумах на каждом шаге процесса. Эти данные используются при безусловной оптимизации. Программный код функции *conditionalOptimization* представлен в листинге 1:

```
1 function conditionalOptimization(data, q){
2   // Объект хранящий данные с условными максимумами.
3   const conditionalMaximums = {};
4   /* Обход каждого шага управляемого процесса, начиная с
5   последнего. */
6   for (let k = data.length - 1; k > 0; k--){
7     // Создание объекта условного максимума на k-м шаге.
8     conditionalMaximums['Z_${k}'] = {};
9     /* Обход массива с данными о времени, подлежащего
10    распределению. */
11    for (const T of data[0]){
12      // Получение допустимых значений времени.
13      for (let t = T; t > 0; t--){
14        // Получение уровня освоения на текущем шаге.
15        const currLevel = data[k][t - 1];
16        /* Если уровень освоения равен нулю, пропустить
17        итерацию. Необходимо освоить каждое занятие. */
18        if (currLevel === 0) continue;
19        // Если существует условный максимум на предыдущем шаге,
20        if (conditionalMaximums['Z_${k + 1}']) {
21          /* получить доступное время для распределения,
22          учитывая условие более высокого уровня освоения при
23          заданном t. */
24          let T_k = T - t + Math.max((t - data[0].at(0))*q, 0);
25          /* Если при заданном времени T_k не существует
26          условного максимума, округлить до целого числа в
27          меньшую сторону. */
28          if (!conditionalMaximums['Z_${k + 1}'][T_k]){
29            T_k = Math.floor(T_k)}
30          /* T_k не может превышать общее время
31          подлежащее распределению. */
32          T_k = Math.min(T_k, data[0].at(-1));
33          // Если занятие не освоено, пропустить итерацию.
```

```

34     if (T_k == 0 || t == 0 || !conditionalMaximums['Z_{k
35         + 1}']['T_k']) continue;
36     // Получение условного максимума.
37     const Z_max = currLevel + conditionalMaximums['Z_{k
38         + 1}']['T_k'][0];
39     /* Если в объекте нет условного максимума или
40     предыдущий условный максимум меньше текущего, */
41     if (!conditionalMaximums['Z_{k}']['T'] ||
42         conditionalMaximums['Z_{k}']['T'][0] < Z_max) {
43         // записать условный максимум в объект.
44         conditionalMaximums['Z_{k}']['T'] = [Z_max, t]}
45     /* Если условный максимум на предыдущем шаге
46     отсутствует, */
47     } else {
48     // записать условный максимум в объект.
49     conditionalMaximums['Z_{k}']['t'] = [currLevel, t];
50     }
51     }
52 }
53 }
54 // Вернуть объект с условными максимумами.
55 return conditionalMaximums;
56 }

```

Листинг 1. Программный код функции условной оптимизации

Функция безусловной оптимизации принимает в качестве параметров объект с условными максимумами *conditionalMaximums*, массив с данными *data* и коэффициент прироста времени *q*. Основная задача функции заключается в итеративном переборе объекта с условными максимумами, начиная с  $Z_1(T_0)$  и заканчивая последним шагом, чтобы определить наилучшее распределение времени для каждой задачи. Программный код функции *unconditionalOptimization* представлен в листинге 2:

```

1 function unconditionalOptimization(conditionalMaximums, data, q){
2     // Объявление массива с данными о распределении времени.
3     const distribution = [];
4     // Объявление переменной с параметром состояния.
5     let T = data[0].at(-1);
6     // Проход по каждому шару объекта conditionalMaximums.
7     for (let i=1; i<= Object.keys(conditionalMaximums).length; i++){
8         // Объявление переменной, для хранения лучшего времени.
9         let time;
10        /* Если параметр состояния превышает общее время для
11        распределения T_0, */
12        if (T > data[0].at(-1)) {
13            // то лучшее время доступно при T_0.
14            time = conditionalMaximums['Z_{i}']['data[0].at(-1)'][1]

```

```

15 // Если параметр состояния на данном шаге отсутствует,
16 } else if (!conditionalMaximums['Z_{i}'][T]) {
17     /* тогда округлить параметр состояния до целого и
18     присвоить переменной времени.*/
19     time = conditionalMaximums['Z_{i}'][Math.floor(T)][1]
20 } else {
21     // Лучшее время доступно при T.
22     time = conditionalMaximums['Z_{i}'][T][1]
23 }
24 // Получение следующего параметра состояния.
25 T = T - time + Math.max((time - data[0].at(0)) * q, 0);
26 // Добавление результата в массив.
27 distribution.push('t_{i}: ${time}');
28 }
29 // Вернуть массив с данными о распределении.
30 return distribution;
31 }

```

Листинг 2. Программный код функции безусловной оптимизации

Применение функций оптимизации позволяет определить искомое оптимальное управление  $(t_1, t_2, \dots, t_n)$  при заданных данных. Продемонстрируем их работу на примере.

Требуется распределить время  $T_0 = 5$  между 4 занятиями при коэффициенте прироста времени  $q$  равном 0 и 0.5. Дан массив *data* с произвольными данными, первый элемент которого - подмассив с временем, подлежащим распределению, остальные элементы - подмассивы занятий, содержащие уровни освоения при заданном времени. Передадим параметры функциям условной и безусловной оптимизации и выведем результат в консоль. Определение параметров функций представлено в листинге 3:

```

1  const data = [
2    [1, 2, 3, 4, 5], // Время подлежащее распределению
3    [9, 10, 13, 15, 18], // Уровни освоения занятия 1
4    [6, 8, 9, 11, 16], // Уровни освоения занятия 2
5    [4, 5, 10, 13, 15], // Уровни освоения занятия 3
6    [7, 9, 11, 14, 15] // Уровни освоения занятия 4
7  ]
8
9  // Вызов функций условной и безусловной оптимизации при q = 0
10 console.log(unconditionalOptimization(
11   conditionalOptimization(data, 0), data, 0));
12
13 // Вызов функций условной и безусловной оптимизации при q = 0.5
14 console.log(unconditionalOptimization(
15   conditionalOptimization(data, 0.5), data, 0.5));

```

Листинг 3. Определение параметров функций оптимизации

Результатом первого вызова функций будет массив, представляющий распределение времени, при котором достигается максимальный уровень освоения, равный 28. Этот результат достигается при следующем распределении времени между занятиями: первому занятию выделяется  $t_1 = 1$ , второму  $t_2 = 2$ , третьему  $t_3 = 1$  и четвертому  $t_4 = 1$ . Данный вызов, при  $q = 0$ , демонстрирует результат оптимизации, который не зависит от коэффициента прироста времени. Таким образом, не выполняется условие, согласно которому скорость освоения следующего занятия обратно пропорциональна уровню освоения предыдущего.

Следующий вызов функций принимает коэффициент времени  $q$  равный 0.5. В результате вызова максимальный уровень освоения равен 32, а значения распределения времени будут следующими:  $t_1 = 1$ ,  $t_2 = 1$ ,  $t_3 = 3$ ,  $t_4 = 1$ . Поскольку на третье занятие выделяется 3 единицы времени, то уровень освоения выше в два раза. В то же время, скорость освоения последующего занятия уменьшается пропорционально. Следовательно, учитывая коэффициент прироста времени, на четвертое занятие выделяется дополнительная единица времени.

Таким образом, представленный алгоритм позволяет находить необходимые управляющие переменные, которые приводят управляемую систему в состояние, при котором результат управления максимален, учитывая заданные условия.

## Заключение

Метод динамического программирования представляет собой мощный инструмент в решении задач оптимизации. Важно понимать, что не существует универсального алгоритма, который мог бы решить любую задачу оптимизации. Вместо этого, динамическое программирование предоставляет общий подход, который позволяет строить индивидуальные модели для каждой конкретной задачи в зависимости от её условий, размерности и других характеристик.

Одной из основных сложностей применения метода динамического программирования является вычислительный процесс первого, условного этапа оптимизации. При увеличении размера задачи увеличивается и объем необходимых вычислений, что влечет за собой дополнительные временные затраты. Для решения этой проблемы, эффективным шагом является перевод модели динамического программирования в программный код, что позволяет значительно сократить время вычислительного процесса.




Представленный в статье алгоритм реализует функции, соответствующие этапам условной и безусловной оптимизации. В зависимости от



конкретных условий задачи, он способен определять оптимальные управляющие последовательности. Это открывает новые перспективы для разработки эффективных образовательных траекторий обучения, способствуя значительному улучшению освоения учебного материала.

Применение данного алгоритма ограничено условием, согласно которому скорость освоения последующего занятия обратно пропорциональна уровню освоения предыдущего. Это ограничение связано с необходимостью определения коэффициента прироста времени, который зависит от конкретных временных рамок задачи. Неправильно подобранный коэффициент, отличный от нуля, может привести к неточным результатам. В связи с этим планируется дальнейшее совершенствование алгоритма и изучение особенностей и условий, при которых можно точно определить такой коэффициент прироста времени, который будет адаптироваться к изменяющимся условиям и временным рамкам. Это позволит более эффективно использовать метод динамического программирования для максимизации результатов обучения при разнообразных входных данных.

### Список литературы

- [1] *Вопросы теоретической и прикладной информатики: учеб. пособие*, В. Б. Голубкова, А. И. Брагинский. – М.: Изд-во МАДИ. – 2019. – 72 с.  <sup>↑2</sup>
- [2] Черняк Т. А., Удахина С. В., Косухина М. А. *Информационное обеспечение субъектов экономической деятельности: Базы данных и знаний*. – СПб.: Изд-во Санкт-Петербургского университета управления и экономики. – 2015. – 200 с.  <sup>↑2</sup>
- [3] Карпов Д. А., Струченков В. И. *Динамическое программирование в прикладных задачах, допускающих сокращение перебора вариантов* // Российский технологический журнал. – 2020. – № 8(4). – С. 96-111. DOI <sup>↑2</sup>
- [4] *Математическое программирование: теория и методы : учебное пособие*, Сост. Н. В. Гредасова, А. Н. Сесекин, А. Ф. Шориков, М. А. Плескунов; научный редактор В. И. Зенков. – Екатеринбург: Изд-во Уральского университета. – 2020. – 200 с.  <sup>↑2</sup>

**Информация об авторе:**

Данил Александрович Гордеев

Аспирант Московского финансово-юридического университета МФЮА. Научные интересы связаны с проектированием и программированием пользовательских интерфейсов.

iD 0009-0006-7237-5815

**e-mail:** [dan.gor220@yandex.ru](mailto:dan.gor220@yandex.ru)

*Автор заявляет об отсутствии конфликта интересов.*

## Dynamic programming for efficient learning: maximizing results through optimal time allocation

Danil Alexandrovich **Gordeev**

Moscow University of Finance and Law (MFUA), Moscow, Russia

 [dan.gor220@yandex.ru](mailto:dan.gor220@yandex.ru)

**Abstract.** The article presents a dynamic programming method for solving the problem of optimal time allocation between study sessions with the aim of maximizing the level of material mastery. It is assumed that the level of mastery of a study session is proportional to the time spent on it, and the rate of mastering the next session is inversely proportional to the level of mastery of the previous one. The problem statement is formulated, a mathematical model of dynamic programming is described, the Bellman recurrence relation is defined, and a program code for the dynamic programming model is provided, implementing functions for both conditional and unconditional optimization. Sequentially calling these functions with specified arguments allowed us to obtain control sequences that bring the controlled system to a state where the control result is maximized. (*In Russian*).

**Key words and phrases:** dynamic programming, principle of optimality, optimization, dynamic programming model code, Bellman equation

2020 *Mathematics Subject Classification:* 90C39; 49L20

**For citation:** Danil A. Gordeev. *Dynamic programming for efficient learning: maximizing results through optimal time allocation*. Submitted for the journal *Program Systems: Theory and Applications* at 07.11.2023 (*In Russ.*).

## References

- [1] *Questions of theoretical and applied computer science*, V.B. Golubkova, A.I. Braginsky, MADI, M., 2019 (in Russian), 72 pp. [URL](#)
- [2] Chernyak T.A., Udakhina S.V., Kosukhina M.A.. *Information support of economic entities: Databases and knowledge*, Izd-vo Saint-Petersburg University of Management Technologies and Economics, SPb., 2015 (in Russian), 200 pp. [URL](#)
- [3] Karpov D.A., Struchenkov V.I.. “Dynamic programming in applied tasks which are allowing to reduce the options selection”, *Russian Technological Journal*, 2020, no. 8(4), pp. 96-111 (in Russian). DOI
- [4] *Mathematical programming: theory and methods: textbook*, Comp. N.V. Gredasova, A.N. Sesekin, A.F. Shorikov, M.A. Pleskunov; scientific editor V.I. Zenkov, Izd-vo Ural University, Ekaterinburg, 2020 (in Russian), 200 pp. [URL](#)