МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра вычислительной техники

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«16» декабря 2016 г.

МНОГОЭТАПНЫЕ ДЕТЕРМИНИРОВАННЫЕ ЗАДАЧИ ПРИНЯТИЯ РЕШЕНИЙ

Методические указания к практическим занятиям по дисциплине «Теория принятия решений» для студентов направления подготовки 09.03.01 Информатика и вычислительная техника

Курск 2016 г.

УДК 519.81
Составители: С.В. Дегтярев, Е.Н. Иванова

Рецензент
Профессор кафедры биомедицинской инженерии,
доктор технических наук
С.А. Филист

Многоэтапные детерминированные задачи принятия решений: методические указания к практическим занятиям / Юго-Зап. гос. ун-т; сост.: С.В. Дегтярев, Е.Н. Иванова. – Курск, 2016. – 30 с.: ил. 7, табл. 13. – Библиограф.: с. 30.

Дана графическая интерпретация модели решения многоэтапной задачи принятия решения. Приведен метод динамического программирования решения многоэтапных задач принятия решения с аддитивным и минимаксным критериями. Рассмотрен принцип Беллмана.

Методические указания соответствуют требованиям программы, утвержденной учебно-методическим объединением по направлению Информатика и вычислительная техника.

Предназначены для студентов направления 09.03.01 Информатика и вычислительная техника дневной формы обучения.

Текст печатается в авторской редакции

Подписано в печать 16.12.16 . Формат 60x84 1/16. .
Усл.печ.л. 1,4. Уч.-изд.л. 1,6. Тираж 50 экз. Заказ 1232. Бесплатно.
Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

1. Цель

Научиться определять компоненты задачи динамического программирования. Освоить методы решения многоэтапных детерминированных задач принятия решения. Получить представление о принципе Беллмана.

2. Математическая формулировка многоэтапной детерминированной задачи принятия решений

Понятие многостадийной (многоэтапной, многошаговой) задачи принятия решений весьма многогранно. Поэтому могут рассматриваться совершенно различные модели многостадийности от простых до достаточно сложных. Мы будем придерживаться следующего определения: многошаговая задача принятия решений — это процесс принятия последовательных решений, направленных на достижение единой цели. Например, часто цель всей операции заключается в максимизации „полезности“ (скорости, производительности) или минимизации „затрат“ (времени, объемов памяти). Предполагается, что получение „полезности“ реализуется на каждом этапе процесса принятия решений, а затем эти „полезности“ суммируются (принцип аддитивности). Представляют интерес и задачи, в которых цель заключается в минимизации (максимизации) требуемого критерия на каждом этапе (минимаксные задачи).

Постановка многоэтапной задачи принятия решения в условиях определенности в самом общем виде может быть представлена следующим логическим высказыванием

$$\langle S_0,\, S_m,\, W,\, U,\, U^* \in U \rangle, \qquad (1)$$

где S_0 – множество исходных состояний;
S_m – множество конечных (целевых) состояний;
W – критерий оптимальности ($W = \sum_{i=1}^{m} w_i(U)$ — суммарная

полезность или $W = \min_i w_i(U)$ — минимаксная полезность);
U – вектор управления;
U^* – выделенный с помощью критерия W оптимальный (наиболее предпочтительный) вектор управления.

Логическое высказывание (1) означает, что из множества векторов U по значениям скалярного показателя качества W выделяется оптимальный вектор U^*, переводящий объект принятия решения из начального состояния, принадлежащего множеству S_0, в конечное состояние, принадлежащее множеству S_m.

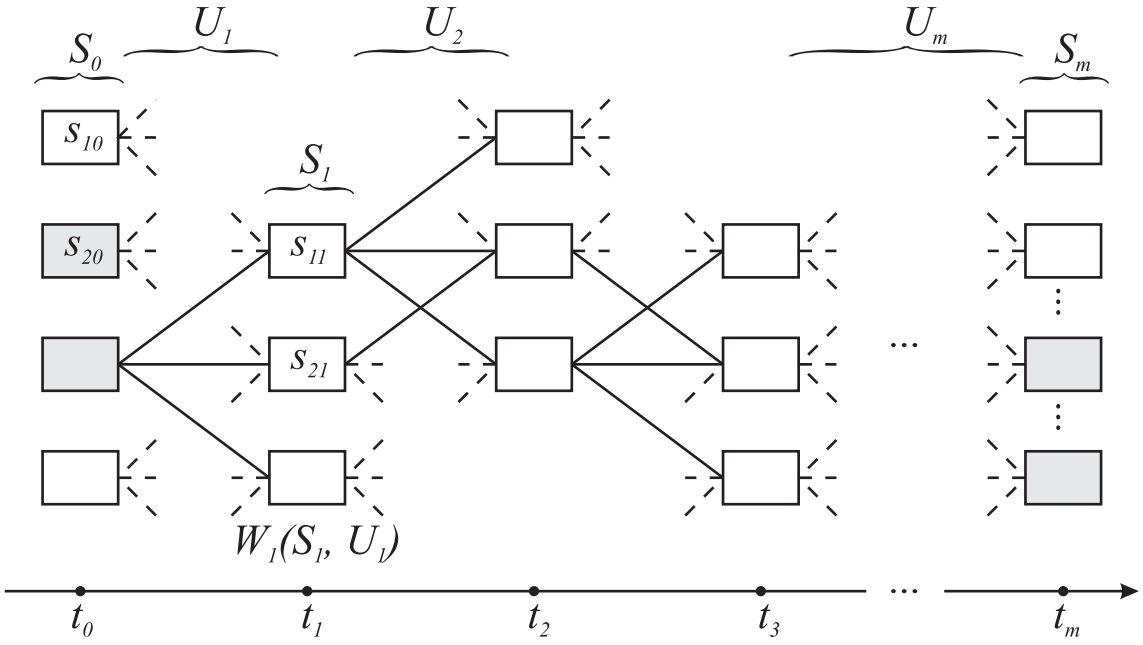

Рисунок 1 – Дерево решений в условиях определенности

Модель решения многостадийной задачи принятия решений может быть проиллюстрирована графом, называемым деревом решений и, по существу, описывающим то, как можно перейти из заданного множества его начальных вершин S_0 в заданное множество его конечных вершин S_m. При этом с каждой вершиной графа ассоциируется некоторое состояние s_{ij}, в котором находится объект принятия решений, а дуги, выходящие из вершины, соответствуют возможным переходам из одного состояния в другое в зависимости от принимаемых решений.

На рисунке 1 дан пример так называемого детерминистского дерева решений. Здесь предполагается, что процесс разворачивается во времени и движение по графу осуществляется слева направо. Допустимые начальные и конечные вершины заштрихованы. Считается, что каждая ветвь графа имеет свой вес — вещественное число, означающее соответствующее локальное значение „затрат“ („полезности“) на переход в другое состояние. Основная задача состоит в оптимальном выборе начальной вершины (из множества допустимых) и пути из нее в любую из допустимых конечных вершин. Оптимальность понимается в смысле построения допустимого пути, реализующего минимальные суммарные затраты (максимальную суммарную полезность). В частном случае множества допустимых начальных и конечных вершин могут быть одноэлементными (одноцелевые задачи).

3. Метод динамического программирования для решения многостадийных задач принятия решений

Совокупность процедур, используемых для оптимизации многошаговых процессов принятия решений, носит название динамического программирования (метод Беллмана). Динамическое программирование применяется для анализа систем, которые характеризуются следующими признаками:
процесс функционирования системы включает последовательные этапы — текущий этап имеет номер i, конечный этап — номер m;
на i-м этапе управление U_i переводит систему из состояния S_{i-1}, достигнутого на $(i-1)$-м этапе, в состояние S_i;
для анализируемой системы выполняется принцип отсутствия последействия, состоящий в том, что состояние S_i зависит только от состояния для предыдущего этапа S_{i-1} и управления U_i, но не зависит от состояний S_{i-2}, S_{i-3}, ..., S_1 и

управлений U_{i-1}, U_{i-2}, ..., U_1.

Известна функция локального дохода $w_i(S_i, U_i)$, определяющая значение полезности, получаемой при применении на i-м этапе управления U_i в случае нахождения системы в состоянии S_i. Доход от функционирования системы за m этапов равен сумме локальных доходов, полученных на каждом из этапов. Необходимо найти такой вектор управлений $U = (U_1, U_2, ..., U_i, ..., U_m)$, который обеспечивает максимизацию суммарноq „полезности“ — $W = \sum_{i=1}^{m} w_i(S_i, U_i) \to max$ или максимизацию „полезности“ каждого этапа — $W = \max_i w_i(S_i, U_i)$.

Для решения такого класса задач могут использоваться два подхода. Первый подход предполагает, что одновременно находятся компоненты вектора управления для всех этапов решения. Такой подход глобальной оптимизации приводит к существенному возрастанию трудоемкости получения решения. Второй подход базируется на нахождении оптимального управления последовательно для каждого из этапов.

Классическая идея динамического программирования состоит в том, что в качестве этапа, для которого на первом шаге находится локальное оптимальное управление, рассматривается последний этап принятия решений. Очевидно, что решения, принятые на этом этапе, не оказывают влияния на последующий этап, т.к. этот этап является заключительным. Однако при таком подходе является неизвестным состояние, в котором будет находиться система перед началом выполнения последнего этапа. Поэтому локальное оптимальное управление необходимо найти для всех возможных состояний, в которых может находиться система, перед выполнением последнего этапа. После этого осуществляется переход к оптимизации управления на предпоследнем этапе.

Оптимальное управление на этом этапе находится с учетом того, что уже известно оптимальное управление на последнем этапе. Таким образом, для каждого состояния предпоследнего этапа находится локальное оптимальное управление. Такая процедура повторяется для всех последующих этапов вплоть до первого этапа, т.е. оптимальное решение для первого этапа определяется с учетом ранее найденных оптимальных решений для последующих этапов. Принцип решения локальной задачи, начиная с последнего этапа, был предложен Беллманом.

Принцип Беллмана. Каково бы ни было состояние системы на m-м этапе, управление должно быть выбрано из условия, что и на последующих этапах управление будет оптимальным. Первоначальная формулировка принципа оптимальности, предложенная Р. Беллманом: „Оптимальное поведение обладает тем свойством, что каковы бы ни были первоначальное состояние и решение в начальный момент, последующие решения должны составлять оптимальное поведение относительно состояния, получающегося в результате первого решения“. Один из вариантов современной трактовки представляется следующим образом: допущенную ошибку на начальных этапах нельзя исправить на последующих этапах принятия решений.

Общая схема решения задачи на основе принципа Беллмана предполагает формальное определение таких компонент как этап, состояние, управление, оператор перехода, локальный доход на i-м этапе, условный оптимальный доход для i-го этапа.

Этапы при решении задачи должны быть перечислены и определены. Необходимо помнить, что количество этапов конечно — m. Качественное определение этапов зависит от природы управляемой системы, и они связаны либо с временной, либо с логической последовательностью принятия

решений.

Состояние — это такой набор параметров, который однозначно характеризует поведение системы. Общее требование к состоянию — это сохранение в нем информации о предыстории процесса, или иначе, состояние должно быть описано с той степенью подробности, которая позволяет провести оценку текущих альтернативных решений. Состояние в общем случае обозначается как S.

Управление U_i — это целенаправленное воздействие на управляемую систему на i-м этапе принятия решения. Условное оптимальное управление $U_i(S_j)$ — это наилучшая стратегия для каждого из состояний. При этом система переходит из одного состояния в другое.

Оператор перехода предназначен для установления связи между состояниями для соседних этапов принятия решений — $S_k = \varphi(S_j, U_i)$, здесь φ — оператор перехода, задающий переход из состояния S_j в состояние S_k при использовании на i-м этапе стратегии U_i.

Локальная полезность на i-м этапе $w_i(S_j, U_i)$ — это величина полелзности, получаемая от функционирования системы на i-м этапе, при условии, что она находилась в состоянии S_j, и была выбрана стратегия U_i.

Условный оптимальный доход $w_i(S_j)$ — это суммарный или минимаксный оптимальный доход, полученный от функционирования системы на i-м, $(i+1)$-м, ..., m-м этапах, при условии, что на i-м этапе система находилась в состоянии S_j.

Тогда функциональное уравнение Беллмана для аддитивного критерия представляется следующим образом: условный оптимальный доход на i-м этапе для состояния S_j — это максимум из суммы локального дохода на i-м этапе и условного оптимального дохода для состояния S_k, вычисленного для следующего $(i+1)$-го этапа:

$$W_i(S_j) = \max_{U_m} \{w_i(S_j, U_i) + W_{i+1}(\varphi(S_j, U_i))\};$$

для минимаксного критерия: условный оптимальный доход на i-м этапе для состояния S_j — это минимум из максимумов локальных доходов на i-м этапе, вычисленных с учетом условного оптимального дохода для состояния S_k для следующего $(i+1)$-го этапа:

$$W_i(S_j) = \min_{U_i} \max \{w_i(S_j, U_i), W_{i+1}(\varphi(S_j, U_i))\}.$$

Далее рассмотрим механизм решения только для задач с аддитивным критерием. В соответствии с алгоритмом обратной прогонки решение функционального уравнения начинается с последнего этапа принятия решений, т.е. i полагается равным m. Тогда

$$W_m(S_j) = \max_{U_m} \{w_m(S_j, U_i)\}.$$

В результате решения этого уравнения находится условное оптимальное управление $U_m(S_j)$ для каждого состояния из множества S_j и условный оптимальный доход $W_m(S_j)$. Найденные значения используются для решения функционального уравнения Беллмана при $i = m-1$:

$$W_{m-1}(S_j) = \max_{U_{m-1}} \{w_{m-1}(S_j, U_{m-1}) + W_m(\varphi(S_j, U_{m-1}))\}.$$

В результате решения этого уравнения для состояния S_j на $(m-1)$-м этапе находятся $U_{m-1}(S_j)$ и $W_{m-1}(S_j)$.

Таким образам, последовательное решение уравнения Беллмана позволяет найти пары: $U_i(S_j)$, $W_i(S_j)$, $i = \overline{1, m}$. Затем задается начальное состояние S_0 и для этого состояния определяется оптимальное управление $U_1(S_0)$. После этого находится оптимальное состояние $S_1 = \varphi(S_0, U_1(S_0))$. Для найденного состояния S_1 определяется оптимальная стратегия $U_2(S_1)$, а затем с помощью оператора перехода вычисляется оптимальное состояние S_2 и т.д.

Таким образом, формируется цепочка:

$$S_0 \to U_1(S_0) \to S_1 = \varphi(S_0, U_1(S_0)) \to U_2(S_1) \to S_2 \ldots U_m(S_{m-1}) \to S_m$$

Эта цепочка содержит оптимальные последовательности состояний и стратегий.

В задаче динамического программирования при использовании схемы прямой прогонки функциональное уравнение Беллмана имеет следующий вид:

$$W_i(S_j) = \max_{U_i} \left\{ w_i(S_j, U_i) + W_{i-1}\left(\phi^{-1}(S_j, U_i)\right) \right\},$$

где ϕ — оператор обратного перехода, устанавливающий состояние для предыдущего этапа.

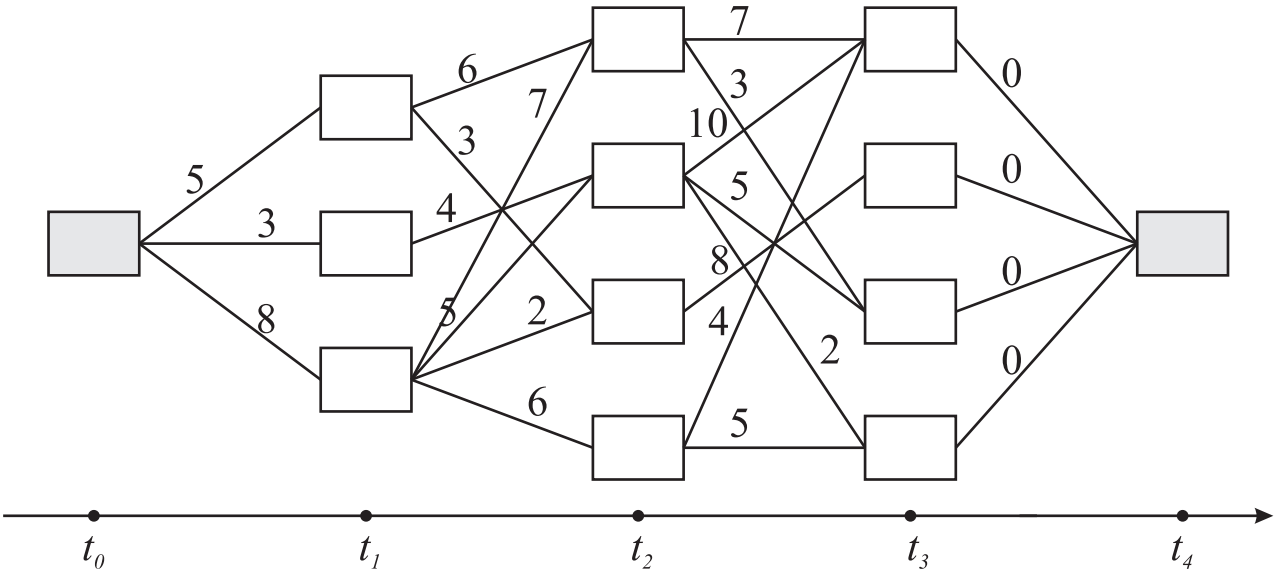

Рисунок 2 – Дерево решений с заданными локальными затратами

Проиллюстрируем метод Беллмана графически. На рисунке 2 приведен пример детерминистского дерева решений. Здесь одна начальная и одна конечная вершина. В случае, если конечных вершин несколько, с помощью введения фиктивной вершины всегда можно свести задачу к графу с одной конечной вершиной. Точно так же можно поступать и с начальными вершинами, если их несколько.) Числа у дуг графа означают трудоемкости (затраты), обеспечивающие переход от одной „решающей“ вершины к другой. Дуги на промежутке $[t_3, t_4]$ имеют нулевые веса, что и означает, что

все вершины, отвечающие t_3, являются конечными. Главная задача заключается в выборе оптимального в смысле суммарных затрат пути, соединяющего начальную и конечную вершины.

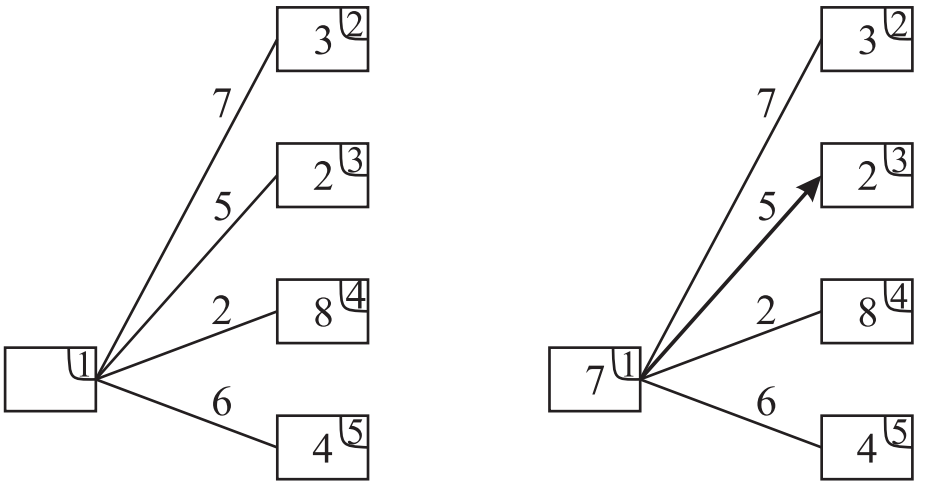

Рисунок 3 – Принцип разметки дерева решений

Реализуем метод Беллмана для приведенного примера. Будем продвигаться по графу справа налево, при этом:
нумеровать вершины в верхнем правом углу;
определять оптиальные направления движения из каждой вершины с помощью одной или нескольких стрелок;
указывать внутри квадратов-вершин суммарные затраты, которые получаются при движении из вершины, выбранной в качестве начальной, по оптимальному пути (т.е. это наименьшие из возможных затрат).

Например, мы имеем ситуацию, изображенную на рисунке 3, *а*). Вершины 2, 3, 4, 5 уже размечены — для каждой из вершин определены затраты. Надо пометить вершину 1. Будем рассуждать следующим образом.

Поиск оптимального пути из вершины 1 сводится к сравнению четырех чисел: $3+7=10$, $2+5=7$, $8+2=10$, $4+6=10$. Наименьшее из этих чисел — $\min\{10;\ 7;\ 10;\ 10\} =$

7, и, следовательно, именно число 7 мы запишем в вершине 1, а стрелочкой соединим вершины 1 и 3 (рисунок 3, *б*)). В трех других возможных случаях мы имеем большие затраты и, следовательно, оптимальный маршрут из вершины 1 лежит через вершину 3. Продвигаясь справа налево, мы, обрабатывая последовательно вертикальные слои вершин, разметим весь граф (рисунок 4).

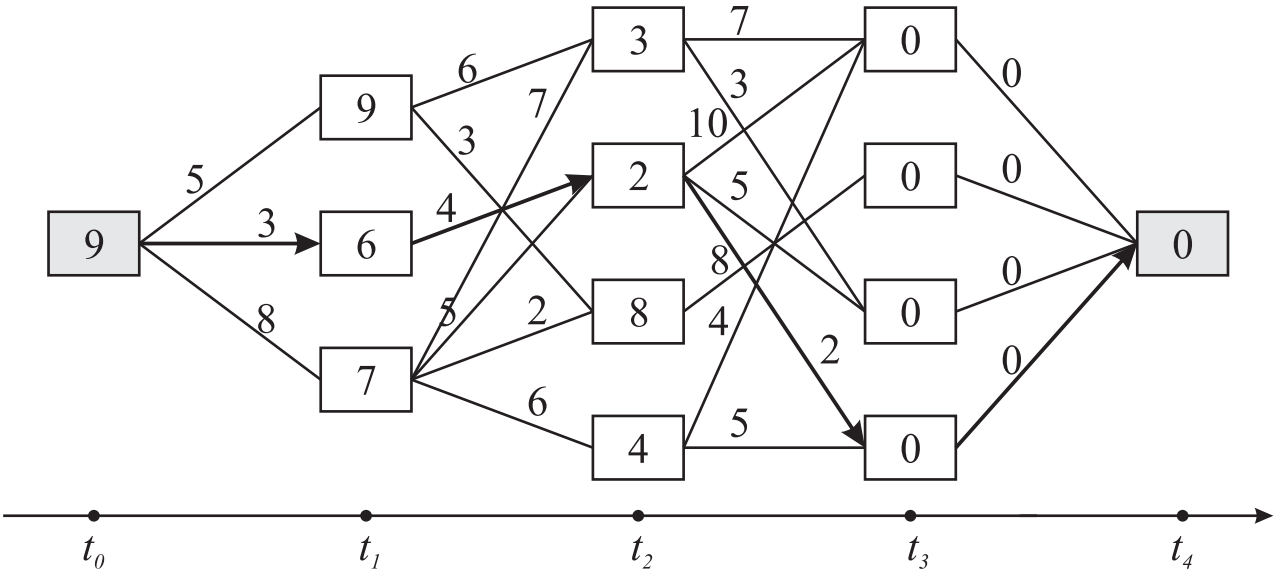

Рисунок 4 – Размеченный граф

Для восстановления искомого оптимального пути достаточно пройти теперь уже слева направо в направлении стрелок, начиная из уже помеченной начальной вершины (оптимальный путь на рисунке 4 выделен). Число 9 означает минимальные возможные затраты и само оно получается уже на первом этапе разметки графа. Оптимальный путь, очевидно, может быть и не единственным. Согласно методу Беллмана одновременно находятся все возможные оптимальные пути.

4. Многостадийные задачи принятия решений

4.1. Минимаксная задача оптимального распределения программных модулей между процессорами

В рассмотренных ранее постановках предполагалось, что критерий оптимальности является аддитивным, т.е. в соответствии с этим критерием суммируются доходы, полученные на различных этапах. Современная трактовка метода динамического программирования позволяет находить оптимальное решение не только для аддитивных критериев, но и для минимаксных. В частности, предположим, что в результате проектирования информационной системы выделено множество программных модулей $R = \{R_1, \ldots, R_i, \ldots, R_m\}$. Эти модули являются информационно-независимыми друг от друга и могут параллельно выполняться на многопроцессорной вычислительной системе. Многопроцессорная система содержит D_0 процессоров. Для каждого из программных модулей определены варианты их реализации, которые формально задаются переменной d_{ij}, определяющий количество процессоров, которые могут использоваться для выполнения R_i-го программного модуля в j-м варианте. Например, $d_{i1} = 1$ — задает, что модуль R_i в первом варианте реализации выполняется на одном процессоре; $d_{i2} = 3$ — соответствует тому, что модуль R_i во втором варианте выполняется на трех процессорах; $d_{i3} = 5$ — указывает на то, что R_i-й модуль в третьем варианте выполняется на пяти процессорах. Необходимо таким образом распределить имеющиеся процессоры по программным модулям, чтобы выполнение всей задачи было закончено в кратчайшее время, т.е. следует уменьшить отрезок времени, начинающийся с момента начала выполнения работ (рисунок 5) и заканчивающийся в момент окончания выполнения последнего модуля.

Значение времени $t(d_{ij(z)})$ выполнения R_i-го модуля в $j(z)$ варианте распределения является известным. Пусть $T_i(z)$ — время выполнения i-го модуля в z-м варианте определяется следующим образом: $T_i(z) = t(d_{ij(z)})$.

Очевидно, что время окончания реализации всех про-

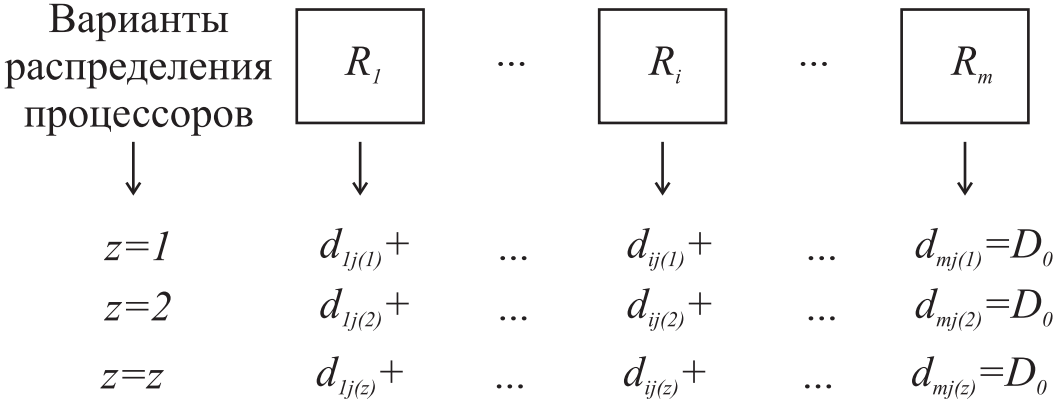

Рисунок 5 – Распределение процессоров

граммных модулей для z-го варианта

$$T(z) = \max_i T_i(z).$$

Тогда из всего множества вариантов необходимо выбрать такой вариант l, для которого

$$T(l) = \min_z T(z) = \min_z \max_i T_i(z) = \min_z \max_i t(d_{ij(z)}).$$

Уравнение Беллмана для минимаксной задачи при реализации алгоритма прямой прогонки имеет следующий вид:

$$W_i(S_j) = \min_{U_i} \max \left\{ w_i(S_j, U_i),\ W_{i-1}\left(\phi^{-1}(S_j, U_i)\right) \right\}$$

где $\phi^{-1}(S_j, U_i)$ — оператор, задающий номер состояния, в котором находилась система на предыдущем $(i-1)$-м этапе.

Для рассматриваемой задачи уравнение Беллмана представляется как:

$$T_i(D_{ik}) = \min_{d_{ij}} \max \left\{ t(d_{ij}),\ T_{i-1}(D_{ik} - d_{ij}) \right\}.$$

Интерпретация основных компонент уравнения Беллмана:

этапы — общее количество этапов m, где m — количество программных модулей. Текущее значение номера этапа i соответствует ситуации распределения имеющихся процессоров между первым, вторым, ..., i-м программными модулями. В частности: $i = 1$ — соответствует ситуации распределения всех процессоров для реализации только первого

программного модуля; $i = 2$ — предполагает распределение процессоров между R_1-м и R_2-м программными модулями; $i = 3$ — предполагает распределение процессоров между R_1-м, R_2-м и R_3-м программными модулями; $i = m$ — соответствует распределению процессоров между всеми программными модулями;

состояние — в качестве состояния выбирается текущее количество процессоров D_{ik}, распределяемых между программными модулями $\min\{d_{ij}\} \le D_{ik} \le D_0,\ k = \overline{1, N(i)}$, где $N(i)$ — максимальное число вариантов распределения прцессоров для i-го этапа;

управление — количество процессоров, которые выделяются на i-м этапе для реализации R_i-го программного модуля. Таким образом, минимум берется по d_{ij}, а максимум выбирается из значений $t(d_{ij})$ и $T_{i-1}(D_{ik} - d_{ij})$;

оператор перехода — количество процессоров, которые должны распределяться между первым, вторым,..., $(i-1)$-м программными модулями при условии, что текущее количество распределяемых процессоров равно D_{ik}, и принято управление, соответствующее выделению d_{ij} процессоров для реализации R_i-го программного модуля, представляется в виде выражения $D_{ik} - d_{ij}$;

локальная „полезность“ на i-м этапе — время выполнения i-го программного модуля при условии выделения для его реализации d_{ij} процессоров, равно $t(d_{ij})$;

условный оптимальный доход $T_i(D_{ik})$ — время окончания реализации первого, второго, ... i-го программных модулей при условии, что распределяется D_{ik} процессоров.

Решение уравнения производится в соответствии со следующим алгоритмом прямой прогонки.

Этап 1.

$i = 1$, текущее количество процессоров D_{1k} распределяется только для выполнения первого программного модуля

R_1:

$$T_1(D_{1k}) = \min_{d_{1j}} \max\left(t\left(d_{1j}\right),\, T_0(D_{1k} - d_{1j})\right) = t(d_{1j}),$$

так как выражение $T_0(D_{1k} - d_{1j})$ неопределено.

Таблица 1 – Результаты выполнения первого этапа

Состояние D_{1k}	Управление U_1	Оптимальный доход $T_1(D_{1k})$
$D_{11} = d_{11}$	$U_1 = d_{11}$	$T_1(D_{11}) = t(d_{11})$
$D_{12} = d_{12}$	$U_1 = d_{12}$	$T_1(D_{12}) = t(d_{12})$
$\ldots$	$\ldots$	$\ldots$
$D_{1j} = d_{1j}$	$U_1 = d_{1j}$	$T_1(D_{1j}) = t(d_{1j})$
$\ldots$	$\ldots$	$\ldots$
$D_{1N(1)} = d_{1N(1)}$	$U_1 = d_{1N(1)}$	$T_1(D_{1N(1)}) = t(d_{1N(1)})$

Результаты выполнения первого шага представляются в таблице 1.

Этап 2.

$i = 2$, текущее количество процессоров D_{2k} распределяется между двумя программными модулями R_1 и R_2. Уравнение Беллмана имеет следующий вид:

$$T_2(D_{2k}) = \min_{d_{2j}} \max\left(t\left(d_{2j}\right),\, T_1\left(D_{2k} - d_{2j}\right)\right),$$

Результаты выполнения второго шага представляются в таблице 2.

При составлении таблицы используются следующие правила и выполняются определенные процедуры:

количество строк равно количеству различных значений управлений U_2. В каждой строке указывается соответствующее значение: d_{21}, d_{22}, $\ldots$, $d_{2N(2)}$;

количество столбцов соответствует количеству состояний на предыдущем $(i-1)$-м шаге, т.е. в заголовке столбцов указываются значения: D_{11}, D_{12}, $\ldots$, $D_{1N(1)}$;

Таблица 2 – Результаты выполнения второго этапа

Управление U_2	Состояние первого этапа			
	D_{11}	D_{12}	...	$D_{1N(1)}$
$U_2 = d_{21}$	$T_2^*(D_{21})$	$T_2^*(D_{22})$	...	$T_2^*(D_{2N(1)})$
	$D_{21} = d_{21} + D_{11}$	$D_{22} = d_{21} + D_{12}$	...	$D_{2N(1)} = d_{21} + D_{1N(1)}$
$U_2 = d_{2i}$	$T_2^*(D_{2(t-1)N(1)+1})$	$T_2^*(D_{2(i-1)N(1)+2})$	...	$T_2^*(D_{2(i-N(1))})$
	$D_{2(i-1)N(1)+1} = d_{2i} + D_{11}$	$D_{2(i-1)N(1)+2} = d_{2i} + D_{12}$	...	$D_{2(i-N(1))} = d_{2i} + D_{1N(1)}$
$U_2 = d_{2N(2)}$	$T_2^*(D_{2(N(2)-1)N(1)+1})$	$T_2^*(D_{2(N(2)-1)N(1)+2})$	...	$T_2^*(D_{2(N(2)-N(1))})$
	$D_{2(N(2)-1)N(1)+1} = d_{2N(2)} + D_{11}$	$D_{2(N(2)-1)N(1)+2} = d_{2N(2)} + D_{12}$	...	$D_{2(N(2)-N(1))} = d_{2N(2)} + D_{1N(1)}$

каждая клетка таблицы разбивается на две части: в нижней части записывается сумма значений, указанных в заголовках соответствующих столбцов и строк. Эта сумма определяет состояние для второго этапа — количество процессоров, распределяемых на этапе. В верхней части каждой клетки, в которой указывается максимальное из следующих двух значений: первое значение — время выполнения второго модуля при реализации управления U_2; второе значение — время выполнения первого модуля на оставшихся процессорах.

Например, в нижней части клетки таблицы, соответствующей управлению $U_2 = d_{21}$ и состоянию предыдущего этапа D_{12} указывается состояние для второго этапа: $D_{22} = d_{21} + D_{12}$. В том случае, если значение D_{2k} превышает предельное количество процессоров D_0, то такое состояние является недопустимым. Недопустимые состояния вычеркиваются, и для них не определяются значения верхней части клетки. В верхней части клетки для состояния D_{22} записывается величина $T^*(D_{22}) = \max(t(d_{21}),\ T_1(D_{22} - d_{21}))$, т.е. определяется максимум из двух величин: времени реализации модуля R_2 на d_{21} процессорах и значения оптимального времени выполнения модуля R_1 на оставшихся процессорах $D_{22} - d_{21}$. Аналогичные действия выполняются для осталь-

ных верхних частей клеток;

Таблица 3 – Окончательная таблица для второго этапа

Состояние D_{2k}	Управление U_2	Условное оптимальное время $T_2(D_{2k})$
D_{21}	U_{21}	$T_2(D_{21})$
D_{22}	U_{22}	$T_2(D_{22})$
...	...	...
D_{2N}	U_{2N}	$T_2(D_{2N})$
...	...	...
$D_{2z(2)}$	$U_{2z(2)}$	$T_2(D_{2z(2)})$

составляется окончательная таблица для второго этапа принятия решений. Эта таблица содержит три столбца: в первом столбце указываются уникальные значения состояний для второго этапа (таблица 3). В промежуточной таблице имеются повторяющиеся значения состояний. Тогда необходимо для состояния D_{21} найти клетки, имеющие такие же значения состояний, и среди них выбрать клетку с минимальным значением $T_2^*(D_{21})$, т.е. $T_2(D_{21}) = \min\{T_2^*(D_{21})\}$. Оптимальное значение управления U_2 для состояния D_{21} указывается во втором столбце окончательной таблицы, а в третьем столбце записывается условное оптимальное время выполнения первых двух программных модулей R_1 и R_2 при наличии D_{21} процессоров — $T_2(D_{21})$. Аналогичные действия выполняются для всех остальных состояний D_{2k}, $k = \overline{1,\, z(2)}$ и критерия. Зафиксированные значения управления U_{2k} и $T_2(D_{2k})$ размещаются соответственно во втором и третьем столбцах.

Этап 3.

$i = 3$, распределение между R_1-м, R_2-м, R_3-м программными модулями текущего количества процессоров D_{3k}. Функциональное уравнение Беллмана для этого этапа имеет слующий вид:

$$T_3(D_{3k}) = \min_{d_{31}} \max(t(d_{3j}),\ T_2(D_{3k} - d_{3j})).$$

Результаты выполнения третьего шага представлены в таблице 4.

Таблица 4 – Результаты выполнения третьего этапа

Управление U_3	Состояние предыдущего второго этапа			
	D_{21}	D_{22}	...	$D_{2z(2)}$
$U_3 = d_{31}$	$T_3^*(D_{31})$	$T_3^*(D_{32})$	...	$T_3^*(D_{3z(2)})$
	$D_{31} = d_{31} + D_{21}$	$D_{32} = d_{31} + D_{22}$	...	$D_{3z(2)} = d_{31} + D_{2z(2)}$
$U_3 = d_{3i}$	$T_3^*(D_{3(i-1)z(2)+1})$	$T_3^*(D_{3(i-1)z(2)+2})$	...	$T_3^*(D_{3iz(2))})$
	$D_{3(i-1)z(2)+1} = d_{3i} + D_{21}$	$D_{3(i-1)z(2)+2} = d_{3i} + D_{22}$	...	$D_{3iz(2)} = d_{3i} + D_{2z(2)}$
$U_3 = d_{3N(3)}$	$T_3^*(D_{3(N(3)-1)z(2)+1})$	...	...	$T_3^*(D_{3(N(3)z(2))})$
	$D_{3(N(3)-1)z(2)+1} = d_{3N(3)} + D_{21}$	...	...	$D_{2(N(2)z(2))} = d_{3N(3)} + D_{2z(2)}$

При формировании элементов таблицы 4 используется следующее соотношение:

$$T_3^*(D_{3(i-1)z(2)+j}) = \max(t(d_{3i}),\ T_2(D_{3(i-1)z(2)+j} - d_{2i})).$$

Значения $T_2(D_{3(i-1)z(2)+j} - d_{2i})$ выбираются из третьего столбца окончательной таблицы для второго этапа. Аналогично этапу 2 на основании таблицы 4 конструируется окончательная таблица для третьего этапа, содержащая в первом столбце перечень уникальных состояний для третьего этапа, во втором столбце оптимальные управления для каждого состояния и в третьем столбце значения условного оптимального времени выполнения трех программных модулей: R_1, R_2, R_3. Выполнение алгоритма прямой прогонки прекращается при $i = m$. После этого в окончательной таблице m-го этапа для состояния, соответствующего предельному количеству процессоров D_0, находится оптимальное время $T_m(D_{mN(m)z(m)})$ выполнения всех программных модулей $R_1, \ldots, R_m$, а также

оптимальное количество процессоров, выделенных для реализации R_m-го модуля. На основании этого значения вычисляется оптимальное количество процессоров, которое назначено для реализации оставшихся модулей: R_1, R_2, . . . , R_{m-1}. Это число процессоров является входом в окончательную таблицу $(m-1)$-го этапа, что позволяется установить оптимальное количество процессоров для R_{m-1}-го модуля. Процесс последовательного обратного просмотра окончательных таблиц позволяет определить оптимальное число процессоров для R_m-го, R_{m-1}-го, R_{m-2}-го, . . . , R_1-го программных модулей.

4.2. Минимизация вероятности отказа технической системы методом динамического программирования

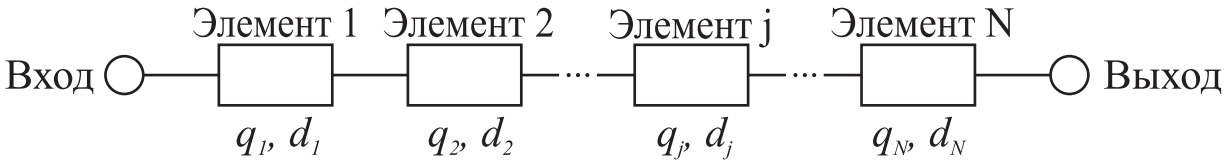

Рисунок 6 – Структура исходной системы

Рассматривается система, состоящая из N последовательно соединенных элементов (рисунок 6). Каждый из элементов характеризуется вероятностью отказа q_j и весом d_j, $j = \overline{1, N}$.

Для повышения надежности системы используется резервирование, которое состоит в параллельном подключении к j-му элементу дополнительных резервных элементов. В результате подключения резервных элементов структура системы может быть представлена следующим образом (рисунок 7). Необходимо найти оптимальное количество элементов в каждом из фрагментов, т.е. такое количество элементов, при котором вероятность отказа системы будет минимальной, и будут выполнены ограничения на суммарный вес элементов системы — D_0.

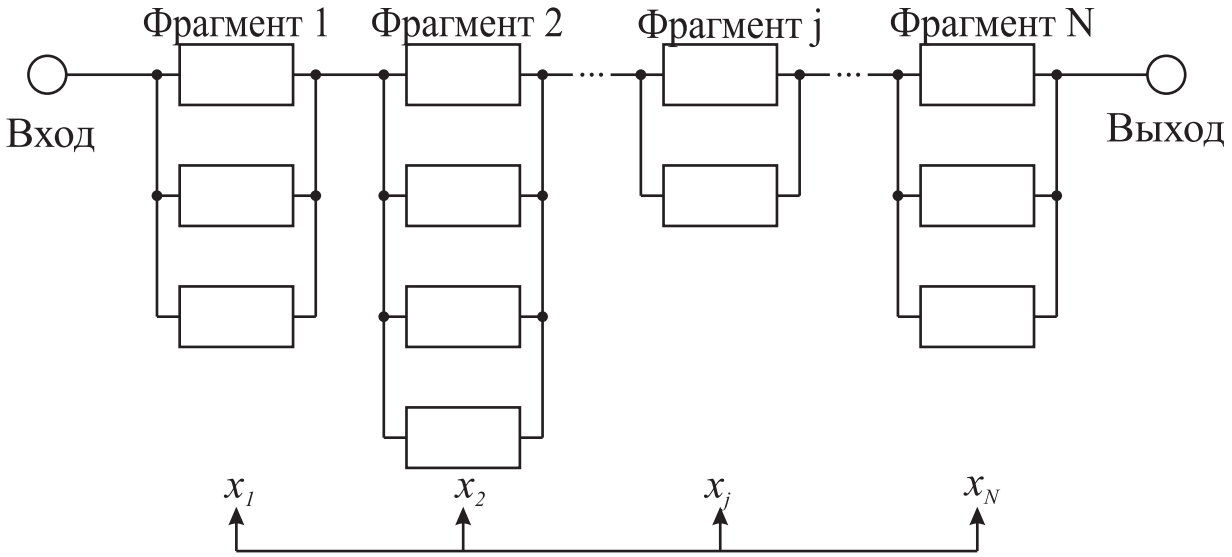

Рисунок 7 – Структура резервированной системы

Для конструирования критерия и ограничений необходимо определить следующие компоненты:

$q_j^{x_j}$ — вероятность того, что все x_j элементов, входящих в j-й фрагмент, откажут;

$(1-q_j^{x_j})$ — вероятность того, что в j-м фрагменте работоспособен хотя бы один элемент и, следовательно, работоспособен j-й фрагмент системы в целом.

Тогда вероятность того, что будет работоспособной вся система, равна произведению вероятностей работоспособности каждого из фрагментов:

$$P_P = \prod_{j=1}^{N} \left(1 - q_j^{x_j}\right).$$

Очевидно, что вероятность отказа системы вычисляется следующим образом:

$$P_O = 1 - \prod_{j=1}^{N} \left(1 - q_j^{x_j}\right).$$

Тогда критерий имеет следующий вид:

$$P_O = 1 - \prod_{j=1}^{N} \left(1 - q_j^{x_j}\right) \to \min,$$

и ограничение на суммарный вес элементов, входящих в систему, представляется следующим образом:

$$\sum_{j=1}^{N} d_j x_j \le D_0.$$

Для решения задачи динамического программирования вводится функция $f_i(D_{ik})$, где D_{ik} — текущий вес элементов системы:

$$f_i(D_{ik}) = \min_{x_j} \left(1 - \prod_{j=1}^{i} \left(1 - q_j^{x_j}\right)\right),$$

где $f_i(D_{ik})$ задает оптимальную вероятность отказа для i фрагментов системы при условии, что текущий вес D_{ik} распределяется между 1, 2, ..., i-м фрагментами системы. По аналогии

$$f_{i-1}(D_{(i-1)k}) = \min_{x_j} \left(1 - \prod_{j=1}^{i-1} \left(1 - q_j^{x_j}\right)\right).$$

Тогда

$$f_i(D_{ik}) = \min_{x_j} \left(1 - \prod_{j=1}^{i-1} \left(1 - q_j^{x_j}\right)\left(1 - q_i^{x_i}\right)\right).$$

Пусть $\prod_{j=1}^{i-1} \left(1 - q_j^{x_j}\right) = A$, тогда

$$f_i(D_{ik}) = \min_{x_j} \left(1 - A \cdot \left(1 - q_i^{x_i}\right)\right) = \min_{x_j} \left(1 - A + Aq_i^{x_i} - q_i^{x_i} + q_i^{x_i}\right) =$$

$$= \min_{x_j} \left(1 - A - q_i^{x_i}(1 - A) + q_i^{x_i}\right) = \min_{x_j} \left(f_{i-1} - q_i^{x_i} f_{i-1} + q_i^{x_i}\right).$$

Окончательно уравнение Беллмана имеет следующий вид:

$$f_i(D_{ik}) = \min_{x_i} \left(q_i^{x_i} - q_i^{x_i} f_{i-1}(D - d_i x_i) + f_{i-1}(D_{ik} - d_i x_i)\right),\ i = \overline{1,\ N}.$$

Интерпретация уравнения Беллмана:

этапы — общее количество этапов равно количеству сегментов системы N. Текущий номер этапа i соответствует нахождению оптимального распределения элементов между 1, 2, . . . , i-м фрагментами;

состояние — это текущий вес D_{ik}, который распределяется между фрагментами системы;

управление — это количество элементов x_i, которое включается в i-й фрагмент системы;

оператор перехода — значение веса, которое может быть распределено между 1, 2, . . . , $(i-1)$-м фрагментами системы при условии, что между первыми i фрагментами распределяется суммарный вес D_{ik} и i-й фрагмент назначено x_i элементов.

Этап 1.

$i = 1$. Текущий вес D_{1k} полностью выделяется для элементов первого фрагмента. Уравнение Беллмана представляется следующим образом:

$$f_1(D_{1k}) = \min_{x_1}(q_1^{x_1}) = q_1^{x_1}$$

Результаты выполнения первого этапа принятия решения представлены в таблице 5.

Максимальное значение x_1 ограничивается допустимым весом, т.е. $d_1 \cdot m(1) \le D_0$.

Этап 2.

$i = 2$, текущий вес D_{2k} распределяется между первым и вторым фрагментами. Уравнение Беллмана имеет следующий вид:

$$f_2(D_{2k}) = \min_{x_2}(q_2^{x_2} - q_2^{x_2} f_1(D_{2k} - d_2 x_2) + f_1(D_{2k} - d_2 x_2)).$$

Результаты выполнения второго этапа представляются в таблице 6

Таблица 5 – Результаты выполнения первого этапа

Состояние D_{1k}	Управление x_1	Условный оптимальный доход $f_1(D_{1k})$
$D_{11} = d_1 \cdot 1$	$x_1 = 1$	$f_1(D_{11}) = q_1^1$
$D_{12} = d_1 \cdot 2$	$x_1 = 2$	$f_1(D_{12}) = q_1^2$
$D_{13} = d_1 \cdot 3$	$x_1 = 3$	$f_1(D_{13}) = q_1^3$
$\ldots$	$\ldots$	$\ldots$
$D_{1m(1)} = d_1 \cdot m(1) \le D_0$	$x_1 = m(1)$	$f_1(D_{1m(1)}) = q_1^{x_1 = m(1)}$

Таблица 6 – Результаты выполнения второго этапа

Управление x_2	Состояние первого этапа			
	D_{11}	D_{12}	$\ldots$	$D_{1m(1)}$
$x_2 = 1$	$f_2^*(D_{21})$	$f_2^*(D_{22})$	$\ldots$	$f_2^*(D_{2m(1)})$
	$D_{21} = d_2 \cdot 1 + D_{11}$	$D_{22} = d_2 \cdot 1 + D_{12}$	$\ldots$	$D_{2m(1)} = d_2 \cdot 1 + D_{1m(1)}$
$x_2 = j$	$f_2^*(D_{2(i-1)m(1)+1})$	$f_2^*(D_{2(i-1)m(1)+2})$	$\ldots$	$f_2^*(D_{2im(1)})$
	$D_{2(i-1)m(1)+1} = d_2 \cdot i + D_{11}$	$D_{2(i-1)m(1)+2} = d_2 \cdot i + D_{12}$	$\ldots$	$D_{2im(1)} = d_2 \cdot i + D_{1m(1)}$
$x_2 = m(2)$	$f_2^*(D_{2(m(2)-1)m(1)+1})$	$\ldots$	$\ldots$	$f_2^*(D_{2m(2)m(1)})$
	$D_{2(m(2)-1)m(1)+1}$	$\ldots$	$\ldots$	$D_{2m(2)m(1)}$

В таблице 6 количество столбцов ровно числу состояний первого этапа принятия решения, а число строк — максимально допустимому $m(2)$ количеству элементов, включаемых во второй сегмент системы. Каждая клетка таблицы делится на две части: в нижней части указывается состояние второго этапа, определяемое управлением x_2 и соответствующим состоянием первого этапа. В частности, если управление $x_2 = 1$ и состояние D_{1j}, то соответствующее состояние второго этапа вычисляется как:

$$D_{2(i-1)\cdot m(1)+j} = d_2 \cdot i + D_{1j}.$$

Состояния второго этапа являются недопустимыми, если их

значения превышают предельный вес D_0.

В верхней части клеток для каждого из допустимых состояний записывается значение вероятности отказа $f_2^*(D_{2(i-1)m(1)+j})$, вычисляемое следующим образом:

$$f_2^*(D_{2(i-1)m(1)+j}) = (q_2^{x_2} - q_2^{x_2} f_1(D_{2(i-1)m(1)+j} - d_2 x_2) + \\ + f_1(d_{2(i-1)m(1)+j} - d_2 x_2)).$$

На основании данных таблицы 6 формируется окончательная таблица для второго этапа, которая включает три столбца. В первом столбце перечисляются уникальные значения состояний второго этапа. Необходимо отметить, что в промежуточной таблице второго этапа содержатся повторяющиеся значения. В частности, для каждого состояния D_{2k} необходимо найти клетки, имеющие такие же значения состояний, и среди них выбрать клетку с минимальным значением $f_2(D_{2k})$, т.е. $f_2(D_{2k}) = \min\{f_2^*(D_{2k})\}$. Тогда оптимальное количество элементов x_2 для рассматриваемого состояния D_{2k}, соответствующее клетке с минимальным значением $f_2^*(D_{2k})$, записывается во втором столбце, а соответственно минимальное значение $f_2(D_{2k})$ — в третьем столбце.

Реализация алгоритма прямой прогонки прекращается при $i = N$, а затем выполняется обратный просмотр окончательных таблиц для определения оптимального количества элементов в каждом из сегментов.

5. Пример решения задачи методом динамического программирования

Задача.

Решить минимаксную задачу распределения программных модулей между процессорами для следующих исходных данных: количество модулей $m = 4$, общее количество процессоров $D_0 = 10$, матрица времени выполнения программных модулей в зависимости от количества выделяемых про-

цессоров T:

$$T = \begin{Vmatrix} 12 & 14 & 0 & 0 \\ 7 & 10 & 9 & 0 \\ 2 & 7 & 5 & 9 \\ 0 & 4 & 2 & 6 \\ 0 & 2 & 0 & 4 \\ 0 & 0 & 0 & 1 \end{Vmatrix}$$

Решение.

Этап 1.

$i = 1$, производится распределение процессоров для выполнения первого программного модуля. Уравнение Беллмана выглядит следующим образом:

$$T_1(D_{1j}) = \min_{d_{1j}}(t(d_{1j})) = t(d_{1j}).$$

Результаты выполнения этапа 1 представлены в таблице 7. Здесь приведены данные для возможных вариантов распределения процессоров для выполнения первого модуля, взятые из первого столбца исходной матрицы T. Данные таблицы 7 означают, что возможно выполнение первого модуля на одном, двух и трех процессорах с временными затратами в 12, 7 и 2 единицы соответственно.

Таблица 7 – Выполнение этапа 1

Состояние D_{1j}	Управление d_{1j}	Условное оптимальное время $T_1(D_{1j})$
1	1	12
2	2	7
3	3	2

Этап 2.

$i = 2$, производится распределение процессоров между первым и вторым программными модулями. Уравнение Беллмана имеет следующий вид:

$$T_2(D_{2j}) = \min_{d_{2j}} \max \{t(d_{2j}),\ T_1(D_{2j} - d_{2j})\}.$$

Результаты выполнения этапа 2 представлены в промежуточной и окончательной таблицах 8 и 9.

Таблица 8 – Промежуточная таблица для этапа 2

Управление d_{2j}	Состояния первого этапа D_{1j}		
	1	2	3
1	14	14	14
	2	3	4
2	12	10	10
	3	4	5
3	12	7	7
	4	5	6
4	12	7	4
	5	6	7
5	12	7	2
	6	7	8

Таблица 9 – Окончательная таблица для этапа 2

Состояние D_{2j}	Управление d_{2j}	Условное оптимальное время $T_2(D_{2j})$
2	1	14
3	2	12
4	1	10
5	3	7
6	3	7
7	4	4
8	5	2

Этап 3.

$i = 3$, производится распределение процессоров между первым, вторым и третьим программныии модулями. Уравнение Беллмана имеет вид:

$$T_3(D_{3j}) \min_{d_{3j}} \max \{t(d_{3j}),\ T_2(D_{3j} - d_{3j})\}.$$

Результаты выполнения этапа 3 представлены в промежуточной таблице 10 и окончательной таблице 11. Недопустимые состояния выделены $*$.

Таблица 10 – Промежуточная таблица для этапа 3

Управление d_{3j}	Состояния второго этапа D_{2j}						
	2	3	4	5	6	7	8
2	14	12	10	9	9	9	9
	4	5	6	7	8	9	10
3	14	12	10	7	7	5	$*$
	5	6	7	8	9	10	
4	14	12	10	7	7	$*$	$*$
	6	7	8	9	10		

Таблица 11 – Окончательная таблица для этапа 3

Состояние D_{3j}	Управление d_{3j}	Условное оптимальное время $T_3(D_{3j})$
4	2	14
5	2	12
6	2	10
7	2	9
8	3	7
9	3	7
10	3	5

Этап 4.

Производится распределение процессоров между всеми программныии модулями. Результаты выполнения этапа 3 представлены в промежуточной таблице 12 и окончательной таблице 13. Недопустимые состояния выделены $*$.

Таблица 12 – Промежуточная таблица для этапа 4

Управление d_{4j}	Состояния третьего этапа D_{3j}						
	4	5	6	7	8	9	10
3	14	12	10	9	$*$	$*$	$*$
	7	8	9	10			
4	14	12	10	$*$	$*$	$*$	$*$
	8	9	10				
5	14	12	$*$	$*$	$*$	$*$	$*$
	9	10					
6	14	$*$	$*$	$*$	$*$	$*$	$*$
	10						

Таблица 13 – Окончательная таблица для этапа 4

Состояние D_{4j}	Управление d_{4j}	Условное оптимальное время $T_4(D_{4j})$
7	3	14
8	3	12
9	3	10
10	3	9

Этап 5.

Задача решается в обратном порядке: $d_1 = 2$, $d_2 = 3$, $d_3 = 2$, $d_4 = 3$.

Время выполнения всего комплекса программ равно $T = 9$.

6. Контрольные вопросы

1. Принцип оптимальности Беллмана.
2. Каковы компоненты многоэтапных задач принятия решения?
3. Поясните принципы разметки детерминисткого графа для модели задачи с аддитивным критерием.
4. Алгоритм решения функционального уравнения Беллмана.
5. Какие задачи могут быть решены методом динамического программирования.
6. В чем состоит принцип отсутствия последействия?
7. Что такое условное оптимальное управление?
8. Что устанавливает оператор перехода?

7. Литература

1. Черноруцкий, И.Г. Методы принятия решений [Текст]: учебник / И.Г. Черноруцкий. — СПб: БХВ-Петербург, 2005. — 416 с.
2. Петровский, А.Б. Теория принятия решений [Текст]: учебник для студ. высш. учеб. заведений / А.Б. Петровский. — М.: Издательский центр „Академия“, 2009. — 400 с. — (Университетский учебник. Сер. „Прикладная математика и информатика“).