Оптимизация алгоритма календарного планирования с помощью динамического программирования

С.А. Фофанов

Тольяттинский государственный университет, Тольятти, Россия

Обоснование. Решение задачи составления плана регулирования численности рабочих на последующие n месяцев имеет большое практическое значение.

Цель — на основе метода динамического программирования (ДП) разработать программный продукт для решения задачи календарного планирования рабочего графика.

Методы. Поставленная задача о календарном планировании рабочего графика в данной работе решается методом динамического программирования. Метод ДП — это алгоритмический подход к решению оптимизационных задач. Он основан на принципе разбиения составной задачи на отдельные подзадачи и использовании комбинаций их решений для получения оптимального результата.

Решим задачу менеджмента. Предположим, что предпринимателю необходимо составить план регулирования численности рабочих на последующие n месяцев, при котором затраты на найм, увольнение и содержание лишних рабочих будут минимальными.

Введем обозначения:

x_i — количество рабочих, которые работают в месяце i; a_i — минимальное количество рабочих, необходимых в месяце i; $f_i(x_{i-1})$ — оптимальные затраты в месяце i, денежных единиц.

Результатом решения рассматриваемой задачи является вектор $x^* = (x_1^*, x_2^*, \ldots x_n^*)$, компоненты которого равны оптимальному количеству рабочих в текущем месяце.

Убытки предпринимателя на содержание лишних рабочих $U_л(x_i - a_i)$ можно представить в виде функциональной зависимости:

$$U_л(x_i - a_i) = 4(x_i - a_i)\ (i = 1; n). \quad (1)$$

Расходы на найм и увольнение рабочих $U_н(x_i - x_{i-1})$ можно представить в виде:

$$U_н(x_i - x_{i-1}) = \begin{cases} 5 + 3(x_i - x_{i-1}), \text{ при } x_i > x_{i-1}, \\ 0, \text{ при } x_i \le x_{i-1}. \ (i = 1; n) \end{cases} \quad (2)$$

Составим рекуррентные формулы (уравнение Беллмана) для вычисления функции цели:

$$f_6(x_5) = \min \{U_л(x_6 - a_6) + U_н(x_6 - x_5)\}; \quad (3)$$

$$f_i(x_{i-1}) = \min \{U_л(x_i - a_i) + U_н(x_i - x_{i-1}) + f_{i+1}(x_i)\}\ (i = \underline{1; n}) \quad (4)$$

Алгоритм решения реализован на языке C++.

При решении задачи методом ДП построение алгоритма начинаем с последнего месяца, двигаясь по месяцам в обратном порядке. Для каждого из месяцев рассчитываются и запоминаются значения оптимальных затрат и оптимального количества рабочих, с учетом возможной численности рабочих в следующем месяце. На первом шаге считаем только сумму затрат на найм и увольнение, а также затрат на содержание лишних рабочих. Начиная со второго шага к полученной сумме прибавляем затраты следующего месяца. Таким образом, на начало первого месяца остается единственный вариант, соответствующий минимальному значению суммарных затрат за первый месяц и все последующие. Используя обратный ход, получаем последовательность управлений начиная с первого месяца и до начала последнего месяца, т. е. за весь рассматриваемый период.

В таблице 1 представлен последний этап реализации алгоритма, оптимизации в целом.

Таблица 1. Оптимальные значения в первом месяце

x_0	$a_1 = 6$				Оптимальное решение	
	$U_л(x_i - a_i) + U_н(x_1 - x_0) + f_2(x_1)$					
	$x_1 = 6$	$x_1 = 7$	$x_1 = 8$	$x_1 = 9$	$f_1(x_0)$	x_1^*
6	0+0+30	4+8+22	8+11+22	12+14+19	30	6

```
std::vector<int> WorkScheduleDyn(std::vector<int> minWorkers, int (*Ul)(int x), int (*Un)(int x)) {
    std::vector< std::vector<int> > limits; //вычисление границ возможных значений числа рабочих в каждом месяце
    limits.push_back({ minWorkers[0], minWorkers[0] });
    int max = 0;
    for (int j = 0; j < minWorkers.size(); j++)
        if (max < minWorkers[j]) max = minWorkers[j];
    for (int i = 1; i <= minWorkers.size(); i++) {
        std::vector<int> v;
        limits.push_back(v);
        limits[i].push_back(minWorkers[i - 1]);
        limits[i].push_back(max);//Выставляем всем максимальную границу на случай если увольнение не бесплатное
    }
    std::vector< std::map<int, int> > bestWorkers;//лучшее число рабочих в месяце относительно предыдущего
    std::map<int, int> bestSpent; //лучшие общие затраты относительно предыдущего месяца
    std::map<int, int> numWorkersFirstMonth;//Строим таблицу для последнего месяца
    for (int i = limits[limits.size() - 2][0]; i <= limits[limits.size() - 2][1]; i++) {
        int s = 1000000, n = 0;
        for (int j = limits[limits.size() - 1][0]; j <= limits[limits.size() - 1][1]; j++) {
            int currentSpent = Ul(j - minWorkers[minWorkers.size() - 1]) + Un(j - i);
            if (s > currentSpent) { s = currentSpent; n = j; }
        }
        bestSpent[i] = s;
        numWorkersFirstMonth[i] = n;
    }
    bestWorkers.push_back(numWorkersFirstMonth);
    for (int i = limits.size() - 2; i > 0; i--) { //Строим таблицы для всех остальных месяцев
        std::map<int, int> numWorkers, spent;
        for (int j = limits[i - 1][0]; j <= limits[i - 1][1]; j++) {
            int s = 1000000, n = 0;
            for (int k = limits[i][0]; k <= limits[i][1]; k++) { //В bestSpent затраты в следующих месяцах
                int currentSpent = Ul(k - limits[i][0]) + Un(k - j) + bestSpent[k];
                if (s > currentSpent) { s = currentSpent; n = k; }
            }
            numWorkers[j] = n;
            spent[j] = s;
        }
        bestSpent = spent;
        bestWorkers.push_back(numWorkers);
    }
    std::vector<int> bestWorkersNum; //Обратным ходом по таблицам собираем искомую последовательноть
    bestWorkersNum.push_back(bestWorkers[bestWorkers.size() - 1][limits[0][0]]);
    for (int i = bestWorkers.size() - 2; i >= 0; i--)
        bestWorkersNum.push_back(bestWorkers[i][bestWorkersNum[bestWorkersNum.size()-1]]);
    bestWorkersNum.push_back(bestSpent[bestWorkersNum[0]]);
    return bestWorkersNum;
}
```

Рис. 1. Листинг алгоритма на основе метода ДП

```
Рабочих до первого месяца – 6
Минимум рабочих: 6 7 9 7 6 8
Результат: 6 7 9 7 6 8  затраты – 30
Ul вызвано 496 раз, Un вызвано 496 раз.

Рабочих до первого месяца – 6
Минимум рабочих: 6 7 9 7 6 8
Результат: 6 7 9 7 6 8  затраты – 30
Ul вызвано 42 раз, Un вызвано 42 раз.
```

Рис. 2. Результаты работы алгоритмов

Оценку сложности вычислений алгоритма методом ДП по временным затратам можно описать формулой (5):

$$O(n(\max - \min)^2). \quad (5)$$

Для сравнения построен алгоритм, который перебирает все возможные последовательности численности рабочих в каждом месяце по порядку и находит среди них ту, при которой суммарные затраты будут минимальными.

Оценку сложности вычислений алгоритма перебора по временным затратам можно описать формулой (6):

$$O((\max - \min)^n). \quad (6)$$

Результаты. Для сравнения эффективности алгоритмов вычислили количество вызовов функций подсчета затрат, которые соответствуют формулам 1 и 2.

Результаты решения методом ДП и перебором выведены на рис. 1 и 2 в соответствующем порядке.

Выводы. Результаты показывают, что для решения задачи перебором всех вариантов было произведено значительно больше вычислений.

Таким образом, применение метода ДП значительно увеличивает скорость решения задачи календарного планирования.

Ключевые слова: динамическое программирование; ДП; календарное планирование; алгоритмы; оптимизация.

Сведения о авторе:

Семён Андреевич Фофанов — студент, группа ПМИб-2102а, направление 01.03.02 «Прикладная математика и информатика»; Тольяттинский государственный университет, Тольятти, Россия. E-mail: semen-fofanov213@yandex.ru

Сведения о научном руководителе:

Наталья Алексеевна Сосина — кандидат технических наук, доцент, доцент кафедры прикладной математики и информатики; Тольяттинский государственный университет, Тольятти, Россия. E-mail: Sosina1959@yandex.ru