МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Государственное образовательное учреждение высшего профессионального образования
«Нижегородский государственный университет им. Н.И. Лобачевского»
Национальный исследовательский университет

Экономический факультет
Кафедра экономической информатики

Визгунов Н.П.

Динамическое программирование в экономических задачах с применением системы SciLab

Учебно-методическое пособие

Рекомендован методической комиссией экономического факультета для студентов высших учебных заведений, обучающихся по специальности 080801 « Прикладная информатика (в экономике) »

Нижний Новгород, 2011

УДК 338.23:519.876
Динамическое программирование в экономических задачах с применением системы SciLab / Н.П.Визгунов. — Н.Новгород: ННГУ, 2011.

Методическая разработка предназначена для студентов экономического факультета дневного, вечернего и заочного обучения. На примерах показано, как решать экономические задачи, которые сводятся к задачам динамического программирования. Задачи решаются не только вручную, но и предлагаются соответствующие программы в системе SciLab. Эти программы позволяют решать реальные экономические задачи, содержащие многие тысячи переменных. Программы являются достаточно простыми, поэтому они могут быть легко адаптированы и для решения аналогичных задач.

Программы на языке SciLab приведены вместе с результатами работы этих программ. Чтобы убедиться, что программы динамического программирования правильно решают задачу, приводятся также тексты и результаты работы программ полного перебора. Работа набиралась с помощью XeLaTeX, для набора текстов программ использовался пакет *listing*. Чтобы проще было скопировать тексты программ, они повторно приведены в конце работы, уже без применения *listing*.

Составитель: доцент Н. П. Визгунов

Рецензенты: зав. кафедрой ГМУ экономического ф-та,
профессор, к.э.н. Ю. А. Лебедев.

Нижегородский государственный университет им. Н.И.Лобачевского
2011

Содержание

1 Динамическое программирование **3**

2 Задача распределения инвестиций **3**
2.1 Решение задачи распределения инвестиций с помощью таблиц. 5
2.1.1 Заполнение таблицы этапа 4. 6
2.1.2 Заполнение таблицы этапа 3 и последующих. . . . 7
2.1.3 Получение оптимального решения в задаче распределения инвестиций. 8
2.2 Графическое решение задачи распределения инвестиций. . 9

3 Задача распределения инвестиций — общий случай **10**
3.1 Графическое решение задачи 11
3.2 Решение с помощью таблиц 13
3.3 Задача распределения инвестиций на компьютере. 14

4 Задача о загрузке(о рюкзаке или о ранце) **19**
4.1 Задача о рюкзаке на компьютере. 20

5 Задача о надежности **27**
5.1 Задача о надёжности на компьютере. 28

6 Задача календарного планирования трудовых ресурсов **32**
6.1 Календарное планирование на компьютере 34

7 Задача о дилижансах **38**

8 Управление запасами **40**
8.1 Вычисление оптимального решения 43
8.2 Управление запасами на компьютере 43

9 Замена оборудования. **47**
9.1 Замена оборудования на компьютере 49

10 Программы на *Python* **53**
10.1 Решение задачи о распределении инвестиций 53
10.1.1 Динамическое программирование 53
10.1.2 Полный перебор 54
10.2 Задача о загрузке 55
10.2.1 Динамическое программирование – вычислить C и R 55
10.2.2 Динамическое программирование – без вычисления C и R 57
10.2.3 Полный перебор 58
10.3 Решение задачи о надёжности 59
10.3.1 Динамическое программирование 59
10.3.2 Полный перебор 61
10.4 Календарное планирование трудовых ресурсов 62
10.4.1 Динамическое программирование 62
10.4.2 Полный перебор 63
10.5 Управление запасами 64
10.5.1 Динамическое программирование 64
10.5.2 Полный перебор 66
10.6 Замена оборудования 67
10.6.1 Динамическое программирование 67
10.6.2 Полный перебор 68
Список литературы 70

Список таблиц

1 Задача распределения 5 миллионов рублей 3
2 Полный перебор 4
3 Задача об инвестициях, этап 4 6
4 Задача об инвестициях, этап 3 7
5 Задача об инвестициях, этап 2 8
6 Задача об инвестициях, этап 1 8
7 Инвестиции 8 миллионов рублей без «пустых» проектов . 10
8 Вычисление состояний y_j в задаче распределения инвестиций 11
9 Инвестиции без пустых проектов, этап 4 13
10 Инвестиции без пустых проектов, этап 3 14
11 Инвестиции без пустых проектов, этап 2 14
12 Инвестиции без пустых проектов, этап 1 14
13 Задача о рюкзаке, этап 3 20
14 Задача о рюкзаке, этап 2 20
15 Задача о рюкзаке, этап 1 20
16 Данные о стоимости и надежности каждой компоненты прибора. 27
17 Задача о надежности, этап 3 27
18 Задача о надежности, этап 2 28
19 Задача о надежности, этап 1 28
20 Календарное планирование, этап 5 33
21 Календарное планирование, этап 4 33
22 Календарное планирование, этап 3 33
23 Календарное планирование, этап 2 33
24 Календарное планирование, этап 1 34
25 Задача о дилижансах, этап 4 39
26 Задача о дилижансах, этап 3 40
27 Задача о дилижансах, этап 2 40
28 Задача о дилижансах, этап 1 40
29 Затраты на производство x_j единиц продукции. 40
30 Управление запасами, этап 4 41
31 Управление запасами, этап 3 42
32 Управление запасами, этап 2 42
33 Управление запасами, этап 1 42
34 Задача о замене оборудования 47
35 Замена оборудования, этап 5 48
36 Замена оборудования, этап 4 48
37 Замена оборудования, этап 3 48
38 Замена оборудования, этап 2 48

Список иллюстраций

1 Пояснение обозначений для отрезка $j, \ldots, n$ в задаче распределения инвестиций. 6
2 Графическое решение задачи о распределении инвестиций. 9
3 Графическое решение задачи о распределении инвестиций. 11
4 Инвестиции $y_4 \in 1:3$ миллионов рублей 12
5 Инвестиции $y_3 \in 2:5$ миллионов рублей 12
6 Инвестиции $y_1 = 8$ миллионов рублей 13
7 Задача календарного планирования трудовых ресурсов. . . 34
8 Графическая иллюстрация исходных данных и ответа в задаче о календарном планировании трудовых ресурсов. . . 34
9 Задача о дилижансах. 39
10 Ответ в графическом виде для задачи о замене оборудования. 49

1. Динамическое программирование

Динамическое программирование - это вычислительный метод для решения задач определенной структуры. Динамическое программирование возникло и сформировалось в 1950–1953 гг. благодаря работам Ричарда Беллмана и его сотрудников. Беллман (Bellman) Ричард Эрнест (1920-84) – американский математик. Основные труды по вычислительной математике и теории оптимального управления. Разработал метод динамического программирования. Задачи управления запасами были первыми задачами, которые решались этим методом.

В этой работе рассматривается несколько достаточно простых задач, которые решаются на основе идей динамического программирования. В том числе и решена классическая задача управления запасами. Для каждой из рассмотренных задача составлена программа на языке $SciLab$, которая может решить не только маленькую учебную задачу, но и претендует на решение реальных задач, с тысячами переменных. Чтобы убедиться, что программа правильно решает задачи методом динамического программирования, написаны также программы полного перебора — для проверки основного алгоритма, и для более ясного понимания каждой из рассмотренных экономических задач.

Пакет $SciLab$ – бесплатный французский пакет. Взять его можно на сайте www.scilab.org. Является одним из аналогов широко известного пакета $MatLab$. Язык пакета $SciLab$ несколько отличается от $MatLab$, но чаще всего в лучшую сторону, так как этот язык создан позднее. Другим известным аналогом $MatLab$ являеся бесплатный пакет $Octave$, который широка используется в американских университетах. Программы $MatLab$ идут без каких-либо изменений в системе $Octave$, но есть проблемы с руским языком, непросто установить в $Windows$ и медленно работает.

Очередным преимуществом пакета $SciLab$ является постоянное обновление. Последнее обновление было в марте 2011 года — выпущена версия 5.3.1. Все программы набраны не в $CP-1251$, как делается обычно в $Windows$ разных версий, а в кодовой таблице $UTF-8$. Эта кодировка становится всё более популярной, так как позволяет использовать сразу несколько языков. Наряду с русским и английским языками можно использовать любые другие языки, например, китайские иероглифы, причём в одной программе!

2. Задача распределения инвестиций

Задачей распределения инвестиций будем называть следующую задачу оптимального планирования.

Задача 1 (распределения инвестиций) *Совет директоров фирмы изучает предложения по модернизации четырех предприятий. Для этих целей выделено 5 миллионов рублей. Для каждого предприятия j разработано несколько альтернативных проектов. Каждый из проектов характеризуется суммарными затратами c_j и будущими доходами R_j. На каждом предприятии можно реализовать только по одному проекту. Соответствующие данные приведены в таблице 1.*

Необходимо выбрать такие проекты для каждого предприятия, чтобы фирма получила максимальный годовой доход.

Проект	Пр-тие 1		Пр-тие 2		Пр-тие 3		Пр-тие 4	
	c_1	R_1	c_2	R_2	c_3	R_3	c_4	R_4
$x_j = 1$	0	0	0	0	0	0	0	0
$x_j = 2$	1	3	3	5	1	4	2	3
$x_j = 3$	–	–	5	9	2	6	–	–

Таблица 1: Задача распределения 5 миллионов рублей

Обратим внимание, что все числа в таблице – целые. Затраты c_j измеряются в миллионах рублей, доходы R_j – в миллионах рублей в год. Через x_j обозначен номер проекта, который выбирается на j–ом предприятии. В нашей конкретной задаче проект с номером 1 пустой, расширения предприятия при этом не предполагается.

Самый простой способ решения задачи — использовать полный перебор.

x_1	x_2	x_3	x_4	Затраты $\sum c_j(x_j)$	Доходы $\sum R_j(x_j)$	План допустим ?
1	1	1	1	0	0	Да
1	1	1	2	2	3	Да
1	1	2	1	1	4	Да
1	1	2	2	1+2=3	4+3=7	Да
1	1	3	1	2	6	Да
1	1	3	2	2+2=4	6+3=9	Да
…	…	…	…	…	…	…
1	3	3	2	0+5+2+2=9	Нет	Нет
2	1	1	1	1+0+0+0=1	3+0+0+0=3	Да
2	1	1	2	1+0+0+2=3	3+0+0+3=6	Да
…	…	…	…	…	…	…
2	3	3	2	1+5+2+2=10	Нет	Нет

Таблица 2: Полный перебор

Задача имеет $2 * 3 * 3 * 2 = 36$ возможных решений, причем некоторые из них не являются допустимыми, так как для реализации такого плана потребуется денег больше, чем 5 миллионов рублей.

Перейдем теперь к математической формулировке задачи распределения инвестиций.

$$\left.\begin{array}{r} f_1(y_1) = \max\limits_{x_1,\ldots,x_n} \left\{\sum_{j=1}^{n} R_j(x_j)\right\}, \\ \sum_{j=1}^{n} c_j(x_j) \le y_1, \\ x_j - \text{целые}, j = 1,\ldots,n. \end{array}\right\} \quad (1)$$

В нашей задаче $y_1 = 5$ миллионов рублей, $n = 4$ – количество предприятий. Будем в дальнейшем использовать следующие обозначения.

y_j – количество денег, выделенных для расширения предприятий $j, j+1,\ldots,n$.

x_j – номер проекта, выбранного на предприятии j.

$c_j(x_j)$– затраты (от слова *cost*) на j-ом предприятии, когда выбран проект с номером x_j. Измеряется в миллионах рублей.

$R_j(x_j)$ – годовой доход (*result, revenue*), который будет получен от реализации проекта x_j. Измеряется в миллионах рублей в год.

$f_1(y_1)$ – максимальный годовой доход, который будет получен от реализации проектов $x_1, x_2, \ldots, x_n$ при заданном объеме инвестиций y_1 миллионов рублей.

Мы пользуемся довольно сложными обозначениями, но их надо просто один раз запомнить. В дальнейшем будем пользоваться только этими обозначениями и новых постараемся не вводить. В задаче 1 выберем фиксированное значение $x_1 = \tilde{x}_1$. Например, сначала $\tilde{x}_1 = 1$.

$$\left.\begin{array}{r} \tilde{f}_1(y_1) = R_1(\tilde{x}_1) + \max\limits_{x_2,\ldots,x_n} \left\{\sum_{j=2}^{n} R_j(x_j)\right\}, \\ c_1(\tilde{x}_1) + \sum_{j=2}^{n} c_j(x_j) \le y_1, \\ x_j - \text{целые}, j = 2,\ldots,n. \end{array}\right\} \quad (2)$$

Можно записать более компактно

$$\tilde{f}_1(y_1) = R_1(\tilde{x}_1) + f_2(y_1 - c_1(\tilde{x}_1)), \quad (3)$$

где $\tilde{f}_1(y_1)$ – наибольший годовой доход при инвестициях y_1 и при фиксированном значении $\tilde{x}_1 = 1$. В нашей конкретной задаче выбираем максимум из первой половины таблицы 1, то есть не из всех 36 значений, а только из первых 18 строчек этой таблицы. $f_2(y_1 - c_1(\tilde{x}_1))$ – это годовой доход от расширения предприятий $2, \ldots, n$, если инвестировано денег $y_1 - c_1(\tilde{x}_1)$ миллионов рублей.

Остается перебрать все возможные значения $\tilde{x}_1$. В нашем конкретном примере осталось посмотреть значения доходов для второй половины таблицы 1, где значение $\tilde{x}_1 = 2$. В результате получим полный перебор, то есть найдем оптимальное значение $f_1(y_1)$.

$$f_1(y_1) = \max_{\tilde{x}_1}\{R_1(\tilde{x}_1) + f_2(y_1 - c_1(\tilde{x}_1))\}. \tag{4}$$

Тильду в обозначении переменной x_1 можно не писать, $y_2 = y_1 - c_1(x_1)$ – количество денег, инвестированных в предприятия $2, \ldots, n$. Из $y_2 \geq 0$ следует $c_1(x_1) \leq y_1$.

Окончательно получаем

$$f_1(y_1) = \max_{x_1 | c_1(x_1) \leq y_1}\{R_1(x_1) + f_2(y_1 - c_1(x_1))\}, \tag{5}$$

О чем говорит выведенная формула? Если мы умеем решать задачу распределения инвестиций для $(n-1)$ предприятия $2, \ldots, n$ для всех возможных значений инвестиций $y_2 = 0, 1, \ldots, 5$, то задача распределения $y_1 = 5$ миллионов рублей легко решается. Но вычисление $f_2(y_2)$ также можно осуществить по аналогичной формуле, используя значения доходов $f_3(y_3)$ для $(n-2)$ предприятий, где $y_3 = y_2 - c_2(x_2)$ – деньги для предприятий $3, \ldots, n$. Формула выводится по известной уже схеме:

$$\begin{aligned} f_2(y_2) = \max_{x_2 | c_2(x_2) \leq y_2}\{R_2(x_2) + f_3(y_2 - c_2(x_2))\}, \\ y_2 \in \{0, \ldots, 5\}. \end{aligned} \tag{6}$$

По той же схеме выводятся формулы для вычислений $f_3(y_3), \ldots, f_n(y_n)$.

Осталось записать формулу в общем случае, для вычисления максимального дохода от модернизации предприятий $j, j+1, \ldots, n$:

$$\left.\begin{aligned} &f_{n+1}(y_{n+1}) = 0, \\ &f_j(y_j) = \max_{x_j | c_j(x_j) \leq y_j}\{R_j(x_j) + f_{j+1}(y_j - c_j(x_j))\}, \\ &\qquad\qquad j = n, n-1, \ldots, 1. \end{aligned}\right\} \tag{7}$$

Эта зависимость носит название *уравнение Беллмана для процедуры обратной прогонки.* Обратная прогонка, потому что сначала надо решить задачу для последнего предприятия n, затем для предприятий $n-1, n$, затем для $n-2, n-1, n$ и так далее, на последнем шаге надо вычислить $f_1(y_1)$ для всех n предприятий $1, 2, \ldots, n$.

Аргументы функций x_j и y_j в динамическом программировании имеют свои названия.

x_j – переменная или *управление* у Беллмана,

y_j – параметр, *состояние* в терминологии Беллмана – количество денег для предприятий $j, \ldots, n$ в нашей задаче о распределении инвестиций.

Обозначения еще раз иллюстрируются для отрезка $j, j+1, \ldots, n$ на рисунке 1.

2.1. Решение задачи распределения инвестиций с помощью таблиц.

Рекуррентное уравнение Беллмана для процедуры обратной прогонки записывается следующим образом для нашей конкретной задачи:

$$\begin{aligned} &f_5(y_5) = 0, \\ &f_j(y_j) = \max_{x_j | c_j(x_j) \leq y_j}\{R_j(x_j) + f_{j+1}(y_j - c_j(x_j))\},\ j = 4, 3, 2, 1. \end{aligned}$$

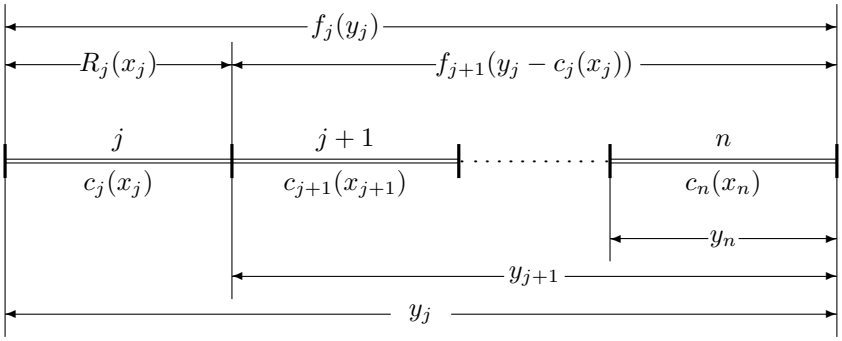


Рис. 1: Пояснение обозначений для отрезка $j, \ldots, n$ в задаче распределения инвестиций.

Сначала надо решить задачу для четвертого предприятия, затем для третьего и четвертого предприятия, затем для 2, 3 и 4 предприятия и, наконец, для предприятий 1, 2 , 3 и 4. В таблицах 3 – 6 приводятся результаты расчетов. Результаты получены с помощью таблиц, в которые помещены значения функции $\{R_j(x_j) + f_{j+1}(y_j - c_j(x_j))\}$ двух переменных y_j и x_j. Функция вычислена для каждого возможного значения состояния y_j и управления x_j. Если вычисление функции для некоторых пар (y_j, x_j) невозможно, то в качестве значения функции ставим черточку.

Этап 4. Предприятие 4.

$$f_4(y_4) = \max_{x_4=1:2 | c_4(x_4) \le y_4} \{R_4(x_4)\}.$$

	$R_4(x_4)$		Оптимальное решение	
y_4	$x_4 = 1$	$x_4 = 2$	$f_4(y_4)$	x_4^*
0	0	–	0	1
1	0	–	0	1
2	0	3	3	2
3	0	3	3	2
4	0	3	3	2
5	0	3	3	2

Таблица 3: Задача об инвестициях, этап 4

2.1.1. Заполнение таблицы этапа 4.

Подробно опишем построение таблицы 3, соответствующей этапу 4. В таблице вычислены значения функции $\{R_4(x_4)\}$ для 6 возможных состояний $y_4 \in \{0, \ldots, 5\}$ (строки таблицы) и 2 возможных управлений $x_4 \in \{1, 2\}$ (столбцы таблицы). Из исходной таблицы 1, расположенной на стр. 3, видно, что $R_4(1) = 0$. (это самое правое верхнее число исходной таблицы). В таблице 3 весь столбец $x_4 = 1$ заполнен этими нулями.

В столбце $x_4 = 2$ стоят тройки везде, кроме 2 первых строчек, где стоят черточки. Проект номер 2 на предприятии 4 стоит 2 миллиона рублей ($c_4(2) = 2$), поэтому для его реализации денег должно быть не менее 2 миллионов, и варианты с $y_4 = 0$ и $y_4 = 1$ недопустимы. Тройка $R_4(2) = 3$ взята из исходной таблицы 1 – это доход от реализации 2-го проекта на 4 предприятии. Если денег на реализацию этого проекта хватает $y_4 \ge 2$, то получим доход в три миллиона рублей в год.

Последние 2 столбца таблицы 3 этапа 4 дают оптимальное решение. В каждой строчке таблицы $\{R_4(x_4)\}$ надо найти максимальное число и поместить его в столбец $f_4(y_4)$. В последний столбец x_4^* помещаем номер того столбца (1 или 2), который найденному максимуму соответствует. Например, в строке $y_4 = 0$ в таблице $\{R_4(x_4)\}$ стоит 0 и черточка. Ноль

находится в столбце $x_4 = 1$, поэтому последние два числа в строке $y_4 = 0$ — ноль и единица ($f_4(0) = 0$ и $x_4^*(0) = 1$).

Этап 3. Предприятия 3 и 4.

$$f_3(y_3) = \max_{x_3=1:3|c_3(x_3)\le y_3} \{R_3(x_3) + f_4(y_3 - c_3(x_3))\}.$$

	$R_3(x_3) + f_4(y_3 - c_3(x_3))$			Оптимальное решение	
y_3	$x_3 = 1$	$x_3 = 2$	$x_3 = 3$	$f_3(y_3)$	x_3^*
0	0+0=0	–	–	0	1
1	0+0=0	4+0=4	–	4	2
2	0+3=3	4+0=4	6+0=6	6	3
3	0+3=3	4+3=7	6+0=6	7	2
4	0+3=3	4+3=7	6+3=9	9	3
5	0+3=3	4+3=7	6+3=9	9	3

Таблица 4: Задача об инвестициях, этап 3

2.1.2. Заполнение таблицы этапа 3 и последующих.

Заполнение таблицы 4 опишем подробно. В первом столбце перечисляются все возможные состояния $y_3 \in 0, \dots, 5$. Последние столбцы 5 и 6 содержат оптимальное решение $f_3(y_3)$ и x_3^*. Содержательно оптимальное решение говорит об максимальном доходе, который можно получить при оптимальном расходе y_3 миллионов рублей, выделенных для предприятий 3 и 4. Оптимальное управление x_3^* – номер выбранного проекта для реализации на 3 предприятии, который позволит получить этот максимальный доход.

Три столбца, соответствующие управлениям $x_3 = 1, 2$ и 3, заполняются довольно просто. Надо для каждой допустимой пары аргументов (y_3, x_3) вычислить значение функции $\{R_3(x_3) + f_4(y_3 - c_3(x_3))\}$. Формула для вычисления этой функции записана сверху в таблице 4 без фигурных скобок.

Первое слагаемое формулы $\{R_3(x_3) + \dots\}$ в столбце $x_3 = 1$ равно нулю ($R_3(1) = 0$), в столбце $x_3 = 2$ первое слагаемое равно 4 ($R_3(2) = 4$), и в столбце ($x_3 = 3$) $R_3(3) = 6$. Числа $R_3(x_3)$ берутся из исходной таблицы 1. Это столбец R_3 для предприятия 3 – доходы, которые будут получены на этом предприятии от реализации проектов 1, 2 или 3.

Обратим внимание, что столбцы $x_3 = 2$ и $x_3 = 3$ нашей таблицы начинаются с черточек, такие сочетания (y_3, x_3) невозможны. Объяснения этому дадим чуть позже.

Второе слагаемое формулы $\{\dots + f_4(y_3 - c_3(x_3))\}$ берется из предыдущей таблицы этапа 4. Вспомним, что $y_3 - c_3(x_3) = y_4$. Если из денег y_3, выделенных на предприятия 3 и 4 вычесть $c_3(x_3)$ миллионов рублей – стоимость проекта x_3 для предприятия 3, то в результате останется y_4 миллионов рублей для предприятия 4. Столбец $x_3 = 1$ соответствует пустому проекту, поэтому $y_3 = y_4$ и в качестве второго слагаемого переписывается оптимальный доход $f_4(y_4)$ из предыдущей таблицы.

Заполняем второй столбец $x_3 = 2$ таблицы 4. Второй столбец соответствует проекту с характеристиками $(c_3, R_3) = (1, 4)$. Проект с номером 2 стоит $c_3(2) = 1$ миллион рублей, поэтому из $y_3 - c_3(x_3) \ge 0$ следует $y_3 \ge 1$. Из этих соображений столбец $x_3 = 2$ начинается с черточки. Если у нас нет миллиона, то проект стоимостью 1 миллион реализовать нельзя. Ниже черточки в качестве второго слагаемого переписывается начало столбца $f_4(y_4)$ из предыдущей таблицы.

Столбец $x_3 = 3$ вычисляется для проекта с характеристиками (c_3, R_3) = $(2, 6)$. Проект стоит 2 миллиона рублей и дает доход 6 миллионов в год. По этой причине 2 первых элемента столбца – черточки. Для допустимых вариантов при $y_3 \ge 2$ первое слагаемое в этом столбце равно 6. Второе слагаемое – начало столбца $f_4(y_4)$ из таблицы 3 этапа 4.

Оптимальное решение в последних 2 столбцах считается традиционно. В каждой строчке y_3 в столбцах $x_3 = 1, 2$ и 3 находим максимум и записываем его в $f_3(y_3)$. В x_3^* записываем номер того столбца, где этот максимум находится. Например, для последней строки таблицы $y_3 = 5$ $\max\{3, 7, 9\} = 9$. Это число записывается в качестве $f_3(5)$ в предпоследнем столбце таблицы, а в столбце $x_3^*(5)$ записываем 3, так как 9 находится в столбце $x_3 = 3$.

Этап 2. Предприятия 2, 3 и 4.

$$f_2(y_2) = \max_{x_2=1:3|c_2(x_2)\le y_2}\{R_2(x_2) + f_3(y_2 - c_2(x_2))\}.$$

	$R_2(x_2) + f_3(y_2 - c_2(x_2))$			Оптимальное решение	
y_2	$x_2 = 1$	$x_2 = 2$	$x_2 = 3$	$f_2(y_2)$	x_2^*
0	0+0=0	–	–	0	1
1	0+4=4	–	–	4	1
2	0+6=6	–	–	6	1
3	0+7=7	5+0=5	–	7	1
4	0+9=9	5+4=9	–	9	1, 2
5	0+9=9	5+6=11	0+9=9	11	2

Таблица 5: Задача об инвестициях, этап 2

Этап 1. Предприятия 1, 2, 3 и 4.

$$f_1(y_1) = \max_{x_1=1:2|c_1(x_1)\le y_1}\{R_1(x_1) + f_2(y_1 - c_1(x_1))\}.$$

	$R_1(x_1) + f_2(y_1 - c_1(x_1))$		Оптимальное решение	
y_1	$x_1 = 1$	$x_1 = 2$	$f_1(y_1)$	x_1^*
0	0+0=0	–	0	1
1	0+4=4	3+0=3	4	1
2	0+6=6	3+4=7	7	2
3	0+7=7	3+6=9	9	2
4	0+9=9	3+7=10	10	2
5	0+11=11	3+9=12	12	2

Таблица 6: Задача об инвестициях, этап 1

Остальные таблицы для этапов $j = 2$ и $j = 1$ заполняются аналогично. Записывается уравнение Беллмана для конкретного j, заполняется таблица для всех допустимых состояний y_j и управлений x_j. После этого ищется максимум по строкам таблицы и находится оптимальное решение, которое располагается в 2 последних столбцах любой таблицы.

2.1.3. Получение оптимального решения в задаче распределения инвестиций.

Когда все 4 таблицы заполнены и получены для каждого j оптимальное решение, надо записать оптимальное решение всей задачи. Из последней таблицы 6 этапа 1 видно, что получаемый максимальный доход составляет $f_1(5) = 12$ миллионов рублей в год. Для этого нужно на 1 предприятии применить 2 проект $x_1^* = 2$. Характеристики второго проекта $(c_1(2), R_1(2)) = (1, 3)$. Эти числа в столбцах c_1 и R_1 подчеркнуты в исходной таблице 1. На предприятия 2, 3 и 4 остается денег $y_2 = y_1 - c_1(2) = 5 - 1 = 4$ миллиона рублей.

Переходим к таблице этапа 2 и подчеркиваем в строке $y_2 = 4$ числа $f_2(4) = 9$ и $x_2^*(4) = 1$. Из-за того, что в качестве $x_2^*(4)$ стоит не только 1 (только что подчеркнутая), но еще и число 2, можем утверждать, что оптимальных решений в нашей задаче больше одного. $x_2^* = 1$ соответствует

проекту с характеристиками $(c_2(1), R_2(1)) = (0, 0)$. Эти числа подчеркиваются в исходной таблице 1. $y_3 = y_2 - c_2(1) = 4 - 0 = 4$, поэтому на предприятия 3 и 4 переходят все 4 миллиона рублей.

Из таблицы этапа 3 выясняем, как эти $y_3 = 4$ миллиона рублей могут быть оптимально использованы. Подчеркиваем $f_3(4) = 9$ и $x_3^*(4) = 3$. Третий проект третьего предприятия $x_3^* = 3$ имеет характеристики $(c_3, R_3) = (2, 6)$. Подчеркиваем эти числа в исходной таблице 1. $y_4 = y_3 - c_3(3) = 4 - 2 = 2$.

Как потратить оптимально $y_4 = 2$ миллиона рублей видно из таблицы этапа 4. Для этого надо применить $x_1^* = 2$ проект на 4 предприятии. Подчеркиваем в исходной таблице $c_4(2) = 2$ и $R_4(2) = 3$ и записываем окончательный ответ.

Для того, чтобы оптимальным образом потратить 5 миллионов рублей на 4 предприятиях, надо выбирать проекты

$$x^* = (2, 1, 3, 2).$$

При этом будет получен максимальный ежегодный доход $f_1(5) = 12$ миллионов рублей в год.

Полезно проверить наши вычисления. Общая стоимость всех проектов $\sum_j c_j(x_j^*) = 1 + 0 + 2 + 2 = 5$ миллионов рублей. Ежегодный доход действительно равняется 12: $f_1(5) = \sum_j R_j(x_j^*) = 3 + 0 + 6 + 3 = 12$ миллионов рублей в год.

В качестве упражнения попробуйте получить все оптимальные решения нашей задачи. Подсказка. Всего в задаче 2 оптимальных решения. Второе решение $x_{(2)}^* = (2, 2, 2, 1)$.

2.2. Графическое решение задачи распределения инвестиций.

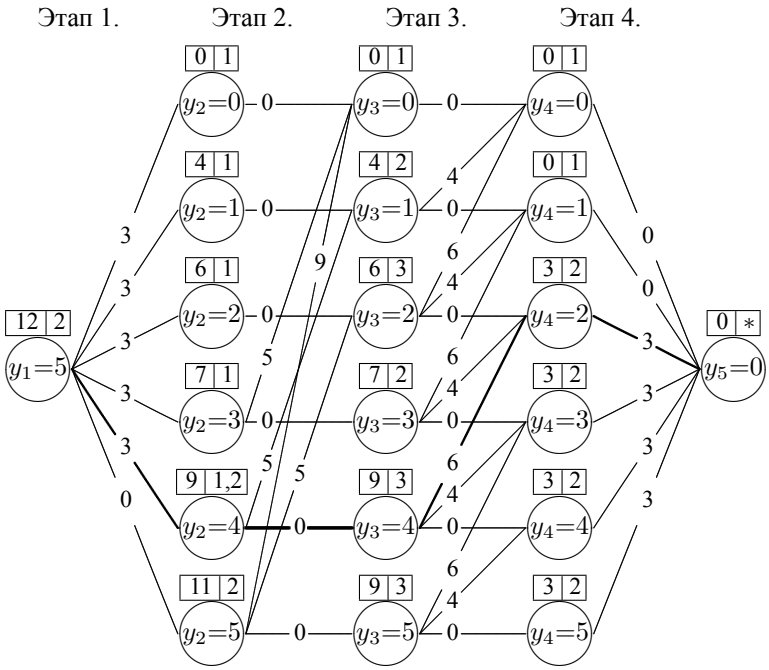


Рис. 2: Графическое решение задачи о распределении инвестиций.

Графическое решение задачи полностью повторяет решение, полученное с помощью таблиц. Каждая вершина графа соответствует некоторому известному состоянию системы, каждой дуге поставлен в соответствие

определенный проект. Считаем, что дуги имеют направление справа налево. Сначала над вершиной $y_5 = 0$ ставим 0 и $*$. Выполняем этап 4. Для каждого состояния $y_4 \in 0 : 5$ вычисляем пару чисел $f_4(y_4)$ и $x_4^*(y_4)$. Эти два числа – максимальный доход, получаемый на 4 предприятии и номер проекта, который следует для этого применять, записываем над вершиной y_4. Понятно, что над вершинами $y_j \in 0 : 5$ стоят те же самые числа, которые стояли в двух правых столбцах таблицы 3, расположенной на странице 6. Более подробное объяснение графического решения будет в следующем параграфе на примере чуть более сложной задачи.

3. Задача распределения инвестиций — общий случай

Существенной особенностью предыдущей задачи является наличие в таблице 1 первой строки, состоящей из одних нулей. Предполагается, что можно модернизацию любого предприятия не проводить вообще. В реальных задачах это не так. Раз уж предприятие мы собираемся модернизировать, то хотя бы самый дешевый способ модернизации мы должны запланировать. Следующая задача как раз такого типа.

Проект	Пр-тие 1		Пр-тие 2		Пр-тие 3		Пр-тие 4	
	c_1	R_1	c_2	R_2	c_3	R_3	c_4	R_4
$x_j = 1$	$\underline{1}$	$\underline{1}$	$\underline{2}$	$\underline{2}$	1	1	$\underline{1}$	$\underline{3}$
$x_j = 2$	3	4	4	6	2	3	3	5
$x_j = 3$	—	—	—	—	$\underline{4}$	$\underline{7}$	—	—

Таблица 7: Инвестиции 8 миллионов рублей без «пустых» проектов

Рекуррентное уравнение Беллмана для процедуры обратной прогонки то же самое

$$f_5(y_5) = 0,$$
$$f_j(y_j) = \max_{x_j | c_j(x_j) \le y_j} \{R_j(x_j) + f_{j+1}(y_j - c_j(x_j))\},\ j = 4, 3, 2, 1.$$

Сначала надо решить задачу для четвертого предприятия, затем для третьего и четвертого предприятия, затем для 2, 3 и 4 предприятия и, наконец, для предприятий 1, 2 , 3 и 4. В таблицах 9 – 12 приводятся результаты расчетов.

Единственная разница с предыдущим примером – более узкие рамки для изменения состояний. Для предприятия 4 самый дешевый проект стоит 1 миллион рублей, поэтому $y_4 \ge 1$. Для предприятий 1, 2 и 3 также надо предусмотреть хотя бы самые дешевые проекты, поэтому верхняя граница y_4 не 8 миллионов, а на $c_1(1)+c_2(1)+c_3(1) = 1+2+1 = 4$ миллиона рублей меньше. Таким образом, на 4-ом предприятии можно тратить деньги в диапазоне $y_4 \in 1 : 4$.

На третьем и четвертом предприятиях нижняя граница y_3 равна $c_3(1)+c_4(1) = 1+1 = 2$. Верхняя граница расходов на этих двух предприятиях равна $8 - c_1(1) - c_2(1) = 8-1-2 = 5$. Итак, на шаге 3 для предприятий 3 и 4 можно тратить деньги в следующих размерах $y_3 \in 2 : 5$. Для предприятий 2,3 и 4 минимальное количество денег равно $c_2(1) + c_3(1) + c_4(1) = 2+1+1 = 4$, максимальное количество денег равно $8 - c_1(1) = 8-1 = 7$. Другими словами, $y_2 \in 4 : 7$.

Аналогичные рассуждения дают нам следующую оценку расходов для $y_1 : y_1 \in 5 : 8$. Впрочем, для первого предприятия мы будем использовать только одно значение $y_1 = 8$, больше нам и не требуется для решения.

Все наши расуждения демонстрируются таблицей (8). Для вычислений y_j используются числа $(c_1(1), c_2(1), c_3(1), c_4(1)) = (1, 2, 1, 1)$, которые в таблице подчёркнуты.

Проект	Пр-тие 1		Пр-тие 2		Пр-тие 3		Пр-тие 4	
	c_1	R_1	c_2	R_2	c_3	R_3	c_4	R_4
$x_j = 1$	$\underline{1}$	1	$\underline{2}$	2	$\underline{1}$	1	$\underline{1}$	3
$x_j = 2$	3	4	4	6	2	3	3	5
$x_j = 3$	—	—	—	—	4	7	—	—
	$y_1 = 8$							
			$y_2 \in 4:7$					
					$y_3 \in 2:5$			
							$y_4 \in 1:4$	

Таблица 8: Вычисление состояний y_j в задаче распределения инвестиций

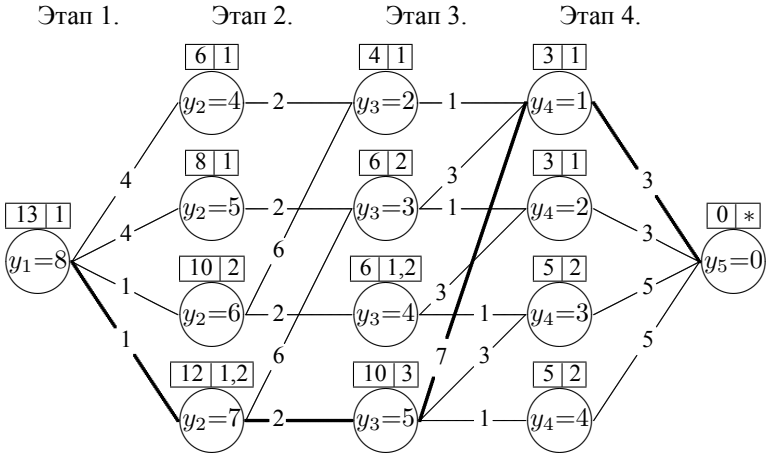


Рис. 3: Графическое решение задачи о распределении инвестиций.

3.1. Графическое решение задачи

Подробно объясним, как это всё делается. Начинаем с шага 4. Максимизируем наши доходы для предприятия 4 для любого остатка денег, выделенных для этого предприятия $y_4 \in 1:4$.

Для расчётов нам потребуются данные для четвертого предприятия из таблицы (7), запишем эти данные рядом с графом. Четвертое предприятия использует свои деньги очень просто. Если хватает денег, $y_4 \geq 3$, то реализуем второй проект. Если же денег выделено меньше, то применяем проект 1. Результаты наших вычислений записаны в прямоугольниках. Например, над вершиной, соответствующей состоянию $y_4 = 4$ стоят числа 5 и 2. Это значит, что наш ежегодный доход будет равен 5 миллионам в год, для этого надо реализовать проект с номером 2. Вершина $y_5 = 0$ это фиктивная вершина, никакого 5-го предприятия нет, но зато можно объяснить наш алгоритм более формально. Если $y_4 - y_5 \geq 3$, применяем второй проект, иначе первый. Здесь $y_4 - y_5$ – количество денег, вложенных в развитие предприятия 4. Если денег вложено 4 миллиона, то лишний 1 миллион не используется, так как он не оказывает влияния на будущие доходы.

Теперь выполняем этап 3 графического решения задачи. Рассмотрим вершину (кружок) $y_3 = 5$. Как 5 миллионов рублей, выделенных для 3 и 4 предприятия, потратить оптимальным образом на третьем предприятиии?

Можно реализовать первый проект, для него затраты $c_3(1) = 1$ и будущие доходы равны $R_3(1) = 1$. На четвёртое предприятие останется денег $5 - c_3(1) = 4$. Другими словами, мы должны соединить линией вершины $y_4 = 4$ и $y_3 = 5$. Доход при этом увеличится с 5 в вершине $y_4 = 4$ до 6 в вершине $y_3 = 5$. Эту линию чаще всего будем называть дугой, она имеет длину $R_3(1) = 1$ и соответствует первому проекту на третьем предприятии.

Попробуем реализовать второй проект с $(c_3(2), R_3(2)) = (2, 3)$. Этот проект стоит 2 миллиона, и для четвёртого предприятия остаётся $5-2 = 3$ миллиона. Следовательно, вершина $y_4 = 3$ соединена дугой длиной 3 с

Проект	Пр-тие 4	
	c_4	R_4
$x_4 = 1$	1	3
$x_4 = 2$	3	5
$x_4 = 3$	—	—

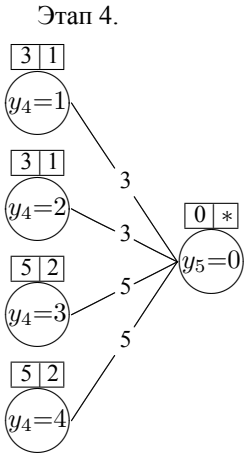


Рис. 4: Инвестиции $y_4 \in 1 : 3$ миллионов рублей

вершиной $y_3 = 5$. При этом будет получен доход 5 + 3 = 8. Больше, чем было при применении первого проекта. Но остался ещй третий проект, для которого $(c_3(3), R_3(3)) = (4, 7)$. Этот проект соответствует дуге от вершины $y_4 = 1$ до вершины $y_3 = 5$, при этом будем получать доход $3 + 7 = 10$, самый большой. Этот максимум записываем в левый прямоугольник над нашим кружком. Рядом стоит число 3 — третий проект надо реализовать для получения дохода в 10 миллионов.

Для остальных вершин этапа 3 всё делается аналогично. Для вершин $y_3 = 3$ и $y_3 = 4$ нам надо проверить только 1 и 2 проекты, а для вершины $y_3 = 2$ допустим только 1 проект.

Проект	Пр-тие 3	
	c_3	R_3
$x_3 = 1$	1	1
$x_3 = 2$	2	3
$x_3 = 3$	4	7

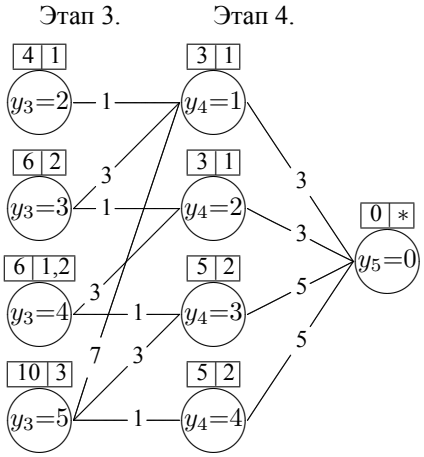


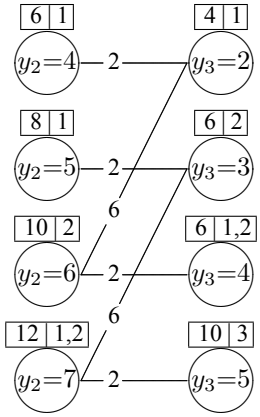
Рис. 5: Инвестиции $y_3 \in 2 : 5$ миллионов рублей

Начинается второй этап. Не будем отображать на рисунке лишнюю информацию. Для решения задачи требуются только оптимальные решения для третьего этапа. Результаты четвёртого этапа на этом шаге нам неинтересны. Они уже учтены в столбце y_3.

Первый этап очень похож на то, что мы делали на этапе 4. Из $y_1 = 8$ вычитаем $y_2 = 7$, получаем 1 миллион, которого хватает на проект 1, но не хватает на проект 2. Далее рассуждения тривиальны. Если денег меньше 3 миллионов, реализуем первый проект, иначе второй.

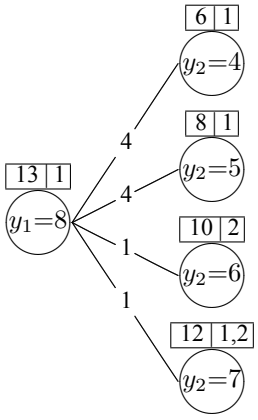
Теперь остаётся получить ответ. Двигаемся в противоположном направлении, а именно, слева направо. Из левой вершины $y_1 = 8$ получаем полный доход $f_1(8) = 13$. Чтобы получить этот максимальный доход, надо применить проект номер $x_1^* = 1$. Числа 13 и 1 записаны в прямоугольниках над нашей вершиной. Число 13 было получено как сумма 1 и 12. Число 12 стоит над вершиной $y_2 = 7$. Из двух решений 1 или 2 выбираем любое. Пусть $x_2^* = 1$. Следующая вершина $y_3 = 5$ даёт очередную компоненту решения $x_3^* = 3$, далее идёт вершина $y_4 = 1$ и $x_4^* = 1$. Это

Этап 2.



Проект	Пр-тие 2	
	c_2	R_2
$x_2 = 1$	2	2
$x_2 = 2$	4	6
$x_2 = 3$	—	—

Этап 1.



Проект	Пр-тие 1	
	c_1	R_1
$x_1 = 1$	1	1
$x_1 = 2$	3	4
$x_1 = 3$	—	—

Рис. 6: Инвестиции $y_1 = 8$ миллионов рублей

всё. Наш путь выделен на графе жирной линией. Итак, ответ:

$$x^* = (1, 1, 3, 1).$$

$$f_1(8) = 13$$

3.2. Решение с помощью таблиц

Решение с помощью таблиц ничем не отличается от решения предыдущей задачи, поэтому объяснения не приводятся. Обратим внимание только на то, что надо предварительно вычислить верхнюю и нижнюю границы состояний y_j.

Этап 4. Предприятие 4.

$$f_4(y_4) = \max_{x_4=1:2 | c_4(x_4) \le y_4} \{R_4(x_4)\}.$$

	$R_4(x_4)$		Оптимальное решение	
y_4	$x_4 = 1$	$x_4 = 2$	$f_4(y_4)$	x_4^*
1	3	—	3	1
2	3	—	3	1
3	3	5	5	2
4	3	5	5	2

Таблица 9: Инвестиции без пустых проектов, этап 4

В последней таблице можно вообще находить только одну строчку, соответствующую состоянию $y_1 = 8$. Но можно для проведения даль-

Этап 3. Предприятия 3 и 4.

$$f_3(y_3) = \max_{x_3=1:3|c_3(x_3)\le y_3}\{R_3(x_3) + f_4(y_3 - c_3(x_3))\}.$$

	$R_3(x_3) + f_4(y_3 - c_3(x_3))$			Оптимальное решение	
y_3	$x_3 = 1$	$x_3 = 2$	$x_3 = 3$	$f_3(y_3)$	x_3^*
2	1+3=4	—	—	4	1
3	1+3=4	3+3=6	—	6	2
4	1+5=6	3+3=6	—	6	1,2
5	1+5=6	3+5=8	7+3=10	10	3

Таблица 10: Инвестиции без пустых проектов, этап 3

Этап 2. Предприятия 2, 3 и 4.

$$f_2(y_2) = \max_{x_2=1:3|c_2(x_2)\le y_2}\{R_2(x_2) + f_3(y_2 - c_2(x_2))\}.$$

	$R_2(x_2) + f_3(y_2 - c_2(x_2))$		Оптимальное решение	
y_2	$x_2 = 1$	$x_2 = 2$	$f_2(y_2)$	x_2^*
4	2+ 4=6	—	6	1
5	2+ 6=8	—	8	1
6	2+ 6=8	6+4=10	10	2
7	2+10=12	6+6=12	12	1, 2

Таблица 11: Инвестиции без пустых проектов, этап 2

Этап 1. Предприятия 1, 2, 3 и 4.

$$f_1(y_1) = \max_{x_1=1:2|c_1(x_1)\le y_1}\{R_1(x_1) + f_2(y_1 - c_1(x_1))\}.$$

	$R_1(x_1) + f_2(y_1 - c_1(x_1))$		Оптимальное решение	
y_1	$x_1 = 1$	$x_1 = 2$	$f_1(y_1)$	x_1^*
5	1+ 6= 7	—	7	1
6	1+ 8= 9	—	9	1
7	1+10=11	4+6=10	11	1
8	1+12=13	4+8=12	13	1

Таблица 12: Инвестиции без пустых проектов, этап 1

нейшего анализа найти еще несколько строчек. Самая верхняя соответствует затратам, необходимым для реализации варианта $x = (1, 1, 1, 1)$. Для этого необходимо затратить $\sum_j c_j(1) = 1 + 2 + 1 + 1 = 5$ миллионов рублей.

Итак, запишем один из ответов нашей задачи. Для того, чтобы оптимальным образом потратить 8 миллионов рублей на 4 предприятиях, надо выбирать проекты

$$x^* = (1, 1, 3, 1).$$

$$f_1(8) = 13$$

миллионов рублей в год при этом составит максимальный годовой доход.

3.3. Задача распределения инвестиций на компьютере.

Решаем задачу динамического программирования с помощью пакета `SciLab-5.3`. Программа называется `InvestDi.sce`. Чтобы ее напечатать на экране, надо набрать `write(6, mgetl('InvestDi.sce'))`, чтобы запустить на выполнение, набирается текст `exec('InvestDi.sce',0)`. Параметр 0 говорит о том, что не надо при выполнении программы печатать текст про-

граммы. Из протокола сессии работы с пакетом Scilab хорошо видно, как это все выполняется. Если какой-либо оператор непонятен, надо набрать слово help и имя непонятного оператора. Scilab вас просветит на английском языке (или на родном для авторов пакета французском!).

```
-->write(6, mgetl("InvestDi.sce"))

// Программа InvestDi.sce
// Решение задачи распределения инвестиций
// методом динамического программирования.
// ==== 23 марта 2011 года ==== Визгунов НП.

clear, clc, mode(0), lines(0,90)

y_1 = 5;
C = [
  0    0    0    0
  1    3    1    2
-%inf  5    2  -%inf ];

R = [
  0    0    0    0
  3    5    4    3
-%inf  9    6  -%inf ];

[x9,n] = size(C);    // x9 - Количество управлений
y9     = y_1 + 1;    // y9 - Количество состояний

// Вычисляем таблицы для
// динамического программирования - ДП

X = -%inf * ones(y9, n+1);
F = X;
F(:,n+1) = zeros(y9, 1); // что находится сейчас в F ?

for j = n : -1 :1
   Fyx = -%inf * ones(y9, x9);
   for y = 0 : y_1          // y - состояние  на шаге j
      for x = 1 : x9        // x - управление на шаге j
         yy = y - C(x,j);
         if 0 <= yy & yy <= y_1
            // Уравнение Ричарда Беллмана
            Fyx(1+y,x) = R(x,j) + F(1+y-C(x,j),j+1);
            // Fyx(1+ y,x) = R(x,j) + F(1+ yy, j+1);
         end
      end
   end
   // F,Fyx

   [cFy, cX] = max(Fyx, 'c');  // c - column, столбец
   F(:,j) = cFy;
   X(:,j) = cX;

   // Печать шапки

   xmaxi = sum(bool2s(R(:,j) > -%inf));
   write(6, ' ')
   write(6, '       Этап ' + string(j)' )
   // tire40 =  '~~~~~~~~~~~~~~~~~~~~|~~~~~~~~~~';
   tire40 = part('~', ones(1,20)) + '|' + ...
      part('~', ones(1, 6*xmaxi ));

   write(6, tire40);
   write(6, 'Y' + string(j) + 'Fj(Yj) Xj* | Fyx');
   write(6, tire40);

   // Печать только нужных строчек и столбцов
```

```
    // Допустимые состояния на этапе j меняются
    // в пределах от ymini до ymaxi включительно

    ymini = sum(C(1, j:$));
    ymaxi = y_1 - sum(C(1, 1: j-1));

    Table = [(0 : y_1)', F(:,j), X(:,j), Fyx];

    write(6,Table(1+(ymini:ymaxi),1:(3+xmaxi)), ...
        '(␣3(i4,2x),␣""␣␣|␣␣"",␣100(i4,2x)␣)' )
    write(6, tire40);
end

// Вычисление fopt, rXopt и rYopt
// ==============================

rXopt = zeros(1, n);  // r - row, строка
rYopt = zeros(1, n);
y = y_1;

for j = 1 : n
    rXopt(j) = X(1+ y, j);
    rYopt(j) = y;
    y = y - C(rXopt(j), j);
end

// Печать ответа:
// ==============

C, R, y_1

write(6,'␣==========␣Ответ:␣==========')
write(6,'␣==============================')

fopt = F(1+y_1, 1);
write(6,'␣fopt␣=␣␣' + string(fopt));
write(6, rXopt, '(␣""␣rXopt␣="",␣100i6␣)' )
write(6, rYopt, '(␣""␣rYopt␣="",␣100i6␣)' )

-->exec("InvestDi.sce",0)
```

Этап 4

Y4	Fj(Yj)	Xj*		Fyx	
0	0	1		0	****
1	0	1		0	****
2	3	2		0	3
3	3	2		0	3
4	3	2		0	3
5	3	2		0	3

Этап 3

Y3	Fj(Yj)	Xj*		Fyx		
0	0	1		0	****	****
1	4	2		0	4	****
2	6	3		3	4	6
3	7	2		3	7	6
4	9	3		3	7	9
5	9	3		3	7	9

Этап 2

```
~~~~~~~~~~~~~~~~~~~~|~~~~~~~~~~~~~~~~~~
  Y2    Fj(Yj)   Xj*   |       Fyx
~~~~~~~~~~~~~~~~~~~~|~~~~~~~~~~~~~~~~~~
   0       0       1    |       0   ****   ****
   1       4       1    |       4   ****   ****
   2       6       1    |       6   ****   ****
   3       7       1    |       7      5   ****
   4       9       1    |       9      9   ****
   5      11       2    |       9     11      9
~~~~~~~~~~~~~~~~~~~~|~~~~~~~~~~~~~~~~~~

         Этап  1
~~~~~~~~~~~~~~~~~~~~|~~~~~~~~~~~~
  Y1    Fj(Yj)   Xj*   |       Fyx
~~~~~~~~~~~~~~~~~~~~|~~~~~~~~~~~~
   0       0       1    |       0   ****
   1       4       1    |       4      3
   2       7       2    |       6      7
   3       9       2    |       7      9
   4      10       2    |       9     10
   5      12       2    |      11     12
~~~~~~~~~~~~~~~~~~~~|~~~~~~~~~~~~
 C  =

    0.     0.     0.     0.
    1.     3.     1.     2.
 - Inf     5.     2.  - Inf
 R  =

    0.     0.     0.     0.
    3.     5.     4.     3.
 - Inf     9.     6.  - Inf
 y_1  =

    5.
 ========== Ответ: ==========
 ================================
 fopt =  12
 rXopt =      2      1      3      2
 rYopt =      5      4      4      2

-->diary(0)
```

Как мы видим, решение, которые выдает наша программа, полностью совпадает с результатом, который получен вручную. Хотя результат и получен в «старинном стиле», в командном окне, но таблицы компьютер вычисляет верно и это делает достаточно простая программа. Единственный существенный недостаток этой программы — она дает один ответ. Чтобы получить все ответы, предложенную программу динамического программирования пришлось бы сильно усложнить.

Для получения всех оптимальных решений, нам проще написать другую программу, а именно, программу полного перебора. Она очень хорошо справляется с небольшими задачами и проверяет наше умение решать задачи методом динамического программирования. К тому же, теперь мы всегда будем знать все оптимальные решения. В частности, для рассмотренной задачи о распределении инвестиций из распечатки видно, что у задачи есть еще одно опмимальное решение

$$x^* = (2, 2, 2, 1).$$

Раз решение оптимальное, то доход также равен $f_1(5) = 12$ миллионов рублей в год.

```
-->write(6, mgetl(nameF))

// Программа InvestPe.sce
// ==== 23 марта 2011 года ==== Визгунов НП..
```

```
mode(0), lines(0,80), clc, clear
y1 = 5
C = [    0     0     0     0
         1     3     1     2
       -%inf   5     2   -%inf ]
R = [    0     0     0     0
         3     5     4     3
       -%inf   9     6   -%inf ]

[u9,n] = size(C);    // u9 - число возможных управлений

x_min = ones(1,n)                    // [ 1   1   1   1 ]
x_max = sum(abs(R) ~= %inf, 'r')     // [ 2   3   3   2 ]

x     = x_min;
x_opt = x_min;
f_opt = -%inf;

j = n;
while %t
   if  x(j) <= x_max(j)
       cost = 0;
       for i = 1 :n
          cost = cost + C(x(i),i);
       end
       if cost <= y1
          f = 0;
          for i = 1:n
             f = f + R(x(i),i);
          end
          if  f_opt < f
             // Для печати данных убрать ;
             f_opt = f;
             x_opt = x;
          elseif f_opt == f
             f_opt;
             x_opt = [x_opt; x ];
          end
       end // cost
       j = n;
   else     // x(j) > x_max(j)
       x(j) = x_min(j);
       j = j - 1;
       if j <= 1e-8
          // Решение получено - выйти из цикла
          break
       end
   end // if
   x(j) = x(j) + 1;
end // while 1

write(6,['␣*********'
         '␣␣␣Ответ␣␣'
         '␣*********'])
write(6, '␣f_opt␣=␣' + string(f_opt))
   // печать в одну строку
x_opt

-->exec(nameF,0)
 y1  =
    5.
 C  =
    0.     0.     0.     0.
    1.     3.     1.     2.
  - Inf    5.     2.   - Inf
 R  =
```

```
    0.    0.    0.    0.
    3.    5.    4.    3.
 - Inf    9.    6. - Inf
 x_min  =
    1.    1.    1.    1.
 x_max  =
    2.    3.    3.    2.
 *********
   Ответ
 *********
 f_opt = 12
 x_opt  =
    2.    1.    3.    2.
    2.    2.    2.    1.
-->diary(0)
```

4. Задача о загрузке(о рюкзаке или о ранце)

Задача 2 (о рюкзаке) *Самолет загружается предметами n различных типов. Каждый предмет типа j дает доход c_j тысяч рублей и весит a_j тонн. Грузоподъемность самолета – b тонн.*

Выбрать предметы, погрузка которых позволит получить максимальный доход без превышения грузоподъемности самолета.

Сначала рассмотрим задачу в общей постановке.

$$f_1(y_1) = \max_{x_1,\dots,x_n} \left\{ \sum_{j=1}^{n} c_j x_j \right\},$$
$$\sum_{j=1}^{n} a_j x_j \le y_1,$$
$$x_j \ge 0 - \text{целые}, j \in 1:n,$$
$$y_1 = b.$$

Легко вывести рекуррентные уравнения Беллмана для процедуры обратной прогонки.

$$f_n(y_n) = \max_{x_n=0,1,\dots,[b/a_n]} \{c_n x_n\},$$
$$f_j(y_j) = \max_{x_j=0,1,\dots,[b/a_j]} \{c_j x_j + f_{j+1}(y_j - a_j x_j)\},$$
$$j = n-1,\dots,2,1$$

Здесь квадратные скобки в выражении $[b/a_j]$ используются для обозначения целой части числа b/a_j, стоящего в этих скобках.

Другие обозначения:

c_j – доход, получаемый от перевозки одного предмета типа j,

a_j – вес этого предмета,

x_j – количество предметов j-го типа (управление),

y_j – часть грузоподъемности самолета, выделенная для предметов $j, j+1,\dots,n$ (состояние),

$f_j(y_j)$ – максимальный доход от погрузки предметов $j, j+1,\dots,n$, если в самолете выделено y_j тонн под эти предметы.

Решим следующую конкретную задачу с помощью этого рекуррентного уравнения.

$$f_1(y_1) = \max_{x_1,x_2,x_3} \{65x_1 + 80x_2 + 30x_3\},$$
$$2x_1 + 3x_2 + 1x_3 \le y_1,$$
$$x_j - \text{целые}, y_1 = 5.$$

В таблицах 13 – 15 выполнены все вычисления, необходимые для получения оптимального решения.

При заданном $y_1 = 5$ оптимальным решением является $x^* = (2,0,1)$, а суммарная стоимость груза равна 160.

Этап 3. Предметы 3 типа.

$$f_3(y_3) = \max_{x_3}\{30x_3\},\ \max x_3 = [5/1] = 5$$

	$30x_3$						Оптимальное решение	
y_3	x_3=0	x_3=1	x_3=2	x_3=3	x_3=4	x_3=5	$f_3(y_3)$	x_3^*
0	0	–	–	–	–	–	0	0
1	0	30	–	–	–	–	30	1
2	0	30	60	–	–	–	60	2
3	0	30	60	90	–	–	90	3
4	0	30	60	90	120	–	120	4
5	0	30	60	90	120	150	150	5

Таблица 13: Задача о рюкзаке, этап 3

Этап 2. Предметы 2 и 3 типа.

$$f_2(y_2) = \max_{x_2}\{80x_2 + f_3(y_2 - 3x_2)\},\ \max x_2 = [5/3] = 1$$

	$80x_2 + f_3(y_2 - 3x_2)$		Оптимальное решение	
y_2	$x_2 = 0$	$x_2 = 1$	$f_2(y_2)$	x_2^*
0	0+ 0= 0	–	0	0
1	0+ 30=30	–	30	0
2	0+ 60=60	–	60	0
3	0+ 90=90	80+0 = 80	90	0
4	0+120=120	80+30=110	120	0
5	0+150=150	80+60=140	150	0

Таблица 14: Задача о рюкзаке, этап 2

Этап 1. Предметы 1, 2 и 3 типа.

$$f_1(y_1) = \max_{x_1}\{65x_1 + f_2(y_1 - 2x_1)\},\ \max x_1 = [5/2] = 2$$

	$65x_1 + f_2(y_1 - 2x_1)$			Оптимальное решение	
y_1	$x_1 = 0$	$x_1 = 1$	$x_1 = 2$	$f_1(y_1)$	x_1^*
0	0+ 0= 0	–	–	0	0
1	0+ 30= 30	–	–	30	0
2	0+ 60= 60	65+ 0= 65	–	65	1
3	0+ 90= 90	65+30= 95	–	95	1
4	0+120=120	65+60=125	130+ 0=130	130	2
5	0+150=150	65+90=155	130+30=160	160	2

Таблица 15: Задача о рюкзаке, этап 1

Заметим, что на этапе 1 достаточно построить только одну строку таблицы для значения $y_1 = 5$. Однако, располагая полной таблицей, можно провести анализ чувствительности решения, то есть посмотреть уменьшение целевой функции при уменьшении грузоподъемности самолета. Вычислительная схема динамического программирования автоматически обеспечивает проведение анализа модели на чувствительность.

Легко решить задачу о рюкзаке и графически. Ничего нового здесь нет, поэтому предлагается сделать это самостоятельно.

4.1. Задача о рюкзаке на компьютере.

Для решения задачи о рюкзаке можно пойти по лёгкому пути, а именно, можно свести задачу к задаче о распределении инвестиций. С точки зрения

математика, задача о рюкзаке — это частный случай задачи об инвестициях с линейной целевой функцией и линейным ограничением. Другими словами, надо построить матрицы C и R, которые фигурировали в предыдущей задаче. В программе это сделано даже двумя способами. Первый из них простой, правда, медленный и громоздкий, а второй, в стиле `SciLab` – очень быстрый и компактный.

```
// function RiukDiCR(c, a, b )
// Решение задачи о загрузке самолета методом ДП.
// ==== 30 марта 2011 года Визгунов===== НП..======

clc, clear, mode(0), lines(0)

c = [ 65  80  30 ];  // доход от перевозки предмета
a = [  2   3   1 ];  // вес этого предмета
b = 5; y1 = b;       // грузоподъемность самолета

n  = length(a);
s9 = y1 + 1;                    // s9 – Кол. состояний
u9 = 1 + floor(b / min(a));     // u9 – Кол. управлений

// 1) Очевидный способ вычисления C и R

C = -%inf * ones(u9, n);
R = C;
for u = 1 : u9          // (u-1) – это количество предметов
   for j = 1 : n
      temp = a(j) * (u -1);  // j – номер предмета
      if temp <= b    // если предметы помещаются в самолет
         C(u, j) = temp;
         R(u, j) = c(j) * (u -1);
      end
   end
end
C; R;

// 2) Вычисления C и R в духе SciLab

C = (0 : u9 -1)' * a
I = find(C > y1)
C(I) = -%inf
R = (0 : u9 -1)' * c;
R(I) = -%inf

// Теперь – почти задача о распределении инвестиций

X = -%inf * ones(s9, n+1);
F = X;
F(:, n + 1) = zeros(s9, 1);

for j = n : -1 : 1
   Fsu = -%inf * ones(s9, u9);
   for s = 1 : s9           // Yj = s -1
      for u = 1 : u9        // Xj = u
         ss = s - C(u,j);
         if 1 <= ss & ss <= s9
            Fsu(s, u) = R(u, j) + F(ss, j +1);
         end
      end
   end
   [fs, index] = max(Fsu, 'c');
   F(:,j) = fs;
   X(:,j) = index - 1;

   // Печать шапки и таблицы
   // ======================
```

```
    u7 = 1 + floor(b / min(a(j)));
    line50 = part('-', ones(1, 22)) + '|' + ...
    part('-', ones(1, 3 + 6 * u7));

    // line50 = code2str(str2code('-') * ones(1,58));

    write(6, '␣')
    write(6, 'Этап␣' + string(j))
    write(6, line50)     // напечатать 50 минусов
    write(6, '␣␣␣␣Yj␣␣␣␣Fj(Yj)␣␣Xj*␣␣|␣␣␣␣␣␣␣␣Fsu');
    write(6, line50)

    Temp =[(0 : y1)', fs, (index - 1), Fsu];
    disp(Temp(:, 1 : 3 + u7))

    write(6, line50)
end

// Вычислить и напечать ответ
// ==========================

x_opt = zeros(1, n);
s = s9;
for j = 1 : n
    x_opt(j) = X(s, j);
    s = s - a(j) * x_opt(j);
end
c, a, x_opt
write(6,'␣f_opt␣=␣' + string(F(s9, 1)) + ...
        '␣␣y1␣=␣'   + string(y1))


exec('RiukDiCR.sce', 0)

 C  =
      0.      0.      0.
      2.      3.      1.
      4.      6.      2.
      6.      9.      3.
      8.     12.      4.
     10.     15.      5.
 I  =
      4.      5.      6.      9.      10.      11.      12.
 C  =
      0.      0.      0.
      2.      3.      1.
      4.   -Inf      2.
   -Inf    -Inf      3.
   -Inf    -Inf      4.
   -Inf    -Inf      5.
 R  =
      0.      0.      0.
     65.     80.     30.
    130.   -Inf     60.
   -Inf    -Inf     90.
   -Inf    -Inf    120.
   -Inf    -Inf    150.

  Этап 3
-------------------------------------------------------------
  Yj    Fj(Yj)   Xj*   |                 Fsu
-------------------------------------------------------------
  0.      0.       0.     0.   -Inf   -Inf   -Inf   -Inf   -Inf
  1.     30.       1.     0.    30.   -Inf   -Inf   -Inf   -Inf
  2.     60.       2.     0.    30.    60.   -Inf   -Inf   -Inf
  3.     90.       3.     0.    30.    60.    90.   -Inf   -Inf
  4.    120.       4.     0.    30.    60.    90.    120.  -Inf
```

```
   5.     150.     5.     0.     30.     60.     90.    120.    150.
 ------------------------------------------------------------------

   Этап 2
 ------------------------------------
  Yj    Fj(Yj)   Xj*   |      Fsu
 ------------------------------------
   0.      0.      0.      0.     -Inf
   1.     30.      0.     30.     -Inf
   2.     60.      0.     60.     -Inf
   3.     90.      0.     90.      80.
   4.    120.      0.    120.     110.
   5.    150.      0.    150.     140.
 ------------------------------------

   Этап 1
 ------------------------------------------------
   Yj    Fj(Yj)   Xj*   |          Fsu
 ------------------------------------------------
   0.      0.      0.      0.     -Inf     -Inf
   1.     30.      0.     30.     -Inf     -Inf
   2.     65.      1.     60.      65.     -Inf
   3.     95.      1.     90.      95.     -Inf
   4.    130.      2.    120.     125.      130.
   5.    160.      2.    150.     155.      160.
 ------------------------------------------------
 c  =
    65.    80.    30.
 a  =
    2.     3.     1.
 x_opt  =
    2.     0.     1.
 f_opt = 160   y1 = 5
```

Но лучше все-таки обойтись без вычисления матриц C и R, это позволяет сделать программу динамического программирования более простой и компактной, к тому же такая программа позволит решать задачи большего размера.

```
// function RiukDi(c, a, b )
// Программа RiukDi.sce. Нет вычислений C и R !!!

// Решение задачи о загрузке самолета методом ДП.
// ===== 30 марта 2011 года ===== Визгунов НП.

clc, clear, lines(0, 100), mode(0)

//доходы =
c = [ 65   80   30 ];
//вес предметов =
a = [  2    3    1 ];
//грузоподъемность самолета =
b = 5; y1 = b;

n  = length(a);
s9 = y1 + 1;                    // s9 - Кол. состояний
u9 = 1 + floor(b / min(a));     // u9 - Кол. управлений

X = -%inf * ones(s9, n+1);
F = X;
F(:, n + 1) = zeros(s9, 1);

for j = n : -1 : 1
   Fsu = -%inf * ones(s9, u9);
   for s = 1 : s9                    // Yj = s -1
      for u = 1 : u9                 // Xj = u
         ss = s - a(j) * (u -1); // - C(u,j) было
         if 1 <= ss & ss <= s9
```

```
            Fsu(s, u) = c(j) * (u - 1) + F(ss, j + 1);
          end
        end
    end
    [fs, index] = max(Fsu, 'c');
    F(:,j) = fs;
    X(:,j) = index - 1;

    // Печать шапки и таблицы
    // ======================

    u7 = 1 + floor(b / min(a(j)));
    line50 = part('-', ones(1, 22)) + '|' + ...
    part('-', ones(1, 3 + 6 * u7));

    // line50 = code2str(str2code('-') * ones(1,58));

    write(6, ' ')
    write(6, 'Этап ' + string(j))
    write(6, line50)     // напечатать 50 минусов
    write(6, '    Yj    Fj(Yj)  Xj*  |    Fsu');
    write(6, line50)

    Temp =[(0 : y1)', fs, index-1, Fsu];
    disp(Temp(:, 1: 3+ u7))

    write(6, line50)
end

// Вычислить и напечатать ответ
// ============================

x_opt = zeros(1, n);
s = s9;
for j = 1 : n
    x_opt(j) = X(s, j);
    s = s - a(j) * x_opt(j);
end
c, a, x_opt
write(6,' f_opt = ' + string(F(s9,1)) + ...
        '   y1= '   + string(y1))
//endfunction

exec('RiukDi.sce',0)
```

Этап 3

Yj	Fj(Yj)	Xj*	Fsu				
0.	0.	0.	0.	-Inf	-Inf	-Inf	-Inf
1.	30.	1.	0.	30.	-Inf	-Inf	-Inf
2.	60.	2.	0.	30.	60.	-Inf	-Inf
3.	90.	3.	0.	30.	60.	90.	-Inf
4.	120.	4.	0.	30.	60.	90.	120.
5.	150.	5.	0.	30.	60.	90.	120.

Этап 2

Yj	Fj(Yj)	Xj*	su
0.	0.	0.	0.
1.	30.	0.	30.
2.	60.	0.	60.
3.	90.	0.	90.
4.	120.	0.	120.
5.	150.	0.	150.

```
————————————————————┼————————————

  Этап 1
————————————————————┼————————————————————
  Yj    Fj(Yj)   Xj*   |          Fsu
————————————————————┼————————————————————
  0.      0.       0.     0.    -Inf    -Inf
  1.      30.      0.     30.   -Inf    -Inf
  2.      65.      1.     60.    65.    -Inf
  3.      95.      1.     90.    95.    -Inf
  4.      130.     2.     120.   125.    130.
  5.      160.     2.     150.   155.    160.
————————————————————┼————————————————————
 c  =
    65.    80.    30.
 a  =
    2.     3.     1.
 x_opt  =
    2.     0.     1.
 f_opt = 160   y1 = 5
```

В распечатке представлена также программа полного перебора. Эта программа решает всё ту же задачу. Убеждаемся, что она имеет одно оптимальное решение. Все промежуточные шаги поиска этого решения распечатаны, чтобы алгоритм был более прозрачен.

```
-->write(6, mgetl('RiukPe.sce'))
// function RiukPe(c, a, b)
// программа RjukPe.sce  Лабораторная номер 3
// 30 марта 2011 года ===== Визгунов НП.

clc, clear, lines(0), mode(0)

c = [ 65   80   30 ]
a = [  2    3    1 ]
b = 5,  y1 = b;

n  = length(a);
s9 = y1 + 1;                // s9 – Кол. состояний
u9 = 1 + floor(b / min(a));
// u9 = 1 + floor(max(b ./ a));

x_min = zeros(1, n);    //  [0   0   0 ]
x = x_min;
x_max = floor(b ./ a); //  [2   1   5 ]

f_opt = -%inf;
j = n;
while %t
   if  x(j) <= x_max(j)

      if a * x' <= b  // если предметы войдут в самолет
         f = c * x';
         if f_opt < f
            disp('f_opt < f <<<<<<<<<<<<<<')
            f_opt = f
            x_opt = x
            //disp(' ')
         elseif  f == f_opt
            disp('f_opt == f ============')
            f_opt
            x_opt = [x_opt; x]
         end
      end

      j = n;
   else     // x(j) > x_max(j)
      x(j) = x_min(j);
```

```
        j = j - 1;

        // закончить бесконечный цикл
        if j <= 1e-5
           break
        end

    end // if
    x(j) = x(j) + 1;
end // while %t

write(6,' =====')
write(6,' ответ')
write(6,' =====')
x_opt, f_opt

//endfunction

-->exec('RiukPe.sce',0)
 c  =
     65.    80.    30.
 a  =
     2.     3.     1.
 b  =
     5.
 f_opt < f <<<<<<<<<<<<<
 f_opt  =
     0.
 x_opt  =
     0.     0.     0.
 f_opt < f <<<<<<<<<<<<<
 f_opt  =
     30.
 x_opt  =
     0.     0.     1.
 f_opt < f <<<<<<<<<<<<<
 f_opt  =
    60.
 x_opt  =
    0.     0.     2.
 f_opt < f <<<<<<<<<<<<<
 f_opt  =
    90.
 x_opt  =
    0.     0.     3.
 f_opt < f <<<<<<<<<<<<<
 f_opt  =
    120.
 x_opt  =
    0.     0.     4.
 f_opt < f <<<<<<<<<<<<<
 f_opt  =
    150.
 x_opt  =
    0.     0.     5.
 f_opt < f <<<<<<<<<<<<<
 f_opt  =
    155.
 x_opt  =
    1.     0.     3.
 f_opt < f <<<<<<<<<<<<<
 f_opt  =
    160.
 x_opt  =
    2.     0.     1.
 =====
 ответ
```

```
 =====
 x_opt  =
    2.     0.     1.
 f_opt  =
    160.

-->diary(0)
```

5. Задача о надежности

Задача 3 (о надежности) *Конструируется электронный прибор, состоящий из трех основных компонентов. Все компоненты соединены последовательно, поэтому выход из строя одной из них приводит к отказу всего прибора. Надежность прибора можно повысить путем дублирования каждой компоненты. Каждая компонента может состоять из одного, двух или трех блоков. Общая стоимость прибора не должна превышать 10 тысяч евро. В таблице 16 приведены данные о стоимости $c_j(x_j)$ и надежности $R_j(x_j)$ j-ой компоненты, состоящей из x_j блоков.*

Требуется определить количество блоков x_j в компоненте j, при котором надежность прибора максимальна, а стоимость не превышает заданной величины.

x_j	$j=1$		$j=2$		$j=3$	
	C_1	R_1	C_2	R_2	C_3	R_3
1	1	0.6	$\underline{3}$	$\underline{0.7}$	2	0.5
2	$\underline{2}$	$\underline{0.8}$	5	0.8	4	0.7
3	3	0.9	6	0.9	$\underline{5}$	$\underline{0.9}$

Таблица 16: Данные о стоимости и надежности каждой компоненты прибора.

Сначала рассмотрим задачу в общей постановке.

$$f_1(y_1) = \max_{x_1,\ldots,x_n} \left\{ \prod_{j=1}^{n} R_j(x_j) \right\},$$
$$\sum_{j=1}^{n} c_j(x_j) \le y_1,$$
$$x_j \ge 1 - \text{целые}, \; j = 1:n.$$

Легко вывести рекуррентные уравнения Беллмана для процедуры обратной прогонки.

$$f_n(y_n) = \max_{x_n | c_n(x_n) \le y_n} \{R_n(x_n)\},$$
$$f_j(y_j) = \max_{x_j | c_j(x_j) \le y_j} \{R_j(x_j) * f_{j+1}(y_j - c_j(x_j))\},$$
$$j = n-1, n-2, \ldots 1.$$

Этап 3. Третья компонента прибора.

$$f_3(y_3) = \max_{x_3} \{R_3(X_3)\}.$$

	$R_3(x_3)$			Оптимальное решение	
y_3	$x_3=1$	$x_3=2$	$x_3=3$	$f_3(y_3)$	x_3^*
2	0.5	–	–	0.5	1
3	0.5	–	–	0.5	1
4	0.5	0.7	–	0.7	2
$\underline{5}$	0.5	0.7	0.9	$\underline{0.9}$	$\underline{3}$
6	0.5	0.7	0.9	0.9	3

Таблица 17: Задача о надежности, этап 3

Этап 2. Вторая и третья компоненты прибора.

$$f_2(y_2) = \max_{x_2=1:3}\{R_2(x_2) * f_3(y_2 - c_2(x_2))\},$$

	$R_2(x_2) * f_3(y_2 - c_2(x_2))$			Оптимальное решение	
y_2	$x_2 = 1$	$x_2 = 2$	$x_2 = 3$	$f_2(y_2)$	x_2^*
5	.7*.5=.35	–	–	.35	1
6	.7*.5=.35	–	–	.35	1
7	.7*.7=.49	.8*.5=.40	–	.49	1
8	.7*.9=.63	.8*.5=.40	.9*.5=.45	.63	1
9	.7*.9=.63	.8*.7=.56	.9*.5=.45	.63	1

Таблица 18: Задача о надежности, этап 2

Этап 1. Первая, вторая и третья компоненты прибора.

$$f_1(y_1) = \max_{x_1=1:3}\{R_1(x_1) * f_2(y_1 - c_1(x_1))\},$$

	$R_1(x_1) * f_2(y_1 - c_1(x_1))$			Оптим. решение	
y_1	$x_1 = 0$	$x_1 = 1$	$x_1 = 2$	$f_1(y_1)$	x_1^*
6	.6*.35=.210	–	–	.210	1
7	.6*.35=.210	.8*.35=.280	–	.280	2
8	.6*.49=.294	.8*.35=.280	.9*.35=.315	.315	3
9	.6*.63=.378	.8*.49=.392	.9*.35=.315	.392	2
10	.6*.63=.378	.8*.63=.504	.9*.49=.441	.504	2

Таблица 19: Задача о надежности, этап 1

Как видим на этом примере, методом динамического программирования можно решать задачи не только с аддитивной целевой функцией, но и с мультипликативной. В остальном решение задачи о надежности ничем не отличается от задачи о распределении инвестиций без пустых проектов.

5.1. Задача о надёжности на компьютере.

Отличие от общей задачи о распределении инвестиций минимально. Надо только не забыть, что здесь надо для целевой функции сложение заменить на умножение. А также вспомнить, что $0 + x = x$ для сложения, а для умножения ноль заменяется на единицу: $1 * x = x$.

```
-->file_ = 'NadDi.sce';
-->write(6, mgetl(file_))

// Программа NadDi.sce
// Решение задачи о надёжности
// методом динамического программирования.
// ==== 24 марта 2011 года ==== Визгунов НП..

clear, clc, mode(0), lines(0,90)

y_1 = 10
C = [ 1    3    2
      2    5    4
      3    6    5  ]
R = [ 0.6  0.7  0.5
      0.8  0.8  0.7
      0.9  0.9  0.9]

[x9, n] = size(R);    // x9          - Количество управлений
y9      = y_1 + 1;    // y9=y_1 +1 - Количество состояний
```

```
// Вычисляем таблицы для
// динамического программирования – ДП

X = –%inf * ones(y9, n+1);
F = X;
F(:,n+1) = ones(y9, 1); // что находится сейчас в F ?

for j = n : –1 :1
   Fyx = –%inf * ones(y9, x9);
   for y = 0 : y_1          // y – состояние  на шаге j
      for x = 1 : x9        // x – управление на шаге j
         yy = y – C(x,j);
         if 0 <= yy & yy <= y_1
            // Уравнение Ричарда Беллмана
            Fyx(1+y, x) = R(x, j) * F(1+y–C(x,j), j+1);
            // Fyx(1+ y,x) = R(x,j) + F(1+ yy, j+1);
         end
      end
   end
   // F,Fyx

   [cFy, cX] = max(Fyx, 'c');  // c – column, столбец
   F(:,j) = cFy;
   X(:,j) = cX;

   // Печать шапки

   xmaxi = sum(bool2s(R(:,j) > –%inf));
   write(6, ' ')
   write(6, '       Этап ' + string(j)' )
   // tire40 =  '~~~~~~~~~~~~~~~~~~~~|~~~~~~~~~~~~~~~';
   tire40 = part('~', ones(1, 21)) + '|' + ...
   part('~', ones(1, 7 * xmaxi + 2 ));

   write(6, tire40);
   write(6, '   Y' + string(j) + ...
      '   Fj(Yj)  Xj*  |      Fyx ');
   write(6, tire40);

   // Печать только нужных строчек и столбцов таблицы

   // Допустимые состояния на этапе j меняются
   // в пределах от ymini до ymaxi включительно

   ymini = sum(C(1, j:$));
   ymaxi = y_1 – sum(C(1, 1: j–1));

   Table = [(0 : y_1)', F(:,j), X(:,j), Fyx];

   write(6, Table(1 + (ymini :ymaxi), 1:(3 +xmaxi)), ...
      '( f6.0, f7.3, f6.0 ), ""  |  "", 100(f7.3) )' )
   write(6, tire40);
end

// Вычисление fopt, rXopt и rYopt
// ==============================

rXopt = zeros(1, n);  // r – row, строка
rYopt = zeros(1, n);
y = y_1;

for j = 1 : n
   rXopt(j) = X(1+ y, j);
   rYopt(j) = y;
   y = y – C(rXopt(j), j);
end
```

```
// Печать ответа:
// ==============

C, R, y_1

write(6,'=========== Ответ: ===========')
write(6,'==============================')

fopt = F(1 + y_1, 1);
write(6,'fopt = ' + string(fopt));
write(6, rXopt, '( "rXopt= ",100i6 )' )
write(6, rYopt, '( "rYopt= ",100i6 )' )


-->exec(file_,0)
 y_1  =
    10.
 C  =
    1.    3.    2.
    2.    5.    4.
    3.    6.    5.
 R  =
    0.6    0.7    0.5
    0.8    0.8    0.7
    0.9    0.9    0.9

        Этап 3
~~~~~~~~~~~~~~~~~~~~|~~~~~~~~~~~~~~~~~~~~~~
   Y3    Fj(Yj)  Xj*  |      Fyx
~~~~~~~~~~~~~~~~~~~~|~~~~~~~~~~~~~~~~~~~~~~
    2.   0.500    1.  |    0.500   -Inf   -Inf
    3.   0.500    1.  |    0.500   -Inf   -Inf
    4.   0.700    2.  |    0.500   0.700  -Inf
    5.   0.900    3.  |    0.500   0.700  0.900
    6.   0.900    3.  |    0.500   0.700  0.900
~~~~~~~~~~~~~~~~~~~~|~~~~~~~~~~~~~~~~~~~~~~

        Этап 2
~~~~~~~~~~~~~~~~~~~~|~~~~~~~~~~~~~~~~~~~~~~
   Y2    Fj(Yj)  Xj*  |      Fyx
~~~~~~~~~~~~~~~~~~~~|~~~~~~~~~~~~~~~~~~~~~~
    5.   0.350    1.  |    0.350   -Inf   -Inf
    6.   0.350    1.  |    0.350   -Inf   -Inf
    7.   0.490    1.  |    0.490   0.400  -Inf
    8.   0.630    1.  |    0.630   0.400  0.450
    9.   0.630    1.  |    0.630   0.560  0.450
~~~~~~~~~~~~~~~~~~~~|~~~~~~~~~~~~~~~~~~~~~~

        Этап 1
~~~~~~~~~~~~~~~~~~~~|~~~~~~~~~~~~~~~~~~~~~~
   Y1    Fj(Yj)  Xj*  |      Fyx
~~~~~~~~~~~~~~~~~~~~|~~~~~~~~~~~~~~~~~~~~~~
    6.   0.210    1.  |    0.210   -Inf   -Inf
    7.   0.280    2.  |    0.210   0.280  -Inf
    8.   0.315    3.  |    0.294   0.280  0.315
    9.   0.392    2.  |    0.378   0.392  0.315
   10.   0.504    2.  |    0.378   0.504  0.441
~~~~~~~~~~~~~~~~~~~~|~~~~~~~~~~~~~~~~~~~~~~
 C  =
    1.    3.    2.
    2.    5.    4.
    3.    6.    5.
 R  =
    0.6    0.7    0.5
    0.8    0.8    0.7
    0.9    0.9    0.9
```

```
 y_1  =
    10.
 ========== Ответ: ===========
 ==============================
 fopt =  0.504
 rXopt =     2      1      3
 rYopt =    10      8      5

-->diary(0)
```

В распечатке представлена также программа полного перебора. Результаты выполнения программы подсказывают нам, что решение только одно.

```
-->file_ = 'NadPe.sce';
-->write(6, mgetl(file_))

// Программа NadPe.sce
// Решение задачи о надёжности
// методом полного перебора.
// ==== 24 марта 2011 года ==== Визгунов НП..

clear, clc, mode(0), lines(0,90)

y_1 = 10
C = [ 1    3    2
      2    5    4
      3    6    5  ]
R = [ 0.6  0.7  0.5
      0.8  0.8  0.7
      0.9  0.9  0.9]

[u9, n] = size(C);    // u9 – число возможных управлений

x_min = ones(1,n)                  // [ 1  1  1  1 ]
x_max = sum(abs(R) ~= %inf, 'r')   // [ 3  3  3  3 ]

x     = x_min;
x_opt = x_min;
f_opt = -%inf;

j = n;
while %t
   if  x(j) <= x_max(j)
      cost = 0;
      for i = 1 :n
         cost = cost + C(x(i),i);
      end
      if cost <= y_1
         f = 1;
         for i = 1:n
            f = f * R(x(i),i);
         end
         if  f_opt < f
            // Для печати промежуточных данных убрать ;
            f_opt = f;
            x_opt = x;
         elseif f_opt == f
            f_opt;
            x_opt = [x_opt; x ];
         end
      end // cost
      j = n;
   else     // x(j) > x_max(j)
      x(j) = x_min(j);
      j = j - 1;
      if j <= 1e-8
         // Решение получено – выйти из цикла
```

```
            break
        end
    end // if
    x(j) = x(j) + 1;
end // while 1

write(6,[' ********'
         '   Ответ  '
         ' ********'])
write(6, ' f_opt = ' + string(f_opt))   // печать в строку
x_opt


-->exec(file_,0)
 y_1  =
    10.
 C  =
    1.    3.    2.
    2.    5.    4.
    3.    6.    5.
 R  =
    0.6    0.7    0.5
    0.8    0.8    0.7
    0.9    0.9    0.9
 x_min  =
    1.    1.    1.
 x_max  =
    3.    3.    3.
 *********
   Ответ
 *********
 f_opt = 0.504
 x_opt  =
    2.    1.    3.

-->diary(0)
```

6. Задача календарного планирования трудовых ресурсов

Задача 4 *Предпринимателю необходимо составить план регулирования численности рабочих на последующие пять месяцев – январь, февраль, март, апрель и май. Он оценивает следующим образом b_j– минимальные потребности в рабочей силе:*
$(b_1, \ldots, b_j, \ldots, b_n) = (5, 7, 8, 4, 6)$. *Другими словами, в январе ему потребуется не меньше 5 человек, в феврале 7 человек или больше и т.д.*

Предприниматель может принимать новых людей на работу и увольнять их. Пусть y_j – количество рабочих, которые работают в месяце j. Очевидно, должно выполняться $y_j \geq b_j$. При лишних рабочих $y_j > b_j$ предприниматель терпит убытки, которые вычисляются по формуле

$$C_л(y_j - b_j) = 3(y_j - b_j),\ j \in 1:5.$$

Расходы, связанные с наймом новых рабочих, вычисляются по другой формуле:

$$C_н(y_j - y_{j-1}) = \begin{cases} 4 + 2(y_j - y_{j-1}), & \text{при } y_j > y_{j-1}, \\ 0 & \text{иначе.} \end{cases}$$

Необходимо составить оптимальный план численности рабочих на 5 месяцев, при условии, что в конце декабря работало 5 человек.

Рекуррентное соотношение записывается в виде:

$$\begin{aligned} f_5(y_4) &= \min_{y_5=b_5}\{C_{\text{Л}}(y_5-b_5)+C_{\text{Н}}(y_5-y_4)\},\\ f_j(y_{j-1}) &= \min_{y_j\geq b_j}\{C_{\text{Л}}(y_j-b_j)+C_{\text{Н}}(y_j-y_{j-1})+f_{j+1}(y_j)\},\\ & j=4,3,2,1. \end{aligned}$$

Перед тем, как приступить к вычислениям, надо определить границы изменения переменных $y_1,\ldots,y_5$. Легко понять, что для получения оптимального решения достаточно рассмотреть значения $y_1\in 5:8,\ y_2\in 7:8,\ y_3=8,\ y_4\in 4:6,\ y_5=6$. Дело в том, что больше 8 человек не потребуется никогда, поэтому $y_j\leq 8$. Более жесткое ограничение $y_4\leq 6$ тоже совершенно понятно. y_4 - это количество рабочих в 4 месяце, в апреле. В мае нам потребуется ровно 6 человек, поэтому больше 6 рабочих нет необходимости оплачивать и в предыдущем месяце, в апреле. Дешевле лишних людей уволить в начале апреля, так как мы увольняем в нашей задаче рабочих без выходного пособия.

Решим задачу с помощью таблиц и рекуррентного уравнения Беллмана.

Этап 5. Май.

	$b_5=6$	Оптимальное решение	
	$C_{\text{Л}}(y_5-6)+C_{\text{Н}}(y_5-y_4)$		
y_4	$y_5=6$	$f_5(y_4)$	y_5^*
4	0+8=8	8	6
5	0+6=6	6	6
<u>6</u>	0+0=0	<u>0</u>	<u>6</u>

Таблица 20: Календарное планирование, этап 5

Этап 4. Апрель, май.

	$b_4=4$			Оптимальное решение	
	$C_{\text{Л}}(y_4-4)+C_{\text{Н}}(y_4-y_3)+f_5(y_4)$				
y_3	$y_4=4$	$y_4=5$	$y_4=6$	$f_4(y_3)$	y_4^*
<u>8</u>	0+0+8=8	3+0+6=9	6+0+0=6	<u>6</u>	<u>6</u>

Таблица 21: Календарное планирование, этап 4

Этап 3. Март, апрель, май.

	$b_3=8$	Оптимальное решение	
	$C_{\text{Л}}(y_3-8)+C_{\text{Н}}(y_3-y_2)+f_4(y_3)$		
y_2	$y_3=8$	$f_3(y_2)$	y_3^*
7	0+6+6=12	12	8
<u>8</u>	0+0+6= 6	<u>6</u>	<u>8</u>

Таблица 22: Календарное планирование, этап 3

Этап 2. Февраль, март, апрель, май.

	$b_2=7$		Оптимальное решение	
	$C_{\text{Л}}(y_2-7)+C_{\text{Н}}(y_2-y_1)+f_3(y_2)$			
y_1	$y_2=7$	$y_2=8$	$f_2(y_1)$	y_2^*
<u>5</u>	0+8+12=20	3+10+6=19	<u>19</u>	<u>8</u>
6	0+6+12=18	3+ 8+6=17	17	8
7	0+0+12=12	3+ 6+6=15	12	7
8	0+0+12=12	3+ 0+6= 9	9	8

Таблица 23: Календарное планирование, этап 2

Этап 1. Январь, февраль, март, апрель, май.

	$b_1 = 5$ $C_{\text{л}}(y_1 - 5) + C_{\text{н}}(y_1 - y_0) + f_2(y_1)$				Оптимальное решение	
y_0	$y_1 = 5$	$y_1 = 6$	$y_1 = 7$	$y_1 = 8$	$f_2(y_1)$	y_2^*
$\underline{5}$	0+0+ +19=19	3+6+ +17=26	6+8+ +12=26	9+10+ +9=28	$\underline{19}$	$\underline{5}$

Таблица 24: Календарное планирование, этап 1

Графическое решение достаточно сложное, поэтому это решение представлено на рисунке 7.

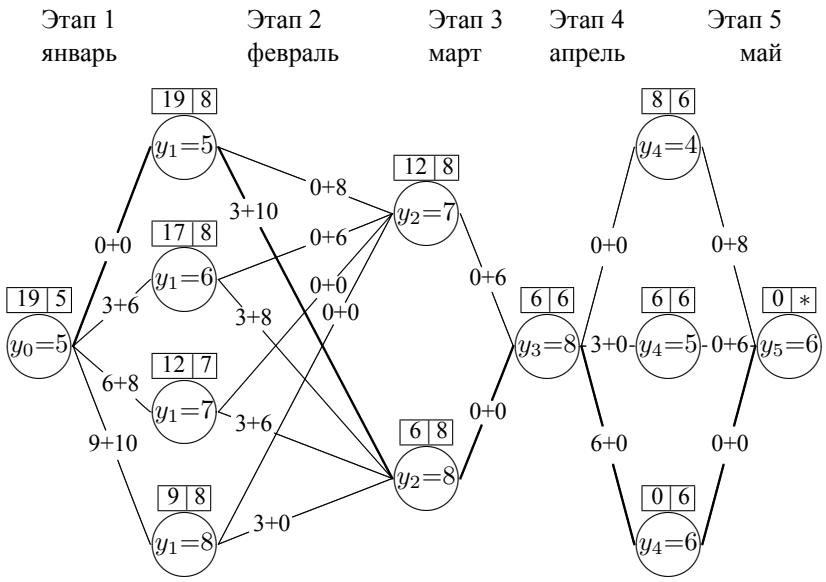


Рис. 7: Задача календарного планирования трудовых ресурсов.

Также изобразим рисунок, поясняющий исходные данные и ответ.

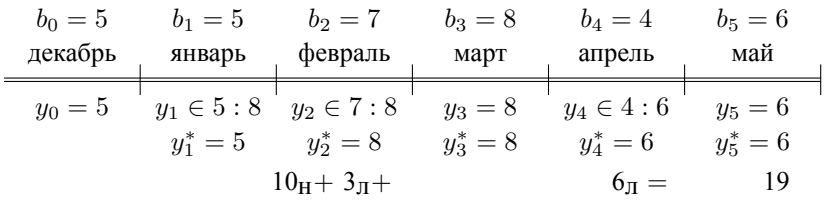


Рис. 8: Графическая иллюстрация исходных данных и ответа в задаче о календарном планировании трудовых ресурсов.

Еще раз подчеркнем, что ответом задачи является:

$$y^* = (5, 8, 8, 6, 6)$$

$$f_1(5) = 19.$$

6.1. Календарное планирование на компьютере

В распечатке есть главная программа `Trud.sce`, содержащая исходные данные. Главная программа обращается к двум подпрограммам (`function`), которые размещены в отдельном файле `Trud.sci` и решают задачу методом динамического программирования и методом полного перебора. Это

стандартный способ организации большой программы, принятый в большинстве языков программирования. В этой работе способ этот применяется впервые, потому что переменные из подпрограмм не видны в SciLab, и у студентов возникает много проблем при отладке.

```
-->write(6, mgetl('Trud.sce'))

// -*- coding: UTF-8 -*-
// программа Trud.sce  =======
// 24 марта 2011  Визгунов НП.

mode(0), lines(0), clc, clear

y0 = 5;
y_max = [8  8  8  6  6 ];
b     = [5  7  8  4  6 ];

// прочитать с диска содержимое файла Trud.sci
// с функциями TrudDi(...) и TrudPe(...)

if 0 == exists('TrudDi')
    TRUD = ".";  //для файлов в текущем директории
    // TRUD = "C:/DD/SciU/DinaPro/Trud"
    exec(TRUD + "/Trud.sci");
end

// Найти решения динамическим программированием
// и перебором

TrudDi(y0, y_max, b);
TrudPe(y0, y_max, b);

-->write(6, mgetl('Trud.sci'))

function [f_opt, y_opt] = TrudDi(y0, y_max, b)
// программа TrudDi.sce Динамическое программирование
// Визгунов НП. 24 марта 2011====================

n  = length(b);
s9 = max(y_max - b + 1);

F = %inf * ones(s9, n + 1);
Y = F;
F(:, n + 1) = zeros(s9, 1);

for j = n : -1 : 1    // j - номер этапа
   s7 = 1;
   if j >= 2
      s7 = y_max(j - 1) - b(j - 1) + 1;
   end

   u7 = y_max(j) - b(j) + 1;
   Fsu = %inf * ones(s7, u7);

   for s = 1 : s7
      Yj_1 = y0;
      if j >= 2
         Yj_1 = s + b(j - 1) - 1;
      end
      for u = 1 : u7
         Yj = u + b(j) - 1;
         Cn = 0;
         if Yj > Yj_1
            Cn = 4 + 2 * (Yj - Yj_1);
         end
         Fsu(s,u) = 3 * (Yj - b(j)) + Cn + F(u, j +1);
      end  // for u
```

```
    end  // for s

    // обработка временной таблицы Fsu

    [fs, index] = min(Fsu, 'c');
    F(1 : s7, j) = fs;
    Y(1 : s7, j) = index;

    // вычисление и печать шапки

    tire = part('=', ones(1,20)) + '|' + ...
       part('=', ones(1, 6 * u7 ));
    kepka = '␣␣␣Y' + string(j - 1) + '␣F(:,j)␣Y(:,j)␣|';
    for u = 1 : u7
       kepka = kepka + '␣Y' + string(j) + ...
          '=' + string(b(j) + u - 1);
    end
    write(6, '␣')
    write(6, '␣␣␣␣Этап␣' + string(j))
    write(6, tire)
    write(6, kepka)
    write(6, tire)

    // печать таблицы

    if j >= 2    // изза- b(0) надо разделять случаи
       disp([(b(j - 1) : b(j - 1) + s7 - 1)', ...
          fs, index + b(j) - 1, Fsu])
    else  // j == 1
       disp([y0, fs, index + b(j) - 1, Fsu])
    end
    write(6, tire)
end

// вычисление и печать результатов
// ================================

y_opt = zeros(1, n);
s = 1;
for j = 1 : n
    s = Y(s,j);
    y_opt(j) = s;
end // for j

f_opt = F(1, 1);
disp('ответ**********************')
write(6, '␣f_opt␣␣=␣' + string(f_opt))

y_opt = y_opt + b - 1;
write(6, '␣y_opt,␣b␣=␣')
write(6, y_opt, '(␣100(i4)␣)')
write(6, b,     '(␣100(i4)␣)')
endfunction  // [f_opt, y_opt] = TrudDi(y0, y_max, b)


//*****************************************
function [f_opt, y_opt] = TrudPe(y0, y_max, b)
//*****************************************

// программа TrudPe.sce Визгунов НП.
// ======= 24 марта 2011 года =======

n      = size(b,'c');
y_max0 = [y0,  y_max];
b0     = [y0,  b    ];
n = n +1
```

```
f_opt = %inf;  //задача на min, поэтому +%inf
y_opt = zeros(1, n +1);
y = b0; j = n;

write(6, '========================')
write(6, 'перебор================')

while %t
   if y(j) <= y_max0(j)

      f = 0;
      for j = 2 : n
         Cn = 0;
         if y(j) > y(j -1)
            Cn = 4 + 2 * (y(j) - y(j - 1));
         end
         f = f + 3 * (y(j) - b0(j)) + Cn;
      end //for j

      if f_opt > f     //задача на min !
         f_opt = f
         y_opt = y(2 : n)

         write(6, '>>>>>>>>>>>>>>>>>>>>>>>>')
         write(6, 'f_opt = ' + string(f))
         write(6, y_opt, '( "y_opt = ",100i4 )')

      elseif f_opt == f
         y_opt = [y_opt; y(2 : n)]

         write(6, '========================')
         write(6, 'f_opt = ' + string(f_opt))
         write(6, y_opt, '( "y_opt = ",100i4 )')
      end //elseif

      j = n;
   else       //y(j) > y_max(j)
      y(j) = b0(j);
      j = j - 1;
      if j <= 0.0001
         break
      end  //if
   end  //if
   y(j) = y(j) + 1;
end  //while %t

write(6, [' ==========='
          '    ответ   '
          ' ==========='])
write(6, ' f_opt равно ' + string(f_opt))
write(6, ' y0     равно  ' + string(y0))
y_opt
b = b0(2 : n)
endfunction  // [f_opt, y_opt] = TrudPP(y0, y_max, b)

-->exec('Trud.sce', 0)

    Этап 5
====================|======
   Y4 F(:,j) Y(:,j) | Y5=6
====================|======
    4.     8.     6.     8.
    5.     6.     6.     6.
    6.     0.     6.     0.
====================|======

    Этап 4
```

```
====================|==================
   Y3  F(:,j)  Y(:,j)  |  Y4=4  Y4=5  Y4=6
====================|==================
     8.      6.      6.      8.      9.      6.
====================|==================

     Этап  3
====================|======
   Y2  F(:,j)  Y(:,j)  |  Y3=8
====================|======
     7.      12.     8.      12.
     8.      6.      8.      6.
====================|======

     Этап  2
====================|============
   Y1  F(:,j)  Y(:,j)  |  Y2=7  Y2=8
====================|============
     5.      19.     8.      20.     19.
     6.      17.     8.      18.     17.
     7.      12.     7.      12.     15.
     8.      9.      8.      12.     9.
====================|============

     Этап  1
====================|========================
   Y0  F(:,j)  Y(:,j)  |  Y1=5  Y1=6  Y1=7  Y1=8
====================|========================
     5.      19.     5.      19.     26.     26.     28.
====================|========================

 ответ*********  ДП***********
 f_opt   = 19
 y_opt , b =
    5    8    8    6    6
    5    7    8    4    6

=========Perebor=========

>>>>>>>>>>>>>>>>>>>>>>>>>
f_opt = 22
y_opt =     5    7    8    4    6
>>>>>>>>>>>>>>>>>>>>>>>>>
f_opt = 20
y_opt =     5    7    8    6    6
>>>>>>>>>>>>>>>>>>>>>>>>>
f_opt = 19
y_opt =     5    8    8    6    6
 ============
     ответ
 ============
 f_opt  равно  19
 y0     равно   5

-->diary(0)
```

7. Задача о дилижансах

Задача 5 *Путешественник собирается пересечь нашу страну с востока на запад и добраться из Владивостока в далекий Калининград. Перед ним лежит карта, на которой указаны все маршруты. Для каждого маршрута указана цена билета. Выбрать самый дешевый маршрут, соединяющий Владивосток и Калининград.*

Карта, которая лежит перед нашим путешественником, приведена на рисунке 9. Карта немножко своеобразная, потому что любой маршрут, соединяющий город Владивосток ($y_5 = 10$) и город Калининград ($y_1 = 1$), состоит из 4 участков. Никаких прямых рейсов нет. Решить графически такую задачу не составляет никакого труда, это и сделано на рисунке. Решать задачу надо сначала для городов 7, 8 и 9 обычным способом. Это будет этап 4. Далее решается задача динамического программирования для городов 5 и 6 (этап 3). Затем надо выполнить для городов 2, 3 и 4(этап 2). Последним выполняется этап 1. Он соответствует самому западному городу Калининграду, который имеет номер $y_1 = 1$ на рисунке.

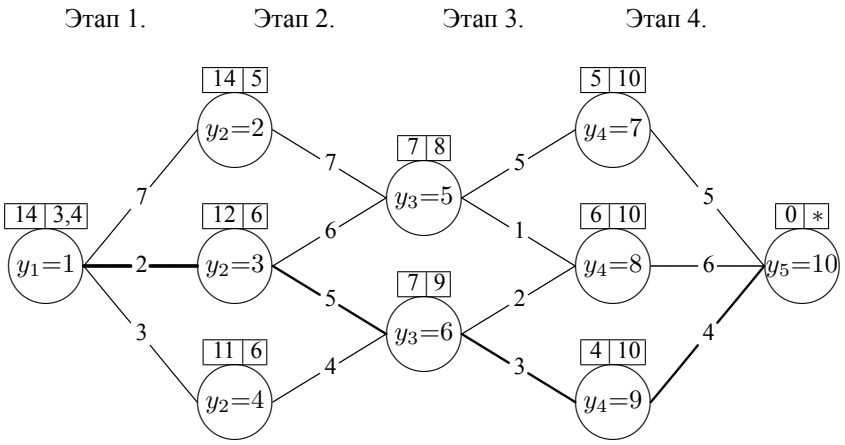


Рис. 9: Задача о дилижансах.

Мы еще не написали ни одной формулы, а уже решили задачу. Ответ сейчас можно написать так. Минимальные расходы равны 14 денежных единиц. Что такое денежная единица, уточнять не будем, так как при езде на дилижансах денежная единица может быть и не совсем традиционной. Один из оптимальных маршрутов – это набор городов $(y_1^*, \ldots, y_5^*) = (1, 3, 6, 9, 10)$.

Эту же задачу можно решить и с помощью таблиц. В этом случае без формул уже не обойтись.

Уравнение Беллмана будет выглядеть следующим образом:

$$f_j(y_j) = \min_{\substack{y_{j+1} | \text{маршрут} \\ (y_j, y_{j+1}) - \text{существует}}} \{R(y_j, y_{j+1}) + f_{j+1}(y_{j+1})\}.$$

Здесь нет традиционных переменных x_j, хотя нет никаких сомнений, что мы только что решили задачу динамического программирования. Состояния здесь – это y_j; $f_j(y_j)$ – минимальные расходы на переезд в соответствующий город этапа j. Что же здесь является управлением? Да, конечно же, переменная y_{j+1}, по которой мы ищем минимум.

Сейчас решим задачу о дилижансах с помощью таблиц. Как и раньше, по строчкам будут располагаться все возможные состояния, по столбцам — перечисляются все значения управлений.

Этап 4:

$$f_4(y_4) = \min_{y_5 = 10} \{R(y_4, y_5) + f_5(y_5)\}.$$

	$R(y_4, y_5) + 0$	Оптимальное решение	
y_4	$y_5 = 10$	$f_4(y_4)$	y_5^*
7	5	5	10
8	6	6	10
9	4	4	10

Таблица 25: Задача о дилижансах, этап 4

Этап 3:

$$f_3(y_3) = \min_{y_4 \in \{7,8,9\}} \{R(y_3, y_4) + f_4(y_4)\}.$$

	$R(y_3, y_4) + f_4(y_4)$			Оптимальное решение	
y_3	$y_4 = 7$	$y_4 = 8$	$y_4 = 9$	$f_3(y_3)$	y_4^*
5	5+5=10	1+6=7	—	7	8
6	—	2+6=8	3+4=7	7	9

Таблица 26: Задача о дилижансах, этап 3

Этап 2:

$$f_2(y_2) = \min_{y_3 \in \{5,6\}} \{R(y_2, y_3) + f_3(y_3)\}.$$

	$R(y_2, y_3) + f_3(y_3)$		Оптимальное решение	
y_2	$y_3 = 5$	$y_3 = 6$	$f_2(y_2)$	y_3^*
2	7+7=14	—	14	5
3	6+7=13	5+7=12	12	6
4	—	4+7=11	11	6

Таблица 27: Задача о дилижансах, этап 2

Этап 1:

$$f_1(y_1) = \min_{y_2 \in \{2,3,4\}} \{R(y_1, y_2) + f_2(y_2)\}.$$

	$R(y_1, y_2) + f_2(y_2)$			Оптимальное решение	
y_1	$y_2 = 2$	$y_2 = 3$	$y_2 = 4$	$f_1(y_1)$	y_2^*
1	7+14=21	2+12=14	3+11=14	14	3 ,4

Таблица 28: Задача о дилижансах, этап 1

8. Управление запасами

Задача 6 (Управление запасами.) *Завод может производить ежемесячно до 4 единиц некоторой продукции. Затраты на производство x_j единиц продукции в месяце j указаны в следующей таблице 29. Каждый месяц завод должен отгрузить 2 единицы продукции своим потребителям. Продукцию можно хранить на складах. Затраты на хранение 1 единицы продукции составляют 1 миллион рублей. Оплата за хранение производится в конце месяца j. Склады могут вместить до 4 единиц продукции. Учитывая, что в начале на складах было 2 единицы продукции, определить оптимальный план производства на 4 месяца.*

x_j	0	1	2	3	4	штук
$C(x_j)$	0	7	9	11	13	млн. руб.

Таблица 29: Затраты на производство x_j единиц продукции.

Введем обозначения для переменных и констант задачи.

n – количество месяцев планового отрезка,

x_j – выпуск продукции в месяце j, $j = 1, \ldots, n$ (управление),

y_j – запасы в конце месяца j, или в начале месяца $j + 1$ (состояние),

d_j – спрос в месяце j, заданная величина.

Переменные x_j и y_j связаны следующим балансовым соотношением

$$y_j = y_{i-1} + x_j - d_j, \quad j = 1, \ldots, n$$

— запасы y_j в конце текущего j-го месяца равны запасам в предыдущем месяце y_{j-1} плюс x_j произведенных в текущем месяце j изделий и минус d_j проданных изделий.

В числовом примере, который мы будет использовать, $n = 4$. Мы будет планировать на январь, февраль, март и апрель. $d_j = 2,\ j = 1, \ldots, n,\ x_j, y_j \in 0 : 4$

Математическая модель задачи выглядит следующим образом. Найти оптимальный план производства $x_1, \ldots, x_n$, минимизирующий затраты

$$\left.\begin{array}{l} f_1(y_0) = \min\limits_{x_1,\ldots,x_n} \left\{ \sum\limits_{j=1}^{n} C(x_j) + 1(y_{j-1} + x_j - 2) \right\} \\ \text{при ограничениях} \\ \begin{array}{llllll} y_0 & +x_1 & -2 & & & = y_1, \\ & y_1 & +x_2 & -2 & & = y_2, \\ & & \ldots\ldots & & & \\ & & y_{j-1} & +x_j & -2 & = y_j, \\ & & & & \ldots\ldots & \\ & & & & y_{n-1} \quad +x_n - 2 & = y_n, \end{array} \\ x_j, y_j \in 0 : 4 \text{ для любого } j, \\ y_0 = 2,\ y_n = 0. \end{array}\right\} \qquad (8)$$

Для решения этой задачи следует записать уравнение Беллмана. Для этого надо сложить затраты в месяце j и затраты на временном отрезке $j + 1, \ldots, n$, а затем минимизировать суммарные затраты выбором x_j:

$$f_j(y_{j-1}) = \min_{x_j}\{C(x_j) + 1 \cdot (y_{j-1} + x_j - 2) + f_{j+1}(y_{j-1} + x_j - 2)\}.$$

Справедливость уравнения Беллмана легко доказывается, несмотря на то, что в исходной задаче 8 довольно много ограничений.

Когда строго выводят уравнение Беллмана из задачи 8, то математики говорят об этом факте следующей фразой: ”< *доказано, что принцип оптимальности Беллмана справедлив*”>. Мы проводили уже один раз такое доказательство в самой первой задаче – в задаче о распределении инвестиций. Не будем повторять эти рассуждения, так как они достаточно простые.

Начинаем решение задачи о запасах с последнего месяца — апреля.

Этап 4: апрель.

$$f_4(y_3) = \min_{x_4 \in 0:4}\{C(x_4) + 1 \cdot (y_3 + x_4 - 2)\}.$$

	Оптимальное решение	
y_3	$f_4(y_3)$	x_4^*
0	9	2
1	7	1
2	0	0

Таблица 30: Управление запасами, этап 4

В конце апреля запасы на складе должны быть равны нулю $y_3 + x_4 - 2 = y_4 = 0$, отсюда $y_3 + x_4 = 2$ и все возможные значения y_3 и x_4 в таблице 30 перечислены. Таких пар всего три. Для каждого возможного значения (y_3, x_4) вычислена целевая функция по совсем простой формуле $C(x_4)$. Для этапа 3 (март, апрель) из $y_3 \in 0 : 4,\ y_3 = y_2 + x_3 - 2$ следует

$$0 \le y_2 + x_3 - 2 \le 4.$$

Левое ограничение $0 + 2 \le y_2 + x_3$ дает запрещенные варианты в левой верхней части таблицы 31, правое ограничение $y_2 + x_3 \le 4 + 2$ соответствует черточкам в правой нижней части таблицы. В этой таблице уже перечислены все пять возможных значений состояния $y_2 \in 0 : 4$.

Для этапа 2 «северо-западные» и «юго-восточные» углы матрицы заполняются черточками из-за ограничения $0 \le y_2 = y_1 + x_2 - 2 \le 4$.

Этап 3: март, апрель.

$$f_3(y_2) = \min_{x_3 \in 0:4} \{C(x_3) + 1 \cdot (y_2 + x_3 - 2) + f_4(y_2 + x_3 - 2)\}.$$

	$C(x_3) + 1 \cdot (y_2 + x_3 - 2) + f_4(y_2 + x_3 - 2)$					Оптим. решение	
y_2	$x_3 = 0$	$x_3 = 1$	$x_3 = 2$	$x_3 = 3$	$x_3 = 4$	$f_3(y_2)$	x_3^*
0	—	—	9+0+9 =18	11+1+7 =19	13+2+0 =15	15	4
1	—	7+0+9 =16	9+1+7 =17	11+2+0 =13	—	13	3
2	0+0+9 =9	7+1+7 =15	9+2+0 =11	—	—	9	0
3	0+1+7 =8	7+2+0 =9	—	—	—	8	0
4	0+2+0 =2	—	—	—	—	2	0

Таблица 31: Управление запасами, этап 3

Этап 2: февраль, март, апрель.

$$f_2(y_1) = \min_{x_2 \in 0:4} \{C(x_2) + 1 \cdot (y_1 + x_2 - 2) + f_3(y_1 + x_2 - 2)\}.$$

	$C(x_2) + 1 \cdot (y_1 + x_2 - 2) + f_3(y_1 + x_2 - 2)$					Оптим. решение	
y_1	$x_2 = 0$	$x_2 = 1$	$x_2 = 2$	$x_2 = 3$	$x_2 = 4$	$f_2(y_1)$	x_2^*
0	—	—	9+0+15 =24	11+1+13 =25	13+2+9 =24	24	2, 4
1	—	7+0+15 =22	9+1+13 =23	11+2+9 =22	13+3+8 =24	22	1,3
2	0+0+15 =15	7+1+13 =21	9+2+9 =20	11+3+8 =22	13+4+2 =19	15	0
3	0+1+13 =14	7+2+9 =18	9+3+8 =20	11+4+2 =17	—	14	0
4	0+2+ 9 =11	7+3+8 =18	9+4+2 =15	—	—	11	0

Таблица 32: Управление запасами, этап 2

Этап 1: январь,февраль, март, апрель.

$$f_1(y_0) = \min_{x_1 \in 0:4} \{C(x_1) + 1 \cdot (y_0 + x_1 - 2) + f_2(y_0 + x_1 - 2)\}.$$

	$C(x_1) + 1 \cdot (y_0 + x_1 - 2) + f_2(y_0 + x_1 - 2)$					Оптим. решение	
y_0	$x_1 = 0$	$x_1 = 1$	$x_1 = 2$	$x_1 = 3$	$x_1 = 4$	$f_1(y_0)$	x_1^*
0	—	—	9+0+24 =33	11+1+22 =34	13+2+15 =30	30	4
1	—	7+0+24 =31	9+1+22 =32	11+2+15 =28	13+3+14 =30	28	3
2	0+0+24 =24	7+1+22 =30	9+2+15 =26	11+3+14 =28	13+4+11 =28	24	0
3	0+1+22 =23	7+2+15 =24	9+3+14 =26	11+4+11 =26	—	23	0
4	0+2+15 =17	7+3+14 =24	9+4+11 =24	—	—	17	0

Таблица 33: Управление запасами, этап 1

8.1. Вычисление оптимального решения

Находим оптимальное решение, начиная с последнего этапа 1. $y_0 = 2$ из условий задачи, поэтому из таблицы 33 следует, что $x_1^* = 0$, $y_1 = y_0 + x_1^* - 2 = 0$. В таблице этапа 2 в строке $y_1 = 0$ видим два оптимальных решения. Рассмотрим сначала решение $x_2^* = 4$. Это решение подчеркнуто. Из уравнения $y_2 = y_1 + x_2^* - 2$ находим $y_2 = 0 + 4 - 2 = 2$. Переходим к таблице этапа 3 к строчке $y_2 = 2$. Понятно, что соответствующая компонента решения $x_3^* = 0$ и в таблице этапа 4 надо смотреть на строку $y_3 = y_2 + 0 - 2 = 0$. Эта строка $y_2 = 0$ дает последнюю компоненту решения $x_4^* = 2$.

Теперь можно записать окончательный ответ.

$$x^* = (0, 4, 0, 2), \quad f_1(2) = 24.$$

В феврале надо произвести 4 единицы продукции и в апреле 2 единицы, при этом за четыре месяца будет затрачено 24 миллиона рублей. С меньшими затратами нельзя выполнить намеченную программу. Этот оптимальный план не единственный. Другой оптимальный план получить самостоятельно.

8.2. Управление запасами на компьютере

В методе полного перебора полезно напечатать в качестве ответа не только x^* — выпуск продукции в каждом месяце, но и запасы в конце каждого месяца y^*. Например, из компьютерного решения видно, что одним из ответов является следующий:

$$\begin{gathered} f_1(2) = 24 \\ x^* = (0, 4, 0, 2) \\ y^* = (0, 2, 0, 0) \end{gathered}$$

```
-->file_ = 'ZapasDi.sce';
-->write(6, mgetl(file_))

// программа ZapasDi.sce Динамическое программирование,
// задача управления запасами.
// Визгунов НП. 25 марта 2011 года
// ==============================

clc, clear, mode(0), lines(0)

//x = 1  2  3  4  5
c1 = [0  7  9 11 13]; // Затраты на выпуск x-1 ед. продукции
x9 = length(c1) - 1;  // Максимум производства в штуках
y0 = 2;               // Запасы на складе в конце декабря
y9 = 4;               // Максимум запасов на складе в штуках
k  = 1;               // Плата за единицу хранения на складе

d = 2;                // Отгрузка потребителям
n = 4;                // Количество месяцев в плановом периоде

s9 = y9 + 1;          // Макс. количество состояний
u9 = x9 + 1;          // Макс. количество управлений

F = %inf * ones(y9 + 1, n);
X = F;

// Начальные вычисления для этапа n.

for s = 1 : s9
   u = d - s + 2;
   if u >= 1
      F(s, n) = c1(u);
      X(s, n) = u;
```

```
   end
end  // for s

// Печать для этапа n

j = n;
write(6, '␣␣␣Этап␣' + string(j))
tire = part('=', ones(1, 19)) + '|';
kepka = '␣␣␣Y' + string(n - 1) + '␣F(:,j)␣Y(:,j)␣|';

s7 = sum(F(:, n) < %inf );

write(6, tire)
write(6, kepka)
write(6, tire)
disp([(0 : s7 -1)', F(1 : s7, n), X(1 : s7, n) - 1])
write(6, tire)

// Произвольный jый- этап.

Fsu = %inf * ones(s9, u9);
for j = n -1 : -1 : 1
   for s = 1 : s9          // s = Yj_1 + 1
      for u = 1 : u9       // u = Xj + 1
         s1 = (s -1 ) + (u - 1) - d + 1;
         if  1  <= s1  & s1 <= n  + 1
            Fsu(s, u) = c1(u) + k * (s1 - 1) + ...
                        F(s1, j + 1);
         end
      end  // for u
   end  // for s
   [Fs, Index] = min(Fsu, 'c');

   // Обработка таблиц Fsu - F, состояние, управление

   F(:, j) = Fs;
   X(:, j) = Index;

   // вычисление и печать шапки

   tire = part('=', ones(1,19)) + '|' + ...
          part('=', ones(1, u9 * 7));
   kepka = '␣␣␣Y' + string(j - 1) + '␣F(:,j)␣Y(:,j)|';
   for u = 1 : u9
      kepka = kepka + '␣␣␣X' + string(j) + ...
                      '=' + string(u - 1);
   end
   write(6, '␣')
   write(6, '␣␣␣Этап␣' + string(j))
   write(6, tire)
   write(6, kepka)
   write(6, tire)
   disp([(0 : s9 - 1)', F(:, j), X(:, j) - 1, Fsu])
   write(6, tire)
end  // for j

// печать результатов
// ==================

x_opt = zeros(1, n);
s = y0 + 1;
for j = 1 : n
   x_opt(j) = X(s, j);
   s = (s - 1) + X(s, j) - d ;
end  // for j

x_opt = x_opt - 1
```

```
f_opt = F(y0 + 1, 1)

-->exec(file_, 0)

   Этап 4
===================|
   Y3 F(:,j) Y(:,j)|
===================|
    0.     9.     2.
    1.     7.     1.
    2.     0.     0.
===================|

   Этап 3
===================|===================================
   Y2 F(:,j) Y(:,j)|    X3=0    X3=1    X3=2    X3=3    X3=4
===================|===================================
    0.     15.    4.     Inf    Inf     18.     19.     15.
    1.     13.    3.     Inf    16.     17.     13.     Inf
    2.     9.     0.     9.     15.     11.     Inf     Inf
    3.     8.     0.     8.     9.      Inf     Inf     Inf
    4.     2.     0.     2.     Inf     Inf     Inf     Inf
===================|===================================

   Этап 2
===================|===================================
   Y1 F(:,j) Y(:,j)|    X2=0    X2=1    X2=2    X2=3    X2=4
===================|===================================
    0.     24.    2.     Inf    Inf     24.     25.     24.
    1.     22.    1.     Inf    22.     23.     22.     24.
    2.     15.    0.     15.    21.     20.     22.     19.
    3.     14.    0.     14.    18.     20.     17.     Inf
    4.     11.    0.     11.    18.     15.     Inf     Inf
===================|===================================

   Этап 1
===================|===================================
   Y0 F(:,j) Y(:,j)|    X1=0    X1=1    X1=2    X1=3    X1=4
===================|===================================
    0.     30.    4.     Inf    Inf     33.     34.     30.
    1.     28.    3.     Inf    31.     32.     28.     30.
    2.     24.    0.     24.    30.     26.     28.     28.
    3.     23.    0.     23.    24.     26.     26.     Inf
    4.     17.    0.     17.    24.     24.     Inf     Inf
===================|===================================

 x_opt  =
    0.     2.     4.     0.
 f_opt  =
    24.

-->diary(0)
```

```
-->file_ = 'ZapasPe.sce';
-->write(6, mgetl(file_))

// Программа ZapasPe.sce
// =====================
// Управление запасами - полный перебор.
// ======== Визгунов НП.. 24 марта 2011.

clc, clear, lines(0), mode(0)

// x:  1  2  3  4  5
c1 =  [0  7  9 11 13]; // Затраты на выпуск x-1 ед. продукции
x9 = length(c1) - 1;  // Максимум производства в штуках
y0 = 2;               // Запасы на складе в конце декабря
```

```
y9 = 4;                 // Максимум запасов на складе в штуках
k  = 1;                 // Плата за единицу хранения на складе

d = 2;                  // Отгрузка потребителям
n = 4;                  // Количество месяцев в плановом периоде

s9 = y9 + 1;            // Макс. количество состояний
u9 = x9 + 1;            // Макс. количество управлений

x = zeros(1, n);
y = zeros(1, n);
f_opt = %inf;
x_opt = x;

j = n;
while %t
   if x(j) <= x9

      y(1) = y0 + x(1) - d;
      for j = 2 : n
         y(j) = y(j - 1) + x(j) - d;
      end  // for

      if  0 <= min(y) & max(y) <= y9

         f = 0;
         for j = 1 : n
            f = f + c1(x(j) + 1) + k * y(j);
         end
         if  f_opt == f
            write(6, '=====␣f_opt␣=␣' + string(f_opt))
            x_opt = [x_opt; x]
            y_opt = [y_opt; y];
         elseif f_opt > f
            f_opt = f;
            write(6, '>>>>>␣f_opt␣=␣' + string(f_opt))
            x_opt = x
            y_opt = y;
         end
      end  // if

      j = n;
   else
      x(j) = 0;
      j = j - 1;
      if j <= 0.0001
         break
      end  // if
   end  // if

   x(j) = x(j) + 1;
end

disp('*******␣Ответ:␣*******')
disp('f_opt␣␣=␣' + string(f_opt))
x_opt, y_opt


-->exec(file_, 0)

>>>>>>>>>>>> f_opt = 27
 x_opt  =
    0.    2.    2.    2.
>>>>>>>>>>>> f_opt = 24
 x_opt  =
    0.    2.    4.    0.
============ f_opt = 24
```

```
 x_opt  =
    0.     2.     4.     0.
    0.     4.     0.     2.

 *******  Ответ:  *******

 f_opt  = 24
 x_opt  =
    0.     2.     4.     0.
    0.     4.     0.     2.
 y_opt  =
    0.     0.     2.     0.
    0.     2.     0.     0.

-->diary(0)
```

9. Замена оборудования.

Задача 7 *К началу текущей пятилетки на предприятии установлено новое оборудование. Производительность этого оборудования и затраты на содержание и ремонт зависят от времени. В таблице 34 показаны эти зависимости. Затраты на приобретение и установку нового оборудования составляют 40 тысяч рублей, старое оборудование списывается. Составить план замены оборудования на пять лет, максимизирующий общую прибыль.*

	Время y_j, в течение которого используется оборудование (лет)				
$y_j =$	0	1	2	3	4
Годовой выпуск продукции $R(y_j)$ (тысяч рублей)	80	75	65	60	60
Ежегодные затраты $C(y_j)$ на содержание и ремонт оборудования (тысяч руб.)	20	25	30	35	45

Таблица 34: Задача о замене оборудования

Введем следующие обозначения.

x_j – управление. Переменная x_j может принимать два значения, в зависимости от того, будем покупать новое оборудование или нет.

y_j – возраст оборудования к началу $j-$го года (или к конце $j-1$ года)

$f_j(y_j)$ – максимальный доход на временном отрезке $j,\dots,n$, при возрасте оборудования y_j лет в начале $j-$го года.

$C_{\text{н}}$ – начальные затраты на приобретение и установку нового оборудования (цена нового оборудования).

Составляем уравнение Беллмана.

$$f_j(y_j) = \max_{x_j \in \{\text{с},\text{н}\}} \begin{cases} R(y_j) - C(y_j) + f_{j+1}(y_j+1) & \text{при } x_j = \text{стар.}, \\ R(0) - C(0) - C_{\text{н}} + f_{j+1}(1) & \text{при } x_j = \text{нов.} \end{cases}$$

Решение начинается с последнего 5 года пятилетки. Пусть это будет 1975 год. Так как в 1971 году оборудование было новым, то к началу 1975 года y_5 может принимать значения 1, 2, 3 или 4.

Теперь приступим к этапу 3. В начале 1973 года возраст оборудования может принимать значения 1 и 2.

Теперь приступим к последнему этапу 2. В начале 1972 года возраст оборудования может принимать значения только одно значение - 1 год.

Этап 5:

$$f_5(y_5) = \max_{x_5 \in \{\text{с,н}\}} \begin{cases} R(y_5) - C(y_5) & \text{при } x_5 = \text{стар.}, \\ R(0) - C(0) - C_{\text{н}} = 20 & \text{при } x_5 = \text{нов.} \end{cases}$$

	$R(y_5) - C(y_5)$	$R(0) - C(0) - C_{\text{н}}$ =20	Оптимальное решение	
y_5	$x_5 =$ старое	$x_5 =$ новое	$f_5(y_5)$	x_5^*
1	75-25 = 50	80-20-40 = 20	50	с
2	65-30 = 35	80-20-40 = 20	35	с
3	60-35 = 25	80-20-40 = 20	25	с
4	60-45 = 15	80-20-40 = 20	20	н

Таблица 35: Замена оборудования, этап 5

Этап 4:

$$f_4(y_4) = \max_{x_4 \in \{\text{с,н}\}} \begin{cases} R(y_4) - C(y_4) + f_5(y_4 + 1) & \text{при } x_4 = \text{стар.}, \\ R(0) - C(0) - C_{\text{н}} + f_5(1) & \text{при } x_4 = \text{нов.} \end{cases}$$

	$R(y_4) - C(y_4) + f_5(y_4 + 1)$	$R(0) - C(0) - C_{\text{н}} + f_5(1)$	Оптимальное решение	
y_4	$x_4 =$ старое	$x_4 =$ новое	$f_4(y_4)$	x_4^*
1	75-25+35 = 85	20+50 = 70	85	с
2	65-30+25 = 60	20+50 = 70	70	н
3	60-35+20 = 45	20+50 = 70	70	н

Таблица 36: Замена оборудования, этап 4

Этап 3:

$$f_3(y_3) = \max_{x_3 \in \{\text{с,н}\}} \begin{cases} R(y_3) - C(y_3) + f_4(y_3 + 1) & \text{при } x_3 = \text{стар.}, \\ R(0) - C(0) - C_{\text{н}} + f_4(1) & \text{при } x_3 = \text{нов.} \end{cases}$$

	$R(y_3) - C(y_3) + f_4(y_3 + 1)$	$R(0) - C(0) - C_{\text{н}} + f_4(1)$	Оптимальное решение	
y_3	$x_3 =$ старое	$x_3 =$ новое	$f_3(y_3)$	x_3^*
1	75-25+70 = 120	20+85 = 105	120	с
2	65-30+70 = 105	20+85 = 105	105	с,н

Таблица 37: Замена оборудования, этап 3

Этап 2:

$$f_2(y_2) = \max_{x_2 \in \{\text{с,н}\}} \begin{cases} R(y_2) - C(y_2) + f_3(y_2 + 1) & \text{при } x_2 = \text{стар.}, \\ R(0) - C(0) - C_{\text{н}} + f_3(1) & \text{при } x_2 = \text{нов.} \end{cases}$$

	$R(y_2) - C(y_2) + f_3(y_2 + 1)$	$R(0) - C(0) - C_{\text{н}} + f_3(1)$	Оптимальное решение	
y_2	$x_2 =$ старое	$x_2 =$ новое	$f_2(y_2)$	x_2^*
1	75-25+105 = 155	20+120 = 140	155	с

Таблица 38: Замена оборудования, этап 2

Осталось из таблиц найти оптимальное решение. В начале 1971 года оборудование было новым. Из последней таблицы этапа 2 следует, что в начале 1972 года надо оборудование не менять, оставить старое. В конце 1972 года этому оборудованию будет 2 года, поэтому в таблице этапа 3 надо выбрать строку, соответствующую $y_3 = 2$. В этой строке записано, что оборудование можно оставить прежнее, а можно и поменять, доход от этого не изменится. Заменим оборудование, купим новое. До конца пятилетки будет пользоваться этим оборудованием. В конце 1975 года списы-

ваем его. Оптимальное решение

$$f_2(1) = 155, \;\; (x_2^*, x_3^*, x_4^*, x_5^*) = (\text{с, н, с, с.})$$

Лучше учесть и доходы от нового оборудования в 1971 году. Они равны $R(0) - c(0) = 60$. Окончательно получаем оптимальное решение в более привычном виде:

$$f_1(0) = 215, \;\; x^* = (x_1^*, \ldots, x_5^*) = (\text{н, с, н, с, с.})$$

На рисунке 10 изображено это оптимальное решение в более наглядном виде.

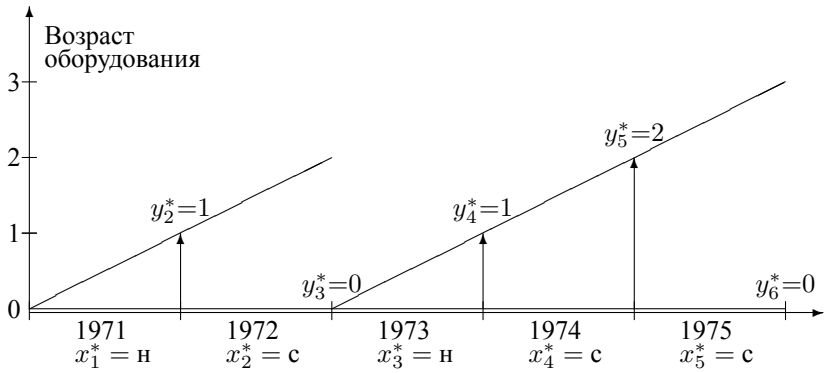


Рис. 10: Ответ в графическом виде для задачи о замене оборудования.

9.1. Замена оборудования на компьютере

Как обычно, приводятся программы для решения задачи методом динамического программирования и полным перебором. Из компьютерного решения видно, что у задачи есть еще одно решение

$$f_1(0) = 215, \;\; x^* = (x_1^*, \ldots, x_5^*) = (\text{н, с, с, н, с.}).$$

Другими словами, можно оборудование поменять не только в начале 1973 года, но и в конце его.

```
-->write(6, mgetl('ZamenaDi.sce'))

// function [] = ZamenaDi(r1, c1, Cn, y1, n)
// программа ZamenaDi.sce Динамическое программирование
// Замена оборудования на 5 лет или замена автомобиля
// Визгунов НП. 25 марта 2011 ============
mode(0), lines(0), clear, clc

// s: 1 2 3 4 5
r1 = [80 75 65 60 60]; // Доход от авто возраста s-1 лет
c1 = [20 25 30 35 45]; // Затраты на него
Cn = 40;               // Цена нового авто
y1 = 0;                // В начале 1971 года авто - новый
n = 5;                 // Кол. месяцев в плановом периоде
if y1 + length(c1) > n
   error('не хватает данных')
end

Rnew = r1(1) - c1(1) - Cn;  // Прибыль от нового авто
Fsu  = %inf * ones(n ,2);
X = %inf * ones(n ,n +1);
F = X;
F(:, n +1) = zeros(n, 1);
Tire = part('-', ones(1, 20)) + '|' + ...
part('-', ones(1, 16));
```

```
for j = n : -1 : 2
   for Yj = 1 : j -1    // s = Yj
      for Xj = 0 : 1    // u = Xj +1
         if Xj == 0     // старая машина
            Fsu(Yj,1) = r1(Yj + 1) - c1(Yj + 1) + ...
               F(Yj + 1, j + 1);
         else       // новая Xj == 1
            Fsu(Yj, 2) = Rnew + F(1, j + 1);
         end  // if
      end  // for Xj
   end  // for Yj
   [Fs, Index] = max(Fsu, 'c');
   F(1 : j - 1, j) = Fs(   1 : j - 1);
   X(1 : j - 1, j) = Index(1 : j - 1) - 1;

   // Печать шапки и таблицы

   write(6, ' ')
   write(6, '    Этап ' + string(j))
   Sj = string(j);
   Kepka = '    Y' + Sj + '      F      X* | ';
   Kepka = Kepka + '  X' + Sj + '=old  X' + Sj + '=new ';

   write(6, Tire);
   write(6, Kepka);
   write(6, Tire);
   disp([(1 : j -1)', Fs(1 : j -1), ...
      Index(1: j -1)-1, Fsu(1: j -1, :)])
   write(6, Tire)
end  // for j

// Вычисление и печать результатов
// ================================

Yj = 1;
for j = 1 : n
   if X(Yj, j) == 0  // старый авто
      Yj = Yj + 1;
      x_opt(j) = 'ста.  ';
   else              // новый  авто
      Yj = 1;
      x_opt(j) = 'нов.  ';
   end  // if
end  // for

disp('x_opt =')
disp(x_opt')

f_opt = F(1,2) + r1(1) - c1(1)

-->exec('ZamenaDi.sce',0)
```

Этап 5

Y5	F	X*	X5=old	X5=new
1.	50.	0.	50.	20.
2.	35.	0.	35.	20.
3.	25.	0.	25.	20.
4.	20.	1.	15.	20.

Этап 4

Y4	F	X*	X4=old	X4=new

```
    1.    85.    0.    85.    70.
    2.    70.    1.    60.    70.
    3.    70.    1.    45.    70.
-------------------------------------------

    Этап 3
-------------------------------------------
    Y3      F      X*  |  X3=old   X3=new
-------------------------------------------
    1.    120.    0.      120.     105.
    2.    105.    0.      105.     105.
-------------------------------------------

    Этап 2
-------------------------------------------
    Y2      F      X*  |  X2=old   X2=new
-------------------------------------------
    1.    155.    0.      155.     140.
-------------------------------------------

 x_opt =
нов.     ста.     ста.     нов.     ста.
 f_opt  =
    215.

-->diary(0)
```

```
-->write(6, mgetl('ZamenaPe.sce'))

//    программа ZamenaPe.sce
// ========================
// Замена оборудования – полный перебор.
// Визгунов НП.  25 марта 2011 года

clc, clear, mode(0), lines(0)

// s:  1  2  3  4  5
r1 = [80 75 65 60 60]; // Доход от авто возраста s–1 лет
c1 = [20 25 30 35 45]; // Затраты на него
Cn = 40;               // Цена нового авто
y1 = 0;                // В начале 1971 года авто– новый

n = 5;                 // Кол. месяцев в плановом периоде
if y1 + length(c1) > n
    error('не хватает данных')
end
// Rnew = r1(1) – c1(1) – Cn;// Прибыль от нового авто

f_opt = –%inf;
x_opt = zeros(1, n + 1);

x_max =  ones(1, n);
x_min = zeros(1, n);

x = zeros(1, n);
j = n;
while %t
    if x(j) <= x_max(j)

        for j = 1 : n
            if x(j) == 1
                y(j) = 0;
            elseif x(j) == 0 & j >= 2
                y(j) = y(j – 1) + 1;
            elseif x(1) == 0
                y(1) = y1;
            end
```

```
        end  // j

        f = Cn;
        for j = 1 : n
            if x(j) == 0 & y(j) >= 1
                f = f + r1(y(j) + 1) - c1(y(j) + 1);
            elseif x(j) == 1
                f = f + r1(1) - c1(1) - Cn;
            end
        end

        if  f_opt < f
            f_opt = f;
            write(6, f_opt, '( ""<<<< f_opt = "", i6 )')
            x_opt = x
        elseif f_opt == f
            write(6, f_opt, '( ""==== f_opt = "", i6 )')
            x_opt = [x_opt; x]

        end  // elseif

        j = n;
    else      // x(j) > x_max(j)
        x(j) = x_min(j);
        j = j - 1;
        if j <= 0.00001
            break
        end  // if
    end  // if
    x(j) = x(j) + 1;
end  // while %t

disp('ответ')
x_opt, f_opt

-->exec('ZamenaPe.sce',0)

<<<<<< f_opt =    165
 x_opt  =
    0.    0.    0.    0.    0.
<<<<<< f_opt =    170
 x_opt  =
    0.    0.    0.    0.    1.
<<<<<< f_opt =    195
 x_opt  =
    0.    0.    0.    1.    0.
====== f_opt =    195
 x_opt  =
    0.    0.    0.    1.    0.
    0.    0.    1.    0.    0.
<<<<<< f_opt =    215
 x_opt  =
    1.    0.    0.    1.    0.
====== f_opt =    215
 x_opt  =
    1.    0.    0.    1.    0.
    1.    0.    1.    0.    0.

 ответ
 x_opt  =
    1.    0.    0.    1.    0.
    1.    0.    1.    0.    0.
 f_opt  =
    215.

-->diary(0)
```

10. Программы на $Python$

10.1. Решение задачи о распределении инвестиций

10.1.1. Динамическое программирование

```
// Программа InvestDi.sce
// Решение задачи распределения инвестиций
// методом динамического программирования.
// ==== 2 января 2011 года ==== Визгунов Н.П.

clear, clc, mode(0), lines(0,90)

y_1 = 5;
C = [
  0    0    0    0
  1    3    1    2
-%inf  5    2  -%inf ];

R = [
  0    0    0    0
  3    5    4    3
-%inf  9    6  -%inf ];

[x9,n] = size(C);   // x9        - Количество управлений
y9     = y_1 + 1;   // y9=y_1 +1 - Количество состояний

// Вычисляем таблицы для
// динамического программирования - ДП

X = -%inf * ones(y9, n+1);
F = X;
F(:,n+1) = zeros(y9, 1); // что находится сейчас в F ?

for j = n : -1 :1
   Fyx = -%inf * ones(y9, x9);
   for y = 0 : y_1         // y - состояние  на шаге j
      for x = 1 : x9       // x - управление на шаге j
         yy = y - C(x,j);
         if 0 <= yy & yy <= y_1
            // Уравнение Ричарда Беллмана
            Fyx(1+y,x) = R(x,j) + F(1+y-C(x,j), j+1);
            // Fyx(1+ y,x) = R(x,j) + F(1+ yy, j+1);
         end
      end
   end
   // F,Fyx

   [cFy, cX] = max(Fyx, 'c');  // c - column (столбец)
   F(:,j) = cFy;
   X(:,j) = cX;

   // Печать шапки

   xmaxi = sum(bool2s(R(:,j) > -%inf));
   write(6, ' ')
   write(6, '       Этап ' + string(j)' )
   // tire40 =  '~~~~~~~~~~~~~~~~~~~|~~~~~~~~~~~~~~~';
   tire40 = part('~', ones(1,20)) + '|' + ...
   part('~', ones(1, 6*xmaxi ));

   write(6, tire40);
```

```
   write(6, '  Y' + string(j) + '   Fj(Yj)  Xj*  |     Fyx ');
   write(6, tire40);

   // Печать только нужных строчек и столбцов таблицы

   // Допустимые состояния на этапе j меняются
   // в пределах от ymini до ymaxi включительно

   ymini = sum(C(1, j:$));
   ymaxi = y_1 - sum(C(1, 1: j-1));

   Table = [(0 : y_1)', F(:,j), X(:,j), Fyx];

   write(6, Table(1+ (ymini:ymaxi), 1:(3+xmaxi)), ...
      '( 3(i4,2x), ""  |  "", 100(i4,2x) )' )
   write(6, tire40);
end

// Вычисление fopt, rXopt и rYopt
// ==============================

rXopt = zeros(1, n);  // r - row (строка)
rYopt = zeros(1, n);
y = y_1;

for j = 1 : n
   rXopt(j) = X(1+ y, j);
   rYopt(j) = y;
   y = y - C(rXopt(j), j);
end

// Печать ответа:
// ==============

C, R, y_1

write(6,' =========== Ответ: ============')
write(6,' ================================')

fopt = F(1+y_1, 1);
write(6,' fopt =  ' + string(fopt));
write(6, rXopt, '( "" rXopt ="", 100i6 )' )
write(6, rYopt, '( "" rYopt ="", 100i6 )' )
```

10.1.2. Полный перебор

```
// Программа InvestPe.sce
// ==== 4 января 2011 года ==== Визгунов Н.П.

mode(0), lines(0,80), clc, clear
y1 = 5
C = [   0    0    0    0
        1    3    1    2
      -%inf  5    2  -%inf ]
R = [   0    0    0    0
        3    5    4    3
      -%inf  9    6  -%inf ]

[u9,n] = size(C);   // u9 - число возможных управлений

x_min = ones(1,n)                // [ 1  1  1  1 ]
```

```
x_max = sum(abs(R) ~= %inf, 'r')  // [ 2  3  3  2 ]

x     = x_min;
x_opt = x_min;
f_opt = -%inf;

j = n;
while %t
   if  x(j) <= x_max(j)
      cost = 0;
      for i = 1 :n
         cost = cost + C(x(i),i);
      end
      if cost <= y1
         f = 0;
         for i = 1:n
            f = f + R(x(i),i);
         end
         if  f_opt < f
            // Для печати промежуточных данных убрать ;
            f_opt = f;
            x_opt = x;
         elseif f_opt == f
            f_opt;
            x_opt = [x_opt; x ];
         end
      end // cost
      j = n;
   else    // x(j) > x_max(j)
      x(j) = x_min(j);
      j = j - 1;
      if j <= 1e-8
         // Решение получено - выйти из цикла
         break
      end
   end // if
   x(j) = x(j) + 1;
end // while 1

write(6,[' *********'
         '   Ответ  '
         ' *********'])
write(6, ' f_opt = ' + string(f_opt))  // печать в одну строку
x_opt
```

10.2. Задача о загрузке

10.2.1. Динамическое программирование – вычислить C и R

```
// function RiukDiCR(c, a, b )
// Решение задачи о загрузке самолета методом ДП.
// ==== 30 марта 2011 года =====Визгунов Н.П.======

clc, clear, mode(0), lines(0)

c = [ 65  80  30 ];   // доход от перевозки предмета
a = [  2   3   1 ];   // вес этого предмета
b = 5; y1 = b;       // грузоподъемность самолета

n  = length(a);
s9 = y1 + 1;                // s9 - Кол. состояний
```

```
u9 = 1 + floor(b / min(a));  // u9 - Кол. управлений

// 1) Очевидный способ вычисления C и R

C = -%inf * ones(u9, n);
R = C;
for u = 1 : u9        // (u-1) - это количество предметов
   for j = 1 : n
      temp = a(j) * (u -1);  // j - номер предмета
      if temp <= b   // если предметы помещаются в самолет
         C(u, j) = temp;
         R(u, j) = c(j) * (u -1);
      end
   end
end
C; R;

// 2) Вычисления C и R в духе SciLab

C = (0 : u9 -1)' * a
I = find(C > y1)
C(I) = -%inf
R = (0 : u9 -1)' * c;
R(I) = -%inf

// Теперь - почти задача о распределении инвестиций

X = -%inf * ones(s9, n+1);
F = X;
F(:, n + 1) = zeros(s9, 1);

for j = n : -1 : 1
   Fsu = -%inf * ones(s9, u9);
   for s = 1 : s9          // Yj = s -1
      for u = 1 : u9       // Xj = u
         ss = s - C(u,j);
         if 1 <= ss & ss <= s9
            Fsu(s, u) = R(u, j) + F(ss, j +1);
         end
      end
   end
   [fs, index] = max(Fsu, 'c');
   F(:,j) = fs;
   X(:,j) = index - 1;

   // Печать шапки и таблицы
   // ======================

   u7 = 1 + floor(b / min(a(j)));
   line50 = part('-', ones(1, 22)) + '|' + ...
   part('-', ones(1, 3 + 6 * u7));

   // line50 = code2str(str2code('-') * ones(1,58));

   write(6, ' ')
   write(6, 'Этап ' + string(j))
   write(6, line50)    // напечатать 50 минусов
   write(6, '    Yj   Fj(Yj)  Xj*  |         Fsu');
   write(6, line50)

   Temp =[(0 : y1)',  fs,  (index - 1),  Fsu];
```

```
   disp(Temp(:, 1 : 3 + u7))

   write(6, line50)
end

// Вычислить и напечать ответ
// ==========================

x_opt = zeros(1, n);
s = s9;
for j = 1 : n
   x_opt(j) = X(s, j);
   s = s - a(j) * x_opt(j);
end
c, a, x_opt
write(6,' f_opt = ' + string(F(s9, 1)) + ...
        '  y1 = '   + string(y1))
```

10.2.2. Динамическое программирование – без вычисления C и R

```
// function RiukDi(c, a, b )
// Программа RiukDi.sce. Нет вычислений C и R !!!

// Решение задачи о загрузке самолета методом ДП.
// =====30 марта 2011 года=====Визгунов Н.П.=====

clc, clear, lines(0, 100), mode(0)

//доходы =
c = [ 65  80  30 ];
//вес предметов =
a = [  2   3   1 ];
//грузоподъемность самолета =
b = 5; y1 = b;

n  = length(a);
s9 = y1 + 1;                 // s9 - Кол. состояний
u9 = 1 + floor(b / min(a));  // u9 - Кол. управлений

X = -%inf * ones(s9, n+1);
F = X;
F(:, n + 1) = zeros(s9, 1);

for j = n : -1 : 1
   Fsu = -%inf * ones(s9, u9);
   for s = 1 : s9                // Yj = s -1
      for u = 1 : u9             // Xj = u
         ss = s - a(j) * (u -1); // - C(u,j) было
         if 1 <= ss & ss <= s9
            Fsu(s, u) = c(j) * (u - 1) + F(ss, j + 1);
         end
      end
   end
   [fs, index] = max(Fsu, 'c');
   F(:,j) = fs;
   X(:,j) = index - 1;

   // Печать шапки и таблицы
   // ======================

   u7 = 1 + floor(b / min(a(j)));
```

```
   line50 = part('-', ones(1, 22)) + '|' + ...
   part('-', ones(1, 3 + 6 * u7));

   // line50 = code2str(str2code('-') * ones(1,58));

   write(6, ' ')
   write(6, 'Этап ' + string(j))
   write(6, line50)    // напечатать 50 минусов
   write(6, '    Yj   Fj(Yj)  Xj*  |         Fsu');
   write(6, line50)

   Temp =[(0 : y1)',  fs,  index-1,  Fsu];
   disp(Temp(:, 1: 3+ u7))

   write(6, line50)
end

// Вычислить и напечать ответ
// ==========================

x_opt = zeros(1, n);
s = s9;
for j = 1 : n
   x_opt(j) = X(s, j);
   s = s - a(j) * x_opt(j);
end
c, a, x_opt
write(6,' f_opt = ' + string(F(s9,1)) + ...
        '  y1= '    + string(y1))
//endfunction
```

10.2.3. Полный перебор

```
// function RiukPe(c, a, b)
// программа RjukPe.sce  Лабораторная номер 3
// =====30 марта 2011 года=====Визгунов Н.П.

clc, clear, lines(0), mode(0)

c = [ 65  80  30 ]
a = [  2   3   1 ]
b = 5,  y1 = b;

n  = length(a);
s9 = y1 + 1;                  // s9 - Кол-во состояний
u9 = 1 + floor(b / min(a));
// u9 = 1 + floor(max(b ./ a));

x_min = zeros(1, n);   //  [0  0  0 ]
x = x_min;
x_max = floor(b ./ a); //  [2  1  5 ]

f_opt = -%inf;
j = n;
while %t
   if  x(j) <= x_max(j)

      if a * x' <= b    // если предметы войдут в самолет
         f = c * x';
         if f_opt < f
            disp('f_opt < f <<<<<<<<<<<<<')
```

```
            f_opt = f
            x_opt = x
            //disp(' ')
         elseif  f == f_opt
            disp('f_opt == f =============')
            f_opt
            x_opt = [x_opt; x]
         end
      end

      j = n;
   else    // x(j) > x_max(j)
      x(j) = x_min(j);
      j = j - 1;

      // закончить бесконечный цикл
      if j <= 1e-5
         break
      end

   end // if
   x(j) = x(j) + 1;
end // while %t

write(6,'ответ:')
write(6,'======')
x_opt, f_opt

//endfunction
```

10.3. Решение задачи о надёжности

10.3.1. Динамическое программирование

```
// Программа NadDi.sce
// Решение задачи о надёжности
// методом динамического программирования.
// ==== 24 марта 2011 года ==== Визгунов Н.П.

clear, clc, mode(0), lines(0,90)

y_1 = 10
C = [ 1    3     2
      2    5     4
      3    6     5  ]
R = [ 0.6  0.7   0.5
      0.8  0.8   0.7
      0.9  0.9   0.9]

[x9, n] = size(R);   // x9        - Количество управлений
y9      = y_1 + 1;   // y9=y_1 +1 - Количество состояний

// Вычисляем таблицы для
// динамического программирования (ДП)

X = -%inf * ones(y9, n+1);
F = X;
F(:,n+1) = ones(y9, 1); // что находится сейчас в F ?

for j = n : -1 :1
   Fyx = -%inf * ones(y9, x9);
```

```
   for y = 0 : y_1         // y - состояние  на шаге j
      for x = 1 : x9       // x - управление на шаге j
         yy = y - C(x,j);
         if 0 <= yy & yy <= y_1
            // Уравнение Ричарда Беллмана
            Fyx(1+y, x) = R(x, j) * F(1+y-C(x,j), j+1);
            // Fyx(1+ y,x) = R(x,j) + F(1+ yy, j+1);
         end
      end
   end
   // F,Fyx

   [cFy, cX] = max(Fyx, 'c');  // c - column, столбец
   F(:,j) = cFy;
   X(:,j) = cX;

   // Печать шапки

   xmaxi = sum(bool2s(R(:,j) > -%inf));
   write(6, ' ')
   write(6, '       Этап ' + string(j)' )
   // tire40 =  '~~~~~~~~~~~~~~~~~~~|~~~~~~~~~~~~~~~';
   tire40 = part('~', ones(1, 21)) + '|' + ...
   part('~', ones(1, 7 * xmaxi + 2 ));

   write(6, tire40);
   write(6, '   Y' + string(j) + ...
      '   Fj(Yj)  Xj*  |     Fyx ');
   write(6, tire40);

   // Печать только нужных строчек и столбцов таблицы

   // Допустимые состояния на этапе j меняются
   // в пределах от ymini до ymaxi включительно

   ymini = sum(C(1, j:$));
   ymaxi = y_1 - sum(C(1, 1: j-1));

   Table = [(0 : y_1)', F(:,j), X(:,j), Fyx];

   write(6, Table(1 +(ymini :ymaxi), 1 :(3 +xmaxi)), ...
      '( (f6.0, f7.3, f6.0 ), ""  |  "", 100(f7.3) )' )
   write(6, tire40);
end

// Вычисление fopt, rXopt и rYopt
// ==============================

rXopt = zeros(1, n);  // r - row, строка
rYopt = zeros(1, n);
y = y_1;

for j = 1 : n
   rXopt(j) = X(1+ y, j);
   rYopt(j) = y;
   y = y - C(rXopt(j), j);
end

// Печать ответа:
// ==============
```

```
C, R, y_1

write(6,' =========== Ответ: ============')
write(6,' ================================')

fopt = F(1 + y_1, 1);
write(6,' fopt =  ' + string(fopt));
write(6, rXopt, '( "" rXopt ="", 100i6 )' )
write(6, rYopt, '( "" rYopt ="", 100i6 )' )
```

10.3.2. Полный перебор

```
// Программа NadPe.sce
// Решение задачи о надёжности
// методом полного перебора.
// ==== 24 марта 2011 года ==== Визгунов Н.П.

clear, clc, mode(0), lines(0,90)

y_1 = 10
C = [ 1    3     2
      2    5     4
      3    6     5  ]
R = [ 0.6  0.7   0.5
      0.8  0.8   0.7
      0.9  0.9   0.9]

[u9, n] = size(C);   // u9 - число возможных управлений

x_min = ones(1,n)                 // [ 1  1  1  1 ]
x_max = sum(abs(R) ~= %inf, 'r')  // [ 3  3  3  3 ]

x     = x_min;
x_opt = x_min;
f_opt = -%inf;

j = n;
while %t
   if  x(j) <= x_max(j)
      cost = 0;
      for i = 1 :n
         cost = cost + C(x(i),i);
      end
      if cost <= y_1
         f = 1;
         for i = 1:n
            f = f * R(x(i),i);
         end
         if  f_opt < f
            // Для печати промежуточных данных убрать ;
            f_opt = f;
            x_opt = x;
         elseif f_opt == f
            f_opt;
            x_opt = [x_opt; x ];
         end
      end // cost
      j = n;
   else    // x(j) > x_max(j)
      x(j) = x_min(j);
      j = j - 1;
```

```
      if j <= 1e-8
         // Решение получено - выйти из цикла
         break
      end
   end // if
   x(j) = x(j) + 1;
end // while 1

write(6,[' *********'
         '   Ответ  '
         ' *********'])
write(6, ' f_opt = ' + string(f_opt))  // печать в одну строку
x_opt
```

10.4. Календарное планирование трудовых ресурсов

10.4.1. Динамическое программирование

```
// function [f_opt, y_opt] = TrudD(y0, y_max, b)
// программа TrudDi.sce Динамическое программирование
//============Визгунов Н.П. 24 марта 2011============

clc, clear, lines(0), mode(0)

y_max = [8  8  8  6  6 ];
b     = [5  7  8  4  6 ];
y0 = 5;

n  = length(b);
s9 = max(y_max - b + 1);

F = %inf * ones(s9, n + 1);
Y = F;
F(:, n + 1) = zeros(s9, 1);

for j = n : -1 : 1   // j - номер этапа
   s7 = 1;
   if j >= 2
      s7 = y_max(j - 1) - b(j - 1) + 1;
   end

   u7 = y_max(j) - b(j) + 1;
   Fsu = %inf * ones(s7, u7);

   for s = 1 : s7
      Yj_1 = y0;
      if j >= 2
         Yj_1 = s + b(j - 1) - 1;
      end
      for u = 1 : u7
         Yj = u + b(j) - 1;
         Cn = 0;
         if Yj > Yj_1
            Cn = 4 + 2 * (Yj - Yj_1);
         end
         Fsu(s,u) = 3 * (Yj - b(j)) + Cn + F(u, j +1);
      end  // for u
   end  // for s

   // обработка временной таблицы Fsu
```

```
   [fs, index] = min(Fsu, 'c');
   F(1 : s7, j) = fs;
   Y(1 : s7, j) = index;

   // вычисление и печать шапки

   tire = part('=', ones(1,20)) + '|' + ...
      part('=', ones(1, 6 * u7 ));
   kepka = '   Y' + string(j - 1) + ' F(:,j) Y(:,j) |';
   for u = 1 : u7
      kepka = kepka + ' Y' + string(j) + ...
         '=' + string(b(j) + u - 1);
   end
   write(6, ' ')
   write(6, '    Этап ' + string(j))
   write(6, tire)
   write(6, kepka)
   write(6, tire)

   // печать таблицы

   if j >= 2   // из-за b(0) надо разделять случаи
      disp([(b(j - 1) : b(j - 1) + s7 - 1)', ...
         fs, index + b(j) - 1, Fsu])
   else  // j == 1
      disp([y0, fs, index + b(j) - 1, Fsu])
   end
   write(6, tire)
end

// вычисление и печать результатов
// ================================

y_opt = zeros(1, n);
s = 1;
for j = 1 : n
   s = Y(s,j);
   y_opt(j) = s;
end // for j

disp('**********ответ ДП***********')
write(6, ' f_opt  = ' + string(F(1, 1)))
y_opt = y_opt + b - 1

//endfunction
```

10.4.2. Полный перебор

```
// function [f_opt, y_opt] = TrudPe(y0, y_max, b)
// программа TrudPe.sce Визгунов Н.П.
// ======= 24 марта 2011г. =======

mode(0), lines(0), clc, clear

y_max = [8  8  8  6  6 ];
b     = [5  7  8  4  6 ];
y0 = 5;

n     = size(b,'c');
y_max0 = [y0,  y_max];
b0     = [y0,  b    ];
```

```
n = n +1

f_opt = %inf;  //задача на min, поэтому +%inf
y_opt = zeros(1, n +1);
y = b0; j = n;

write(6,' ========полный перебор========')

while %t
   if y(j) <= y_max0(j)

      f = 0;
      for j = 2 : n
         Cn = 0;
         if y(j) > y(j -1)
             Cn = 4 + 2 * (y(j) - y(j - 1));
         end
         f = f + 3 * (y(j) - b0(j)) + Cn;
      end //for j

      if  f_opt > f //задача на min !
         write(6, '>>>>>>>>>>>>>>>>>>>>>>>>>')
         f_opt = f
         y_opt = y(2 : n)
      elseif f_opt == f
         write(6, '=========================')
         f_opt
         y_opt = [y_opt; y(2 : n)]
      end //elseif

      j = n;
   else      //y(j) > y_max(j)
      y(j) = b0(j);
      j = j - 1;
      if j <= 0.0001
         break
      end  //if
   end  //if
   y(j) = y(j) + 1;
end  //while %t

write(6, [' ==========='
          '    ответ'
          ' ==========='])
write(6, ' f_opt равно '  + string(f_opt))
write(6, ' y0    равно  ' + string(y0))
y_opt
b = b0(2 : n)
//endfunction [f_opt,y_opt]=TrudPP(y0,y_max,b)
```

10.5. Управление запасами

10.5.1. Динамическое программирование

```
// программа ZapasDi.sce Динамическое программирование,
// задача управления запасами.
// Визгунов Н.П. 25 марта 2011 года
// ================================

clc, clear, mode(0), lines(0)
```

```
//x = 1  2  3  4  5
c1 = [0  7  9 11 13]; // Затраты на пр-во x-1 ед. продукции
x9 = length(c1) - 1;  // Максимум производства в штуках
y0 = 2;             // Запасы на складе в конце декабря
y9 = 4;             // Максимум запасов на складе в штуках
k  = 1;             // Плата за единицу хранения на складе

d = 2;              // Отгрузка потребителям
n = 4;              // Количество месяцев в плановом периоде

s9 = y9 + 1;        // Макс. количество состояний
u9 = x9 + 1;        // Макс. количество управлений

F = %inf * ones(y9 + 1, n);
X = F;

// Начальные вычисления для этапа n.

for s = 1 : s9
   u = d - s + 2;
   if u >= 1
      F(s, n) = c1(u);
      X(s, n) = u;
   end
end  // for s

// Печать для этапа n

j = n;
write(6, '   Этап ' + string(j))
tire = part('=', ones(1, 19)) + '|';
kepka = '   Y' + string(n - 1) + ' F(:,j) Y(:,j) |';

s7 = sum(F(:, n) < %inf );

write(6, tire)
write(6, kepka)
write(6, tire)
disp([(0 : s7 -1)', F(1 : s7, n), X(1 : s7, n) - 1])
write(6, tire)

// Произвольный j-ый этап.

Fsu = %inf * ones(s9, u9);
for j = n -1 : -1 : 1
   for s = 1 : s9          // s = Yj_1 + 1
      for u = 1 : u9       // u = Xj + 1
         s1 = (s -1 ) + (u - 1) - d + 1;
         if  1  <= s1  & s1 <= n  + 1
            Fsu(s, u) = c1(u) + k * (s1 - 1) + ...
                        F(s1, j + 1);
         end
      end  // for u
   end  // for s
   [Fs, Index] = min(Fsu, 'c');

   // Обработка таблиц Fsu   F(состояние, управление)

   F(:, j) = Fs;
   X(:, j) = Index;
```

```
   // вычисление и печать шапки

   tire = part('=', ones(1,19)) + '|' + ...
          part('=', ones(1, u9 * 7));
   kepka = '   Y' + string(j - 1) + ' F(:,j) Y(:,j)|';
   for u = 1 : u9
      kepka = kepka + '   X' + string(j) + ...
                      '=' + string(u - 1);
   end
   write(6, ' ')
   write(6, '   Этап ' + string(j))
   write(6, tire)
   write(6, kepka)
   write(6, tire)
   disp([(0 : s9 - 1)', F(:, j), X(:, j) - 1, Fsu])
   write(6, tire)
end  // for j

// печать результатов
// ==================

x_opt = zeros(1, n);
s = y0 + 1;
for j = 1 : n
   x_opt(j) = X(s, j);
   s = (s - 1) + X(s, j) - d ;
end  // for j

x_opt = x_opt - 1
f_opt = F(y0 + 1, 1)
```

10.5.2. Полный перебор

```
// Программа ZapasPe.sce
// =====================
// Управление запасами - полный перебор.
// ======== Визгунов Н.П. 24 марта 2011.

clc, clear, lines(0), mode(0)

// x:  1  2  3  4  5
c1 =  [0  7  9 11 13]; // Затраты на пр-во x-1 ед. продукции
x9 = length(c1) - 1;   // Максимум производства в штуках
y0 = 2;             // Запасы на складе в конце декабря
y9 = 4;             // Максимум запасов на складе в штуках
k  = 1;             // Плата за единицу хранения на складе

d = 2;              // Отгрузка потребителям
n = 4;              // Количество месяцев в плановом периоде

s9 = y9 + 1;        // Макс. количество состояний
u9 = x9 + 1;        // Макс. количество управлений

x = zeros(1, n);
y = zeros(1, n);
f_opt = %inf;
x_opt = x;

j = n;
while %t
   if x(j) <= x9
```

```
      y(1) = y0 + x(1) - d;
      for j = 2 : n
         y(j) = y(j - 1) + x(j) - d;
      end  // for

      if  0 <= min(y) & max(y) <= y9

         f = 0;
         for j = 1 : n
            f = f + c1(x(j) + 1) + k * y(j);
         end
         if  f_opt == f
            write(6, '============ f_opt = ' + string(f_opt))
            x_opt = [x_opt; x]
            y_opt = [y_opt; y];
         elseif f_opt > f
            f_opt = f;
            write(6, '>>>>>>>>>>>> f_opt = ' + string(f_opt))
            x_opt = x
            y_opt = y;
         end
      end  // if

      j = n;
   else
      x(j) = 0;
      j = j - 1;
      if j <= 0.0001
         break
      end  // if
   end  // if

   x(j) = x(j) + 1;
end

disp('******* Ответ: *******')
disp('f_opt  = ' + string(f_opt))
x_opt, y_opt
```

10.6. Замена оборудования

10.6.1. Динамическое программирование

```
// function [] = ZamenaDi(r1, c1, Cn, y1, n)
// программа ZamenaDi.sce Динамическое программирование
// Замена оборудования на 5 лет или замена автомобиля
// Визгунов Н.П 25 марта 2011  ============
mode(0), lines(0), clear, clc

// s :  1  2  3  4  5
r1 = [80 75 65 60 60]; // Доход от авто возраста s-1 лет
c1 = [20 25 30 35 45]; // Затраты на него
Cn = 40;               // Цена нового авто
y1 = 0;                // В начале 1971 года авто - новый
n = 5;                 // Кол. месяцев в плановом периоде
if y1 + length(c1) > n
   error('не хватает данных')
end

Rnew = r1(1) - c1(1) - Cn;  // Прибыль от нового авто
```

```
Fsu  = %inf * ones(n ,2);
X = %inf * ones(n ,n +1);
F = X;
F(:, n +1) = zeros(n, 1);
Tire = part('-', ones(1, 20)) + '|' + ...
part('-', ones(1, 16));

for j = n : -1 : 2
   for Yj = 1 : j -1   // s = Yj
      for Xj = 0 : 1   // u = Xj +1
         if Xj == 0    // старая машина
            Fsu(Yj,1) = r1(Yj + 1) - c1(Yj + 1) + ...
            F(Yj + 1, j + 1);
         else   // новая Xj == 1
            Fsu(Yj, 2) = Rnew + F(1, j + 1);
         end  // if
      end  // for Xj
   end  // for Yj
   [Fs, Index] = max(Fsu, 'c');
   F(1 : j - 1, j) = Fs(   1 : j - 1);
   X(1 : j - 1, j) = Index(1 : j - 1) - 1;

   // Печать шапки и таблицы

   write(6, ' ')
   write(6, '    Этап ' + string(j))
   Sj = string(j);
   Kepka = '    Y' + Sj + '     F     X* |';
   Kepka = Kepka + '  X' + Sj + '=old  X' + Sj + '=new ';

   write(6, Tire);
   write(6, Kepka);
   write(6, Tire);
   disp([(1 : j -1)', Fs(1 : j -1), ...
      Index(1: j -1)-1, Fsu(1: j -1, :)])
   write(6, Tire)
end  // for j

// Вычисление и печать результатов
// ===============================

Yj = 1;
for j = 1 : n
   if X(Yj, j) == 0  // старый авто
      Yj = Yj + 1;
      x_opt(j) = 'ста.  ';
   else              // новый  авто
      Yj = 1;
      x_opt(j) = 'нов.  ';
   end  // if
end  // for

disp('x_opt =')
disp(x_opt')

f_opt = F(1,2) + r1(1) - c1(1)
```

10.6.2. Полный перебор

```
//   программа ZamenaPe.sce
// ======================
```

```
// Замена оборудования - полный перебор.
// Визгунов Н.П. 25 марта 2011 года

clc, clear, mode(0), lines(0)

// s:  1  2  3  4  5
r1 = [80 75 65 60 60]; // Доход от авто возраста s - 1 лет
c1 = [20 25 30 35 45]; // Затраты на него
Cn = 40;               // Цена нового авто
y1 = 0;                // В начале 1971 года авто - новый

n = 5;                 // Кол-во месЯцев в плановом периоде
if y1 + length(c1) > n
    error('не хватает данных')
end
// Rnew = r1(1) - c1(1) - Cn;// Прибыль от нового автомобилЯ

f_opt = -%inf;
x_opt = zeros(1, n + 1);

x_max =  ones(1, n);
x_min = zeros(1, n);

x = zeros(1, n);
j = n;
while %t
    if x(j) <= x_max(j)

       for j = 1 : n
            if x(j) == 1
                y(j) = 0;
            elseif x(j) == 0 & j >= 2
                y(j) = y(j - 1) + 1;
            elseif x(1) == 0
                y(1) = y1;
            end
        end  // j

        f = Cn;
        for j = 1 : n
            if x(j) == 0 & y(j) >= 1
                f = f + r1(y(j) + 1) - c1(y(j) + 1);
            elseif x(j) == 1
                f = f + r1(1) - c1(1) - Cn;
            end
        end

        if  f_opt < f
            f_opt = f;
            write(6, f_opt, '( ""<<<<<< f_opt = "", i6 )' )
            x_opt = x
        elseif f_opt == f
            write(6, f_opt, '( ""====== f_opt = "", i6 )' )
            x_opt = [x_opt; x]

        end  // elseif

        j = n;
    else     // x(j) > x_max(j)
        x(j) = x_min(j);
        j = j - 1;
```

```
        if j <= 0.00001
            break
        end  // if
    end  // if
    x(j) = x(j) + 1;
end  // while %t

disp('ответ')
x_opt, f_opt
```

Список литературы

[1] *Introduction to SciLab*, Consortium SciLab, Nomember, 2010 introscilab.pdf

[2] Акулич И.Л. *Математическое программирование в примерах и задачах: Учеб. пособие для студентов эконом. спец. вузов.* — М.: Высшая школа, 1986. — 319с.

[3] Алексеев Е. Р. *Scilab: Решение инженерных и математических задач / Е.Р.Алексеев, О.В.Чеснокова, Е.А.Рудченко.* — М. : ALT Linux, БИНОМ. Лаборатория знаний, 2008. — 260с.

[4] Зайченко Ю.П. *Исследование операций: Учеб. пособие для студентов вузов.— 2-е изд. перераб. и доп.* — Киев.: Вища школа. Головное изд-во, 1979. — 392с.

[5] *Исследование операций в экономике: Учебное пособие для вузов/ Н.Ш.Кремер, Б.А.Путко, И.М.Тришин, М.Н.Фридман; Под ред. проф. Н.Ш.Кремера.* — М.: Банки и биржи, ЮНИТИ, 1997. — 407с.

[6] Таха Х. *Введение в исследование операций: В 2-х книгах. Кн.1. Пер. с англ.* — М.: Мир, 1985. — 479с.

[7] Таха, Хэмди, А. *Введение в исследование операций, 6-е издание: Пер. с англ.* — М.: Издательский дом «Вильямс» , 2001. — 912с.