МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Институт Математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

02.03.03 Математическое обеспечение и администрирование информационных систем
(код и наименование направления подготовки / специальности)

WEB - дизайн и мультимедиа
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
(БАКАЛАВРСКАЯ РАБОТА)

на тему «Разработка программного обеспечения для решения алгоритмических задач методом динамического программирования»

Обучающийся С.А. Булгин
(Инициалы Фамилия) (личная подпись)

Руководитель к.т.н., доцент, Н.А. Сосина
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант к.ф.н., доцент, М.М. Бажутина
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2022

Аннотация

Бакалаврская работа выполнена на тему «Разработка программного обеспечения для решения алгоритмических задач методом динамического программирования».

Введение содержит подробное описание актуальности выбранной темы, объект и предмет исследования, цель работы и задачи, необходимые для достижения цели.

В первой главе была описана сущность динамического программирования, была поставлена задача динамического программирования. Рассмотрели два основных подхода к решению задачи.

Во второй главе была рассмотрена математическая модель, где было выявлено, что при использовании динамического программирования многошаговая задача решается двумя способами: от конца к началу и от начала к концу.

В третьей главе прописали требования к проектируемой программе. Было сделано проектирование для решения задачи, где была представлена блок-схема и описаны входные данные. С помощью метода функционального тестирования мы выбрали несколько пунктов, по которым и протестировали программу.

В заключение представлены результаты выполнения выпускной квалификационной работы.

Бакалаврская работа состоит из введения, трёх глав, заключения и списка использованной литературы.

Бакалаврская работа состоит из 40 страниц текста, 19 рисунка, 1 таблица, 2 листинга и 25 источников.

Abstract

The topic of the present graduation work is *Software development for solving algorithmic problems by dynamic programming method*.

The research is devoted development of algorithms of the dynamic programming method for solving algorithmic problems, as well as for implementing the software of the obtained algorithms.

The introduction contains a detailed description of the relevance of the chosen topic, the object and subject of research, the purpose of the work and the tasks necessary to achieve the goal.

The first chapter the essence of dynamic programming was described, the task of dynamic programming was set. We considered two main approaches to solving the task.

The second chapter a mathematical model was considered, where it was revealed that when using dynamic programming, a multistep problem is solved in two ways: from end to beginning and from beginning to end.

The third chapter the requirements for the projected program were prescribed. A design was made to solve the problem, where a block diagram was presented and input data was described. Using the functional testing method, we selected several points, according to which we tested the program.

In conclusion, the conclusions of the entire work are drawn.

The bachelor's thesis consists of an introduction, three chapters, a conclusion and list of used literature.

The volume of the bachelor's thesis is 40 pages, it also contains 19 figures, 1 table, 2 listings and a list of 25 references.

Оглавление

Введение

Актуальность исследования темы обусловлена тем, что в настоящее время задачи принятия рациональных решений, оптимального управления и выбора наилучших вариантов решаются на предприятиях разных профилей и направлений работы. Решение алгоритмических задач с использованием методов динамического программирования является одним из наиболее эффективных методов оптимизации и занимает особое положение. Высокая эффективность и привлекательность этого метода достигается благодаря простоте основного принципа динамического программирования – принципа оптимальности.

Объект исследования – алгоритмические задачи, решаемые методом динамического программирования.

Предмет исследования – алгоритм метода динамического программирования.

Целью работы является использование метода динамического программирования для решения алгоритмических задач.

Задачи работы:

- исследовать основной принцип метода динамического программирования;
- определить перечень алгоритмических задач, решаемых с помощью метода динамического программирования;
- продемонстрировать решение алгоритмических задач с применением метода динамического программирования;
- реализовать программное обеспечение полученных алгоритмов.

Глава 1 Сущность метода динамического программирования

1.1 Метод динамического программирования

Динамическое программирование относится к математическому программированию, в котором процесс принятия решения и управления может быть разбит на отдельные шаги, такой процесс называется многошаговым.

В динамическом программировании не существует универсального метода решения задач. Одним из основных методов решения задач динамического программирования является метод рекурентных соотношений, который основан на принципе оптимальности который был сформулирован американским математиком Р.Беллманом (Ричард Эрнест Беллман) в 1953 году. «Оптимальная политика обладает, тем свойством, что каковы бы ни были начальное состояние и первоначально принятое решение, последующие решения должны составлять оптимальную политику относительно состояния, получившегося в результате первоначально принятого решения» На сегодняшний день этот метод применяется в большом количестве научных областей: от экономики до криптографии [16].

«Динамическое программирование – это подход к решению сложных задач, суть которого заключается в разбиении исходной задачи и последующем решении нескольких упрощенных подзадач, после чего объединить результаты подзадач в общее решение. Такой метод решения применяется к задачам, структура которых позволяет разбить их на задачу меньшей сложности, чем исходная задача» [9].

Как правило, многие подзадачи одинаковы и каждую подзадачу необходимо решить один раз, тогда как при использовании других подходов одна и та же подзадача может быть решена несколько раз. Применение этого метода позволяет значительно сократить время решения задачи за счет уменьшения количества необходимых вычислений, гарантируя при этом

оптимальное решение задачи. Применение метода динамического программирования особенно эффективно, когда повторяющихся подзадач экспоненциально много.

Стоит отметить, что в некоторых случаях алгоритмы перебора могут быть эффективней по времени, чем алгоритмы динамического программирования, но оптимального решения задачи они не обеспечивают.

Общую постановку задачи динамического программирования можно сформулировать следующим образом: рассматривается некоторый управляемый процесс, в результате управления система (объект управления), обозначим буквой S, переводится из начального состояния S_0 в состояние S_n. Предполагается, что управление можно разбить на n шагов. То есть решение принимается последовательно на каждом шаге, а управление, переводящее систему из начального состояния в конечное, представляет собой совокупность n пошаговых управлений. Таким образом, для решения задачи необходимо определить такое допустимое управление, переводящую систему в конечное состояние, при котором обеспечивается экстремум целевой функции.

В динамическом программировании существует два основных подхода к решению задач:

- динамическое программирование сверху заключается в рекурсивном разбиении исходной задачи на подзадачи меньшей размерности, пока не будет достигнута задача, решение которой является тривиальным;
- динамическое программирование снизу заключается в решении всех подзадач результат решения, которых необходим для решения исходной задачи. На этом строится решение исходной задачи.

Динамическое программирование сверху требует меньшего количества памяти стека и меньшее количество вызовов функции.

Сформулируем основные этапы решения задачи методом динамического программирования:

- разбить исходную задачу на подзадачи меньшего размера;

- составить рекуррентную формулу для нахождения оптимального решения;
- рассчитать значения, которые соответствуют оптимальному решению подзадач;
- использовать результаты решения подзадач в решении исходной задачи.

«Условиями применимости метода динамического программирования являются оптимальная подструктура решаемой задачи и аддитивность целевой функции задачи. Условие оптимальной подструктуры означает, что любая подзадача, полученная после разбиения исходной задачи, в свою очередь, так же делятся на подзадачи меньшей размерности, до тех пор, пока решение не станет элементарным» [14].

Условие аддитивности заключается в том, что исходная задача состоит из перекрывающихся подзадач, что подразумевает использование решения подзадач в решении задач большего размера.

Если возможно рекурсивно вложить подзадачи в более крупные, так, что к ним можно применить метод динамического программирования, то между значением исходной задачи и подзадачами существует связь. Это соотношение называют динамическим уравнением Беллмана [13].

1.2 Динамическое уравнение Беллмана

Пусть имеется управляемая система.

S – ее текущее состояние.

W_i=$f_i(S,x_i)$ – функция выигрыша (стоимости) при использовании управления X на i–ом шаге.

$S' = \phi_i(S,x_i)$ – состояние, в которые переходят системы при воздействия X.

Независимо от значения S нужно выбрать управление на этом шаге так, чтобы выигрыш на данном шаге плюс оптимальный выигрыш на всех последующих шагах был максимальным.

Тогда уравнение Беллмана имеет вид:

$$\left(f_i(\mathrm{s},\mathrm{x}_i)+\mathrm{W}_{\mathrm{i}+1}\left(\varphi_i(\mathrm{S},\mathrm{x}_i)\right)\right)=\mathrm{W}_{\mathrm{i}}(S). \qquad (1)$$

Впервые уравнение Беллмана использовалось в теории управления техническими системами и в других темах прикладной математики, а затем стало популярным и важным элементом экономики.

С помощью уравнения Беллмана решается ряд проблем, которые решаются с помощью использования теории оптимального уравнения. «Уравнение Беллмана относится к уравнению динамического программирования и связано с задачами оптимизации в дискретном времени. Уравнением для задач непрерывной оптимизации является уравнение в частных производных, которое называют уравнением Гамильтона-Якоби-Беллмана» [15].

Для того чтобы наиболее углубить знания в уравнении Беллмана, необходимо разобрать несколько основных понятий:

- любая оптимизационная задача имеет конечную цель: минимизация или максимизация того или иного показателя исследуемой системы (минимизация времени в пути; максимизация прибыли и т.д.). Математическая функция описываемая поставленную задачу называется целевой функцией;
- применение метода динамического программирования позволяет разбить сложную задачу многопериодного планирования на простые этапы в разных моментах времени. Из этого следует, что появляется необходимость контролировать как будет меняться со временем ситуация с решениями. «Состоянием» называется информация о

ситуации на данном этапе, которая необходима для принятия рационального решения;

- переменные, используемые в момент времени называют контрольными. Основываясь на значениях этих переменных, принимается решение о дальнейшем управлении. Выбор управляющих переменных может быть эквивалентным выбору следующего состояния, но в общем случае на следующее состояние кроме текущего контроля влияют другие факторы;
- подход к решению задач методом динамического программирования описывает оптимальный план, находя правило, которое сообщает, какие элементы управления необходимо применить с учетом возможных значений состояния.

Таким образом, оптимальным правилом принятия решения является то, которое достигает наилучшего из возможных значения цели [24].

Благодаря исследованию Ричарда Беллмана задача динамической оптимизации в дискретном времени была представлена в рекурсивной форме, известной как обратная индукция, записав взаимосвязь между функцией значения в одном периоде и функцией значения в следующем периоде. Соотношение между этими функциями и называется «уравнением Беллмана». При таком подходе оптимальное управление в последний период времени заранее рассматривается как функция значения переменной состояния в то время, и итоговое оптимальное значение переменной состояния [25].

«Разберем особенности математической модели метода динамического программирования:

- исходная задача представляется как поэтапный многошаговый процесс управления;
- целевая функция является аддитивной и равна сумме целевых функций каждого шага оптимизации;

- предыдущие шаги управления не влияют на выбор управления на каждом следующем шаге. Выбор управления на следующем шаге зависит только от состояния системы на этом шаге;
- после каждого шага управления состояние системы S_k зависит исключительно от предыдущего состояния системы и управляющего воздействия X_k;
- управление X_k на каждом шаге зависит от конечного числа управляющих переменных, а состояние системы S_k – от конечного числа параметров;
- вектор X представляет собой оптимальное управление и является последовательностью оптимальных управлений на каждом шаге, число которых определяет количество шагов» [15].

«Перечислим свойства задач, допускающих применение метода динамического программирования:

- необходимо иметь возможность интерпретации задачи как многошагового процесса принятия решений;
- задача должна быть определена для любого числа шагов и иметь структуру которая не зависит от их числа;
- применение управления на каком либо шаге не должно оказывать влияния на предыдущие решения» [3].

Выводы по главе

В первой главе выпускной квалификационной работы описывается сущность динамического программирования.

Приводится уравнение Беллмана, как основная концепция для решения алгоритмических задач методом динамического программирования.

Рассматриваются особенности и свойства математической модели, для которой применим метод динамического программирования.

Глава 2 Постановка алгоритмической задачи

2.1 Математическая модель задачи динамического программирования

«Математическая модель – это математическое представление реальности [4].

Математическое моделирование – это процесс построения и изучения математических моделей.

Все естественные и общественные науки, использующие математический аппарат, по сути, занимаются математическим моделированием: заменяют реальный объект его математической моделью.

Одним из методов математического моделирования является метод динамического программирования» [15].

«Под динамическим программированием понимают некоторый специальный метод оптимизации, суть которого состоит в отыскании оптимального решения путем выполнения вычислений в несколько шагов (этапов). Вся задача оптимизации разделяется на несколько шагов, причем все шаги могут быть уникальными или одинаковыми и чередоваться друг с другом.

При использовании динамического программирования многошаговая задача решается двумя этапами: от конца к началу и от начала к концу. Первый этап длительный и трудоемкий, второй - короткий и уточняет решение первого» [12].

2.2 Метод решения задач

Решение задач – процесс, являющийся составной частью мышления, выполнение действий или мыслительных операций, направленное на достижение цели, заданной в рамках проблемной ситуации.

«Основное функциональное уравнение динамического программирования:

$$F_i(x_i; U_i) = extr_{U_i}\big(Z_i(x_{i-1}; U_i) + F_{i+1}(x_i)\big), i = 1,\acute{}N, \qquad (2)$$

где:

- x_{i-1} – множество состояний, в которых система находится перед i–м шагом;
- x_i – множество состояний системы в конце i-го шага;
- U_i – множество управлений на i-ом шаге, под воздействием которых система переходит в одно из состояний множества x_i;
- $F_i(x_{i-1}; U_i)$ – условно-оптимальное значение целевой функции на интервале от i–го до n–го шага включительно;
- $Z_i(x_{i-1}; U_i)$ – значение целевой функции на i-ом шаге для всех управлений из множества U_i;
- $F_{i+1}(x_i)$ – условно-оптимальное значение целевой функции на интервале от (i+1)–го шага до N-го включительно.

На последнем N шаге справедлива следующая формула:

$$F_N(x_{N-1}; U_N) = extr_{U_N} Z_N(x_{N-1}; U_N) \qquad (3)$$» [8].

Рассмотрим решение сетевой задачи методом динамического программирования.

Сеть дорог с двухсторонним движением задана матрицей расстоянийв км (матрицей весовых коэффициентов). Необходимо найти для данной сети дорог самый короткий маршрут из пункта 1 в пункт 10. Решим задачу методом динамического программирования [2],[7].

Схематически сетевая модель задачи изображена на рисунке 1 в виде неориентированной сети.

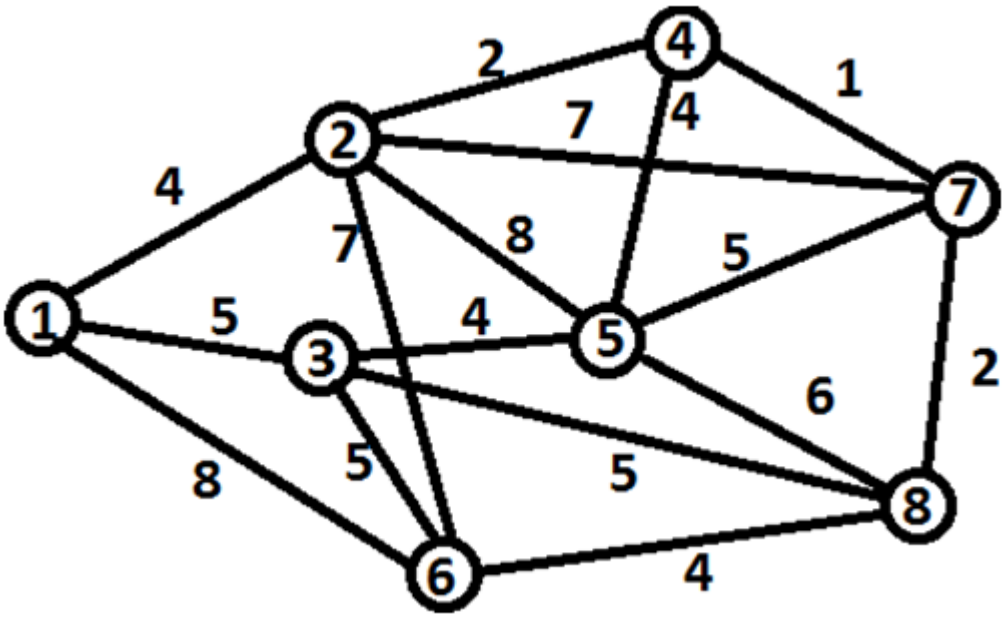

Рисунок 1 – Сетевая модель задачи

Продемонстрируем работу алгоритма Дейкстры, отмечая временные и постоянные пометки непосредственно в узлах сети. Для этого каждый узел изобразим в виде овала и разделим его на три части следующимобразом.

На рисунке 2 изображен алгоритм Дейкстры.

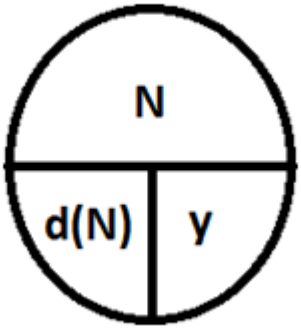

Рисунок 2 – Алгоритм Дейкстры

Где N-номер узла, d(N)-временная пометка узла N, если она становится постоянной, то ее подчеркиваем. Т.е. получим d(N)y – номер узла, придя из которого получена пометка d(N).

На первом шаге решения мы к исходному узлу приписываем постоянную пометку, равную нулю, т.е d(1)=0 и y:=1. Всем остальным узлам, приписываем временные пометки, отметим это на сети (рисунок 3).

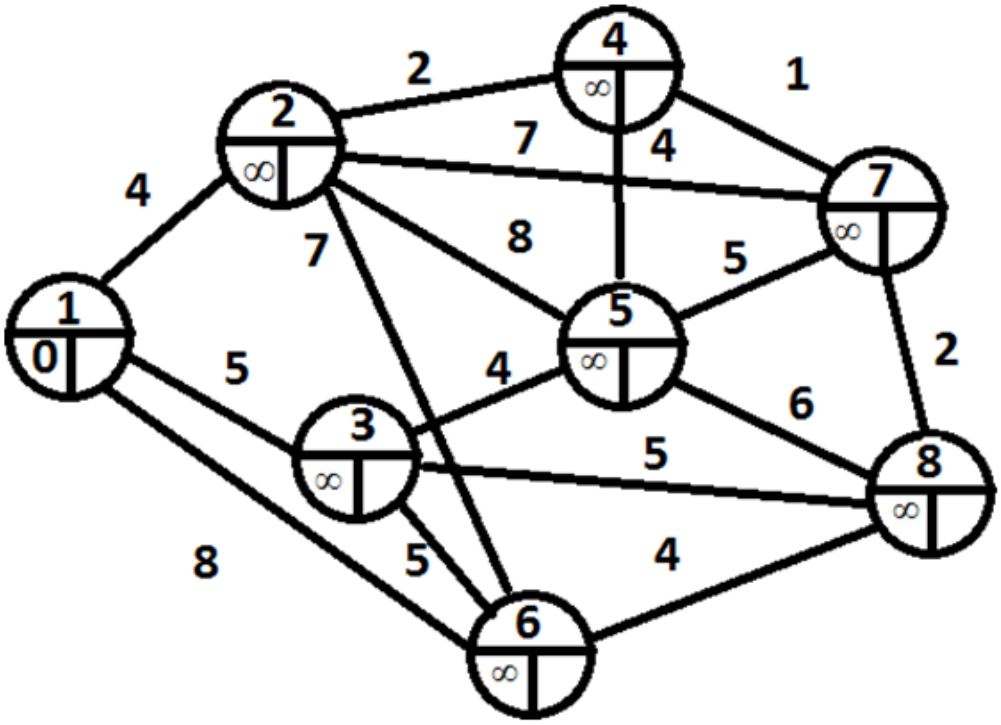

Рисунок 3 – Первый шаг решения задачи

«На первом шаге узлы 2,3 и 6смежные с последним, получившим постоянную пометку узлом 1. Новые значения временных пометок этих узлов будут» [15]:

$$d(2) := min\{d(2), d(1) + c_{12}\} = min\{\infty, 0 + 3\} = 3;$$
$$d(3) := min\{d(3), d(1) + c_{13}\} = min\{\infty, 0 + 4\} = 4;$$
$$d(6) := min\{d(6), d(1) + c_{16}\} = min\{\infty, 0 + 9\} = 9.$$

«Минимальной временной пометкой является пометка узла 2. Поэтому она становится постоянной. Подчеркиваем ее и y:=2. Помечаем дугу (1,2). Отметим это на сети (рисунок 4)» [15].

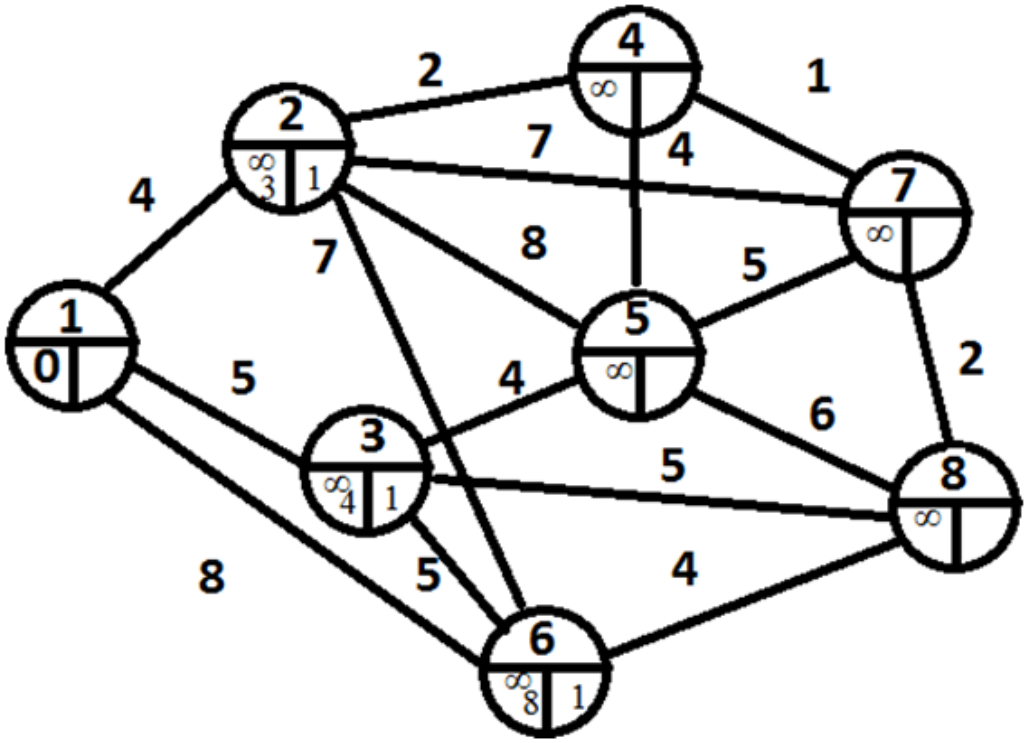

Рисунок 4 – Второй шаг решения задачи

«На втором шаге поскольку узел 8 не помечен постоянной пометкой, переходим к шагу 1. Узлы 4, 5, 6 и 7 смежные с последним, получившим постоянную пометку узлом 2. Новые значения временных пометок этих узлов будут» [15]:

$$d(4) := min\{d(4), d(2) + c_{24}\} = min\{\infty, 3 + 2\} = 5;$$
$$d(5) := min\{d(5), d(2) + c_{25}\} = min\{\infty, 3 + 9\} = 12;$$
$$d(6) := min\{d(6), d(2) + c_{26}\} = min\{9,3 + 8\} = 9;$$
$$d(7) := min\{d(7), d(2) + c_{27}\} = min\{\infty, 3 + 8\} = 11.$$

«Минимальной временной пометкой является пометка узла 3. Поэтому она становится постоянной. Подчеркиваем ее и y:=3. Помечаем дугу (1,3). Отметим это на сети (рисунок 5)» [15]:

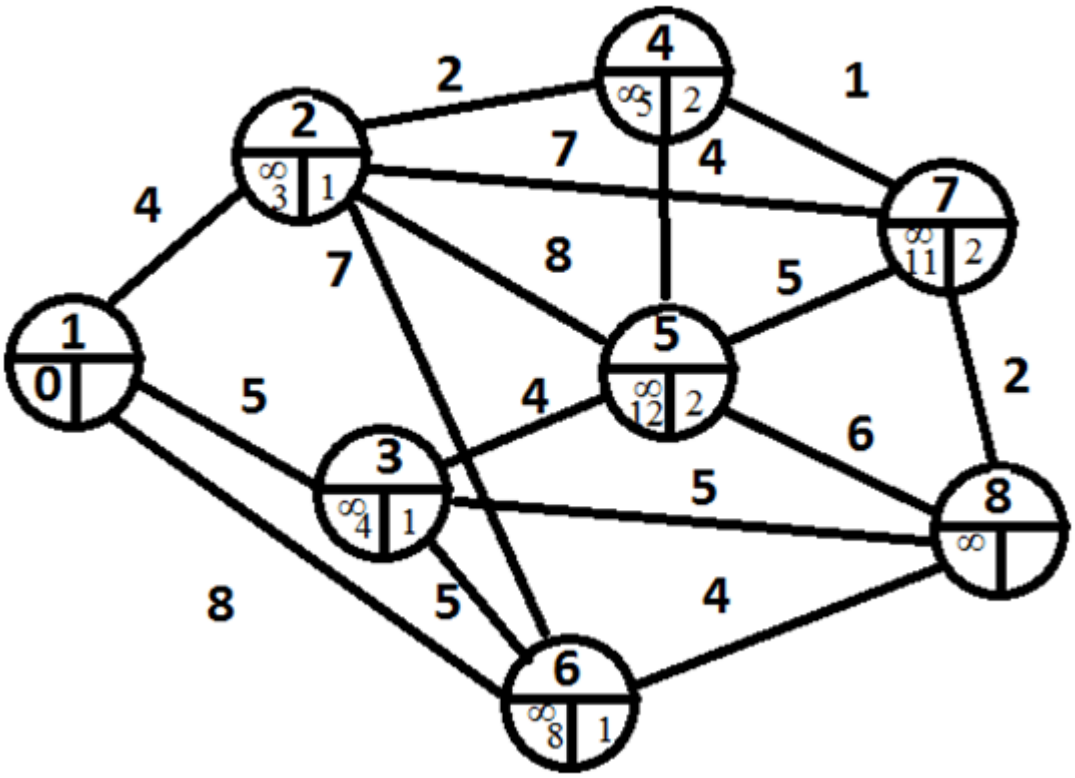

Рисунок 5 – Третий шаг решения задачи

«На третьем шаге поскольку узел 8 не помечен постоянной пометкой, переходим к шагу 2. Узлы 5, 6 и 8 смежные с последним, получившим постоянную пометку узлом 3. Новые значения временных пометок этих узлов будут» [15]:

$$d(5) := min\{d(5), d(3) + c_{35}\} = min\{12{,}4 + 6\} = 10;$$
$$d(6) := min\{d(6), d(3) + c_{36}\} = min\{9{,}4 + 6\} = 9;$$
$$d(8) := min\{d(8), d(3) + c_{38}\} = min\{\infty, 4 + 7\} = 11.$$

«Минимальной временной пометкой является пометка узла 4. Поэтому она становится постоянной. Подчеркиваем ее и y:=4. Помечаем дугу (2,4). Отметим это на сети (рисунок 6)» [15]:

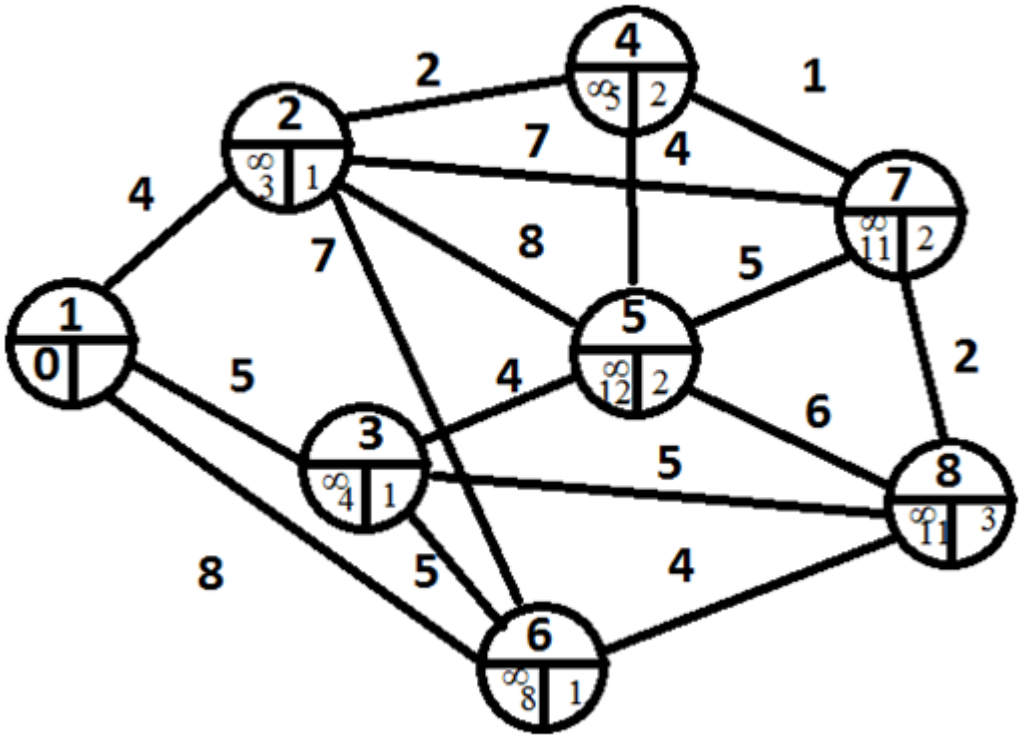

Рисунок 6 – Четвертый шаг решения задачи

«На четвертом шаге поскольку узел 8 не помечен постоянной пометкой, переходим к шагу 3. Узлы 5 и 7 смежные с последним, получившим постоянную пометку узлом 4. Новые значения временных пометок этих узлов будут» [15]:

$$d(5) := min\{d(5), d(4) + c_{45}\} = min\{10{,}5 + 5\} = 10;$$
$$d(7) := min\{d(7), d(4) + c_{47}\} = min\{11{,}5 + 1\} = 6.$$

«Минимальной временной пометкой является пометка узла 7. Поэтому она становится постоянной. Подчеркиваем ее и y:=7. Помечаем дугу (4,7). Отметим это на сети (рисунок 7)» [15]:

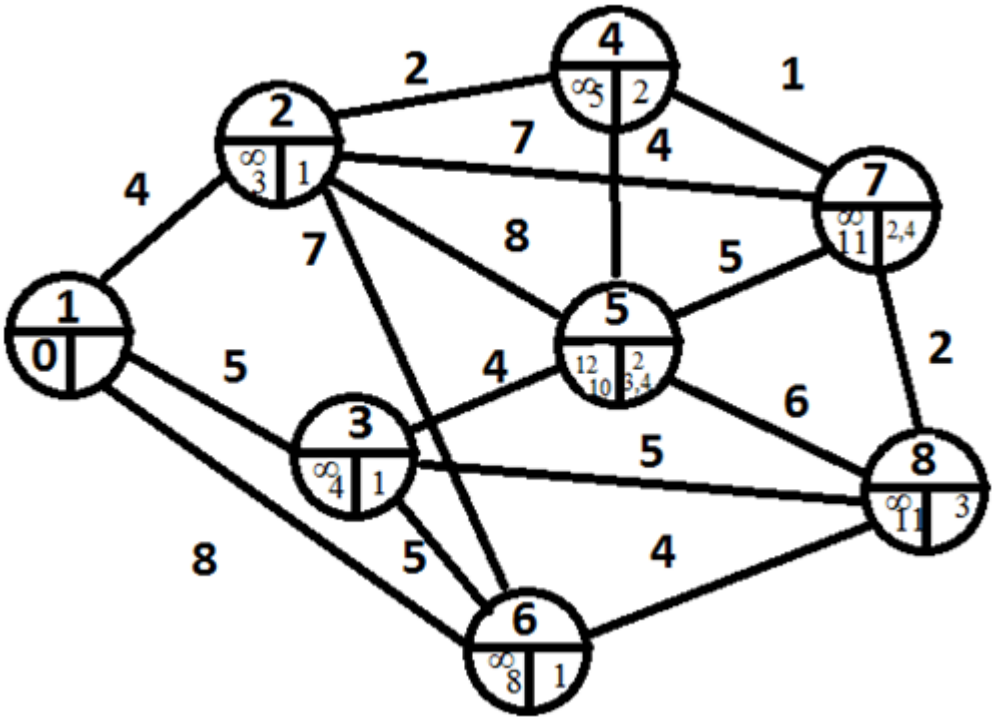

Рисунок 7 – Пятый шаг решения задачи

«На пятом шаге поскольку узел 8 не помечен постоянной пометкой, переходим к шагу 4. Узлы 5 и 8 смежные с последним, получившим постоянную пометку узлом 7. Новые значения временных пометок этих узлов будут» [15]:

$$d(5) := min\{d(5), d(7) + c_{75}\} = min\{10{,}6 + 6\} = 10;$$
$$d(8) := min\{d(8), d(7) + c_{78}\} = min\{11{,}6 + 2\} = 8.$$

«Минимальной временной пометкой является пометка узла 8. Поэтому она становится постоянной. Подчеркиваем ее и y:=8. Помечаем дугу (7,8). Отметим это на сети (рисунок 8)» [15]:

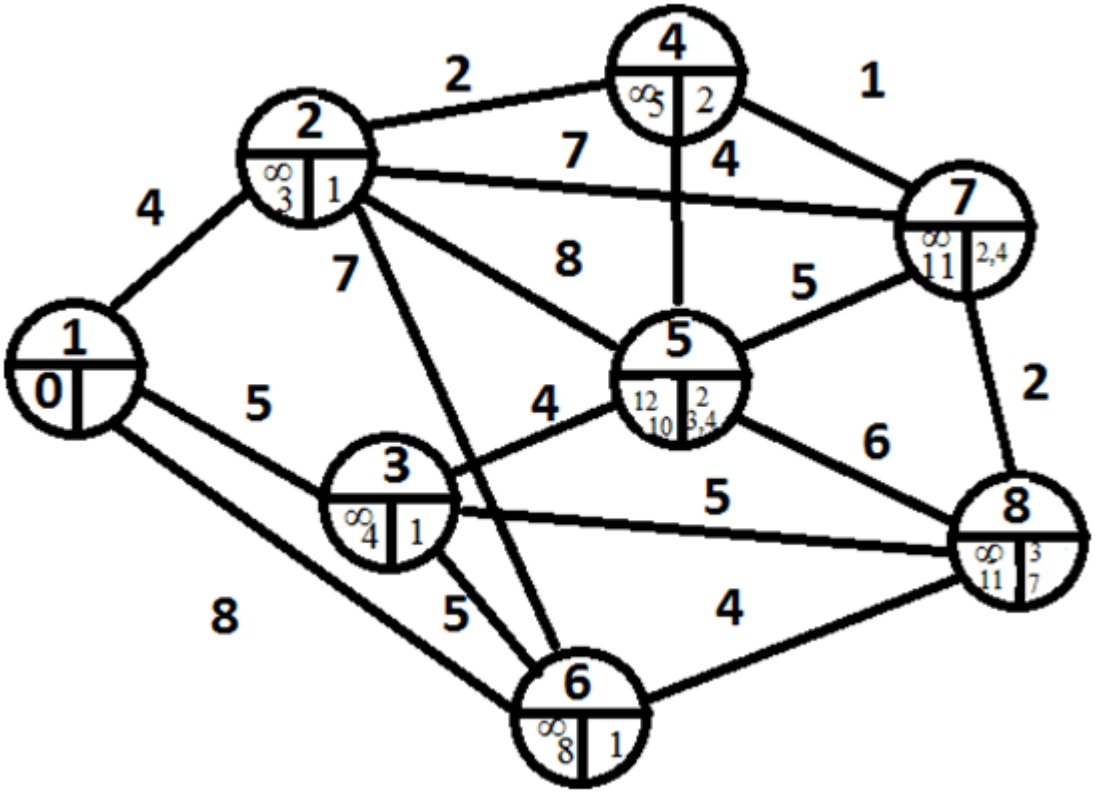

Рисунок 8 – Шестой шаг решения задачи

«На шестом шаге поскольку узел 8 помечен постоянной пометкой, то алгоритм завершает работу: кратчайший путь из узла 1 в узел назначения 8 найден.

Построенное дерево кратчайших путей состоит из дуг (1,2), (1,3), (2,4), (4,7), (7,8), они помечены на последней сети. Кратчайший путь от узла 1 до узла 8 составляет 8 км, т.к. постоянная пометка этого узла d(8)=8 и состоит из дуг (1,2), (2,4), (4,7), (7,8) » [15].

Рассмотрим задачу о запуске комплекса взаимосвязанных работ:

На предприятии необходимо запустить в эксплуатацию два комплекса взаимосвязанного оборудования. На рисунке 9 показана схема задачи.

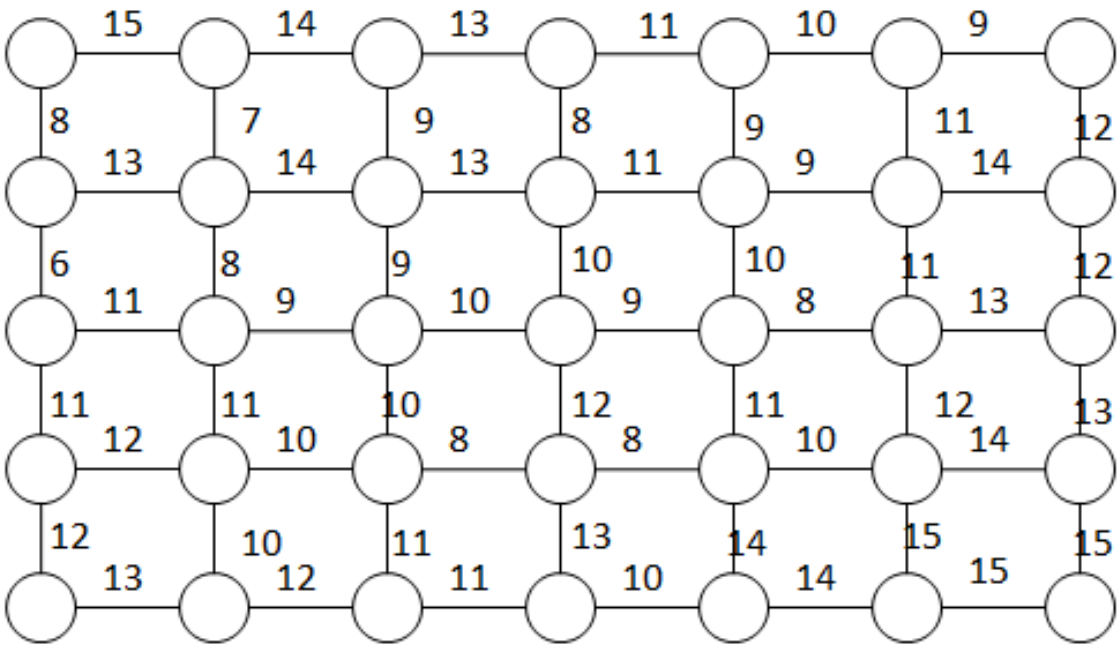

Рисунок 9 – Схема задачи

«Запуск 1–го комплекса состоит из шести промежуточных этапов, запуск 2–го комплекса состоит из четырех промежуточных этапов. Числа на ребрах, расположенных горизонтально, означают затраты (в тыс. рублей) при запуске определенного этапа 1–го комплекса. Числа на ребрах, расположенных вертикально, означают затраты (в тыс. рублей) при запуске определенного этапа 2–го комплекса. Вершины графа соответствуют возможным состояниям системы. Комплексы взаимосвязаны: затраты по запуску очередного этапа одного комплекса зависят от того на каком этапе находится запуск другого комплекса. Работы на двух комплексах одновременно не ведутся [20].

Необходимо найти управление последовательностью этапов запуска комплексов, при котором общие расходы были бы минимальными» [19].

Решение:

«По горизонтали отметим этапы запуска 1–го комплекса, по вертикале – этапы запуска 2–го комплекса. Вершина x_0 соответствует начальному состоянию системы (начало запуска). Вершина x_{10} соответствует конечному состоянию системы (запуск обоих комплексов). Весь процесс разбивается на 6+4=10 этапов (шагов).

Задачу можно решить следующим образом: рассмотреть все возможные траектории от x_0 до x_{10}, определить затраты для каждой траектории и выбрать ту траекторию, вдоль которой затраты будут наименьшими. Решение на первый взгляд простое, но если число этапов достаточно велико, то перебор всех траекторий, даже при использовании компьютерной техники, займет очень много времени.

Решим эту задачу, используя принцип оптимальности. Оптимизацию начнем с последнего, 10–го шага. В вершину x_{10} можно попасть либо из вершины A_1, либо из вершины A_2. Если на предпоследнем девятом шаге система была в состоянии A_1 (завершен запуск 2–го комплекса и завершен предпоследний этап запуска 1-го комплекса), то чтобы перейти в состояние x_{10}, необходимо выполнить запуск последнего этапа 1–го комплекса. Затраты при этом составят 9 тыс. рублей. Управление единственно, следовательно, оно и будет условно оптимальным. Аналогично, если на предпоследнем шаге система была в состоянии A_2, то управление так же единственно, и оно будет условно оптимальным. Соответствующие затраты запишем в вершинах графа, а условно оптимальные управления отметим стрелками (рисунок 10)» [19].

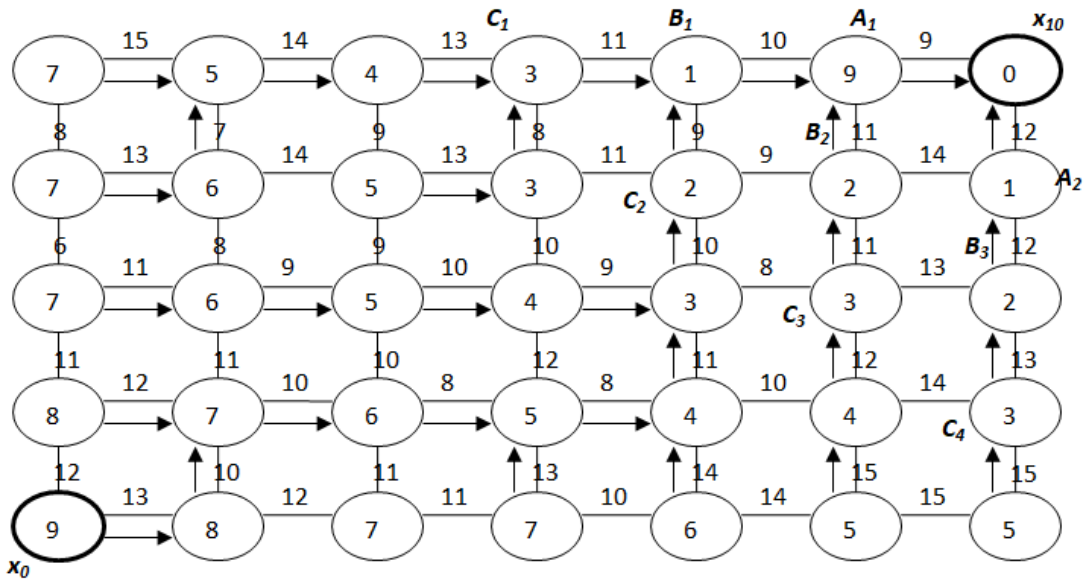

Рисунок 10 – Решение задачи принципом оптимальности

«Рассмотрим возможные состояния системы на восьмом шаге: B_1, B_2, B_3. Вершине B_1 соответствует единственное управление, оно проходит через вершину A_1, аналогично из вершины B_3 можно построить единственное управление, оно пройдет через вершину A_2.

Из вершины B_2 возможны два пути: либо через вершину A_1 (выполнить последний этап запуска 2-го комплекса, а затем последний этап запуска 1-го комплекса), либо через вершину A_2 (выполнить последний этап запуска 1-го комплекса, а затем последний этап запуска 2-го комплекса). В первом случае затраты составят 20 тыс. рублей, во втором – 26 тыс. рублей. Выберем условно оптимальное управление, соответствующее наименьшим затратам, через вершину A_1. Предполагаемые затраты 20 тыс. руб. занесем в вершину B_2, управление укажем стрелкой. Далее аналогично оптимизируем 8–й шаг. На этом шаге, например для вершины C_2 получим два условно оптимальных управления: через вершину B_1 и через вершину B_2. В первом случае затраты составят 28 тыс. руб., во втором 29 тыс. руб. Выберем управление, проходящее через вершину B_1. Аналогично, продолжая процесс для оставшихся шагов, придем в вершину x_0» [19].

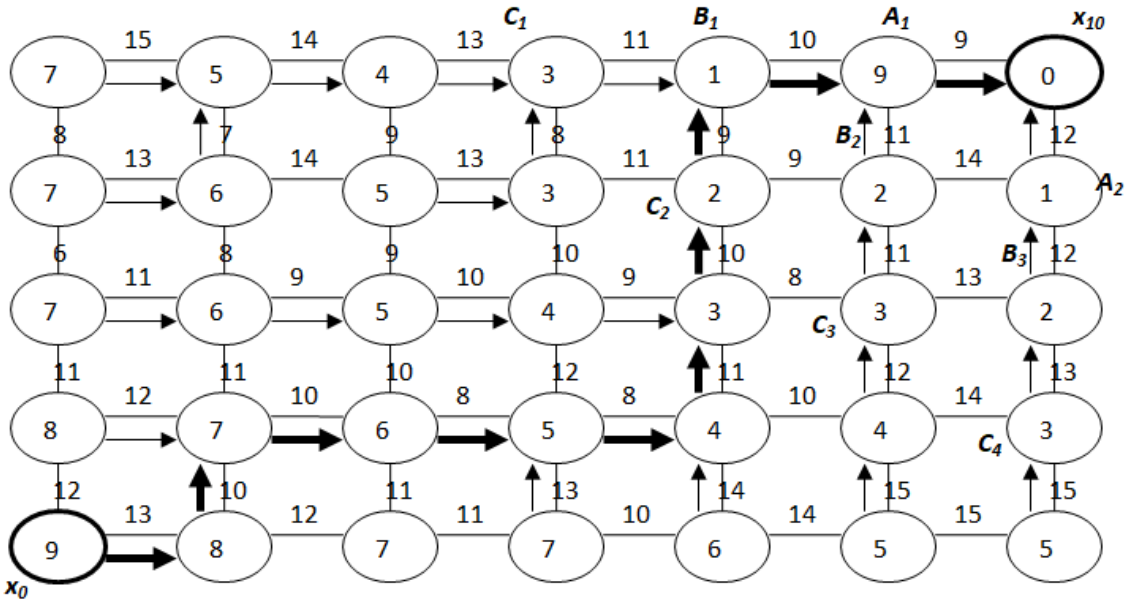

Рисунок 11 – Оптимальный путь

«Затем, двигаясь от вершины x_0 по условно-оптимальным управлениям (отметим это управление жирной стрелкой), придем в вершину x_{10} и получим оптимальное управление всего процесса (рисунок 11).

Наименьшие затраты по запуску двух комплексов оборудования составят 98 тыс. руб. Для того, чтобы достичь минимальных затрат необходимо выполнить запуск комплексов в следующем порядке: 1–ый этап первого комплекса; 1–ый этап второго комплекса; 2–ой, 3–ий, 4–ый этапы первого комплекса; 2–ой, 3–ий, 4–ый этапы второго комплекса; 5–ый, 6–ой этапы второго комплекса» [19].

Выводы по главе

Во второй главе построена математическая модель сетевой задачи и задачи о запуске комплекса взаимосвязанных работ, которые подчиняются алгоритмизации и могут быть решены методом динамического программирования.

Глава 3 Программная реализация алгоритмических задач

3.1 Анализ требований к программной реализации алгоритма

Для формирования требований к ПО мы будем использовать классификацию FURPS+. Данная классификация требований разработана Робертом Грэйди в 1992 году и успешно применяется для создания приложений и информационных систем в настоящее время. «Аббревиатура FURPS расшифровывается как Functionality (функциональность), Usability (удобство использования), Reliability (надёжность), Performance (производительность) и Supportability (поддерживаемость). Затем в классификацию был добавлен символ «+», который обозначает различные дополнительные факторы и ограничения, влияющие на разработку, такие как юридические вопросы, ограничения интерфейса, проектирования, физические ограничения и т.д.» [18]. Схематичное представление классификации FURPS+ представлено на рисунке 12.

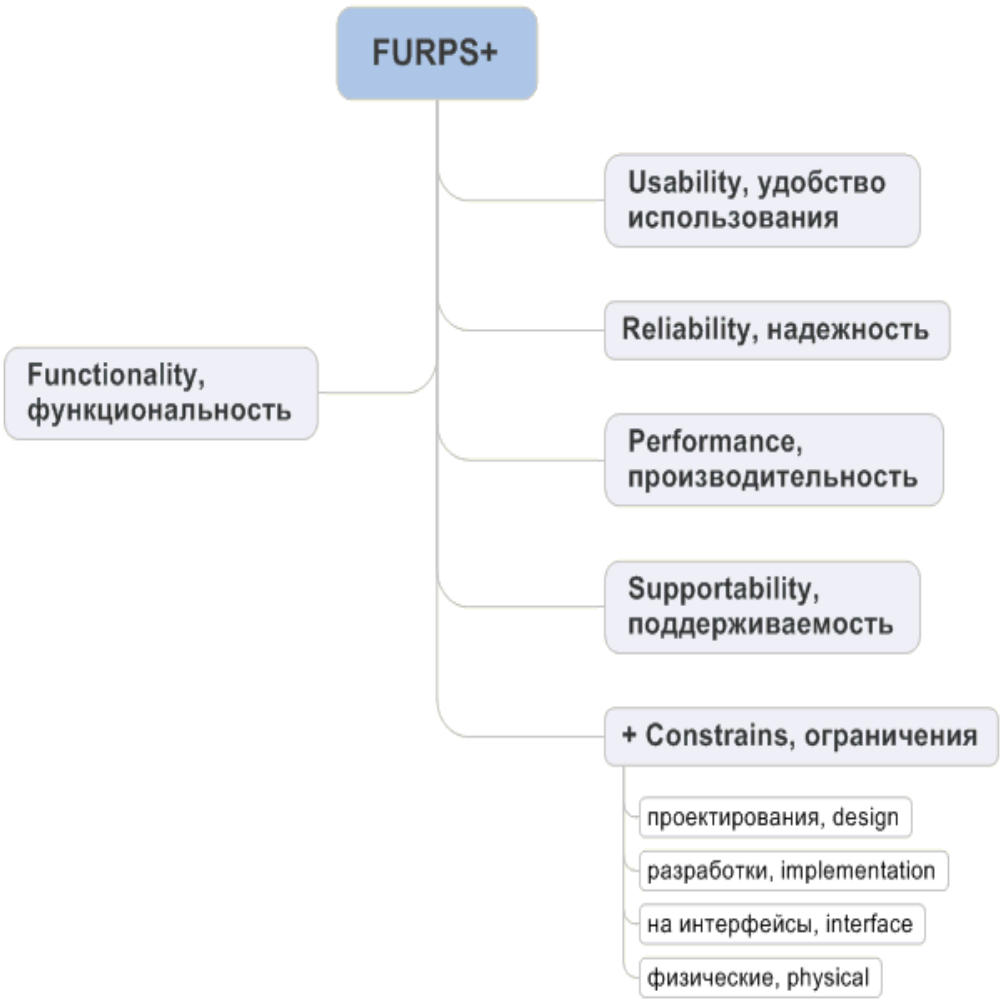

Рисунок 12 – Схематическое представление классификации требований FURPS+

Основа любого приложения или системы, согласно классификации FURPS+, это её функционал, то есть то, для чего предназначена система, и какие возможности в ней реализованы. На основании функциональных требований строятся диаграммы вариантов использования.

Перечислим требования к проектируемой программе. Functionality: решение алгоритмических задач. Usability: удобный и понятный интерфейс, контекстная справка, комфортная цветовая гамма программы, читабельные и понятные шрифты. Reliability: низкая периодичность сбоев. Performance: время запуска программы не более 3-х секунд, время отклика программы на

действия пользователя не более 1 секунды. Supportability: возможность расширения функционала системы.

3.2 Проектирование алгоритма задачи

Процесс проектирования программы является одной из ключевых стадий ее жизненного цикла [1].

Проектирование программы осуществляется с учетом требований, сформулированных в предыдущем разделе.

На рисунке 13 представлена блок-схема алгоритма, реализуемого проектируемой программой.

Предварительно опишем входные данные программы. Входными данными программы являются [6]:

- Начальное состояние системы S;
- Количество вариантов эффективного вложения n;
- Количество объектов распределения p;
- Критерии эффективности для каждого предприятия относительно вложения.

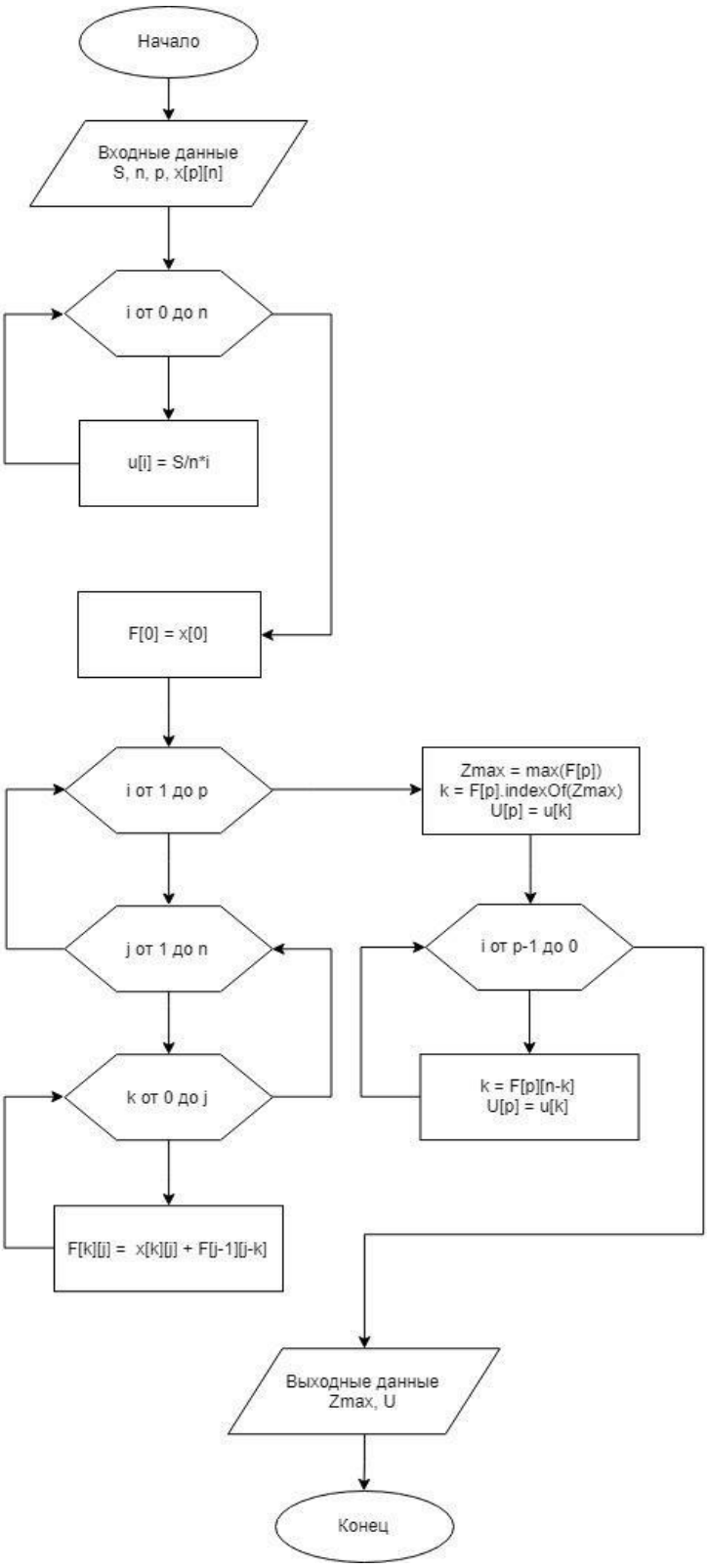

Рисунок 13 – Блок-схема алгоритма программы

Для реализации программы необходимо рассмотреть структурную схему программы (рисунок 14).

Программный продукт имеет конструкцию построения – состав и взаимосвязь программный модулей. Пример структурной схемы программы представлен на рисунке.

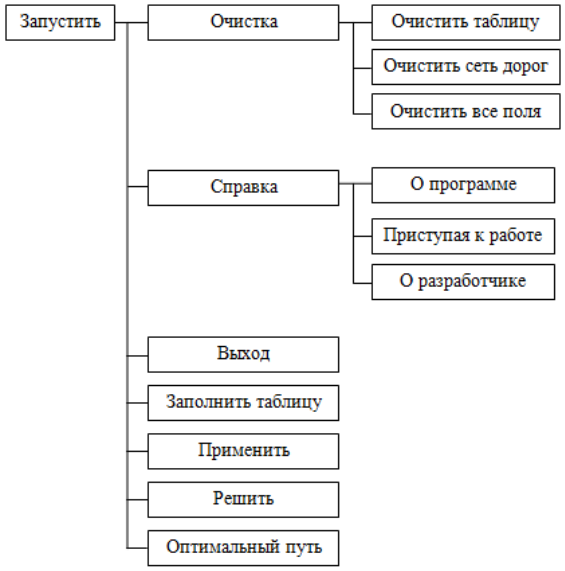

Рисунок 14 – Структурная схема программы

Программа содержит 3 модуля. Схема взаимодействия модулей программы изображена на рисунке 15.

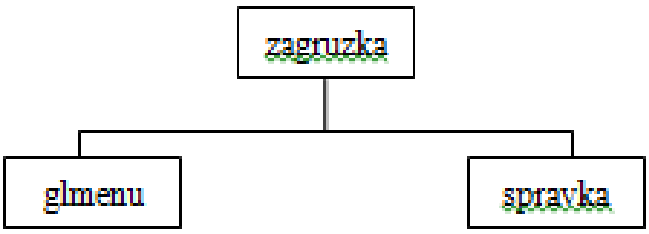

Рисунок 15 – Структурная схема программы

3.3 Выбор языка программирования

Для реализации программы необходимо выбрать, какой язык программирования будем использовать. Для этого сделаем сравнения по нескольким критериям. Приоритет выбора языка представлен значениями от 1 до 5, где 1 – наивысший приоритет, 5 – наименьший. В таблице 1 проводится сравнительный анализ языков программирования [10],[11].

Таблица 1 – Сравнение языков программирования

Критерии\язык	C++	C#	Java	Python	Borland Delphi 7
Скорость работы	1	1	4	2	1
Объем занимаемой ОП	1	1	5	3	2
Скорость разработки	3	4	1	3	1
Ориентированность	2	2	2	2	2
Кроссплатформенность	2	2	1	4	3
Скорость тестирования	2	3	1	5	1

Сравнив языки программирования, было решено реализовывать программу в среде программирования Borland Delphi 7 [21].

«Borland Delphi 7 – это среда предназначена для быстрой (RAD) разработки прикладного ПО для операционных систем Windows, Linux, а также iOS и Android. Благодаря уникальной совокупности простоты языка и генерации машинного кода позволяет непосредственно, и, при желании, достаточно низкоуровневого взаимодействовать с операционной системой, а также с библиотеками, написанными на C++» [17],[22].

Небольшие части кода программы в среде программирования Borland Delphi 7, продемонстрированы в листинге 1 и в листинге 2.

Листинг 1 – Оператор Собеля

```
Mat sobelFilter(Mat &src)
{	Mat gradValue = src.clone();
	Mat gradDirect = src.clone();
	double sum1, sum2;
	double model1[9] = { -1, -2, -1, 0, 0, 0, 1, 2, 1 };
	double model2[9] = { -1, 0, 1, -2, 0, 2, -1, 0, 1 };
	int p;
	for (int i = 0; i < src.rows; i++)
		for (int j = 0; j < src.cols; j++)
		{
			p = 0, sum1 = 0, sum2 = 0;
					for (int n = -1; n <= 1; n++)
				for (int m = -1; m <= 1; m++)
				{

					if (i + n < 0 || j + m < 0 || i + n >= src.rows || j + m
>= src.cols)
						p++;
					else
					{
						sum1 += src.at<uchar>(i + n, j + m) *
model1[p];
						sum2 += src.at<uchar>(i + n, j + m) *
model2[p];
						p++;
					}
				}
```

Листинг 2 – Фильтр Гаусса

```
{
	Mat dst = src.clone();
	double** model = getGaussianArray(size, sigma);
	double sum;
		for (int i = 0; i < src.rows; i++)
		for (int j = 0; j < src.cols; j++)
		{
					sum = 0;
					for (int n = -size / 2; n <= size / 2; n++)
				for (int m = -size / 2; m <= size / 2; m++)
				{
					if (i + n < 0 || j + m < 0 || i + n >= src.rows || j + m
>= src.cols)
						sum += 0;
					else
						sum += model[n + size / 2][m + size / 2] *
src.at<uchar>(i + n, j + m);
				}
			dst.at<uchar>(i, j) = pixesDeal(sum);
		}		return dst;
}
```

3.4 Тестирование реализованной программы

Для тестирования программы используем метод функционального тестирования [5].

Благодаря этому методу, мы можем проверить нашу программу на работоспособность. Проверять программу мы будем по нескольким пунктам:

- запуск программы;
- построение графиков;
- работоспособность основных функций;
- вывод справки;
- корректный вывод результата программы [23].

Вначале мы запустим программу. На рисунке 16 показана загрузка программы.

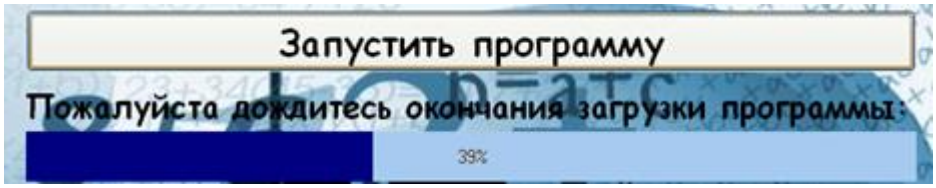

Рисунок 16 – Запуск программы

Дальше мы вводим данные, потом необходимо нажать на кнопку «Заполнить таблцу» и после этого нажмем кнопку «Применить» с помощью которой программа построит сеть дорог, дальше необходимо нажать «Оптимальный путь» и программа построит оптимальный путь и покажет длину пути. На рисунке 17 показано построение сети дорог.

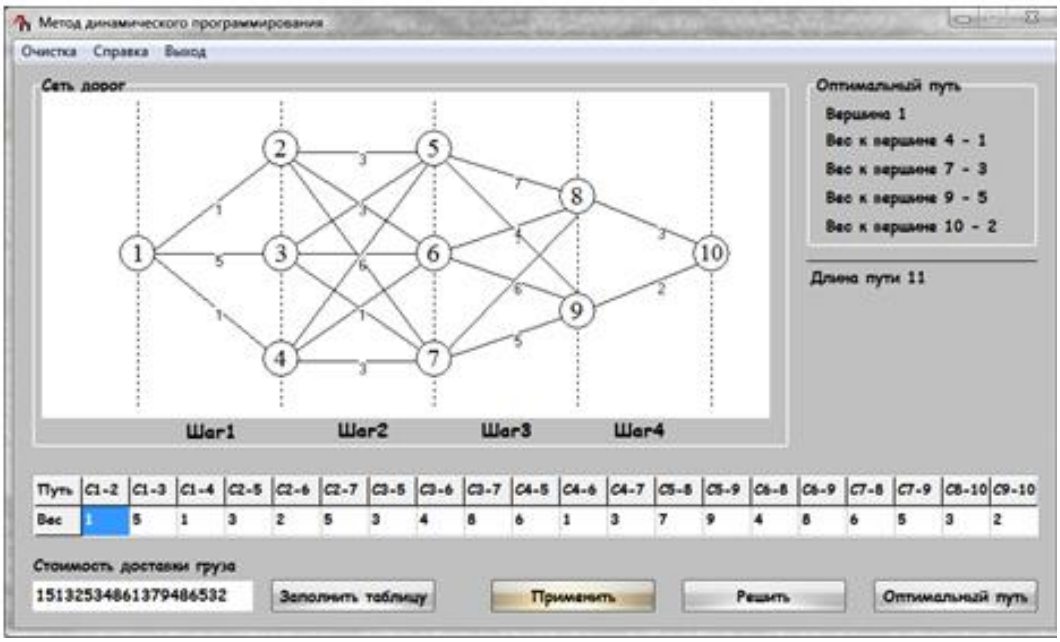

Рисунок 17 – Построение сети дорог

После построения схемы, нам необходимо запустить и программа сама оставит на графике оптимальный путь. На рисунке 18 показан оптимальный путь программы.

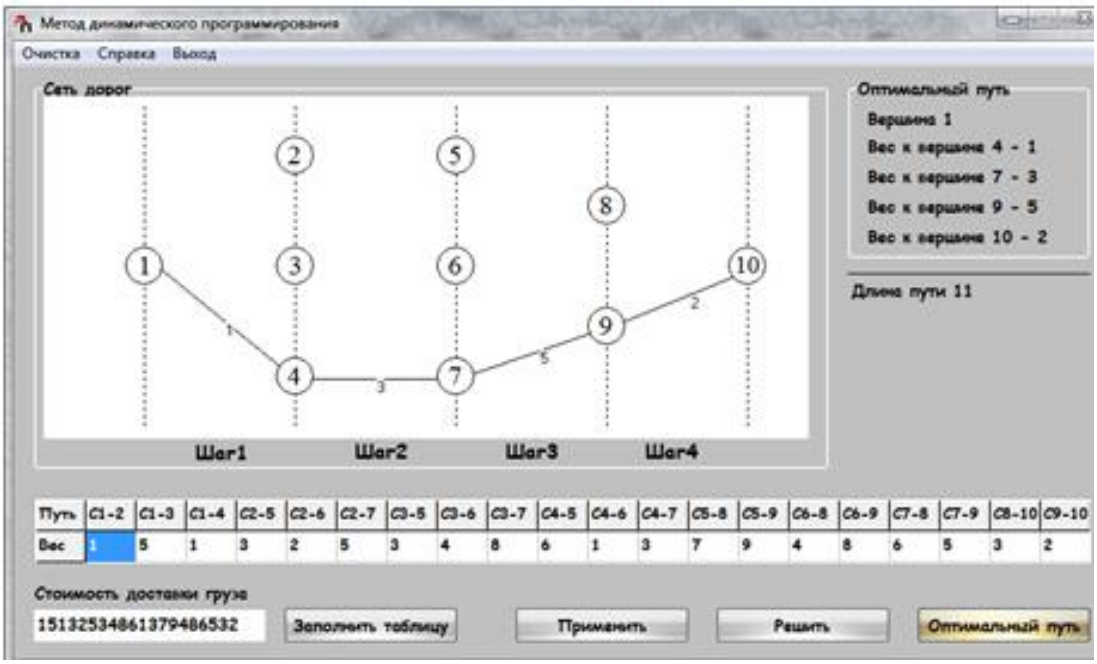

Рисунок 18 – Оптимальный путь

После нахождения оптимального пути, мы можем перейти в окно «О программе», где в справке видим, как вычислялся наш путь математически.

На рисунке 19 показана справка «О программе».

Рисунок 19 – Справка "О программе"

Выводы по главе

В третьей главе обоснованы требования к проектируемой программе; составлена блок-схема и описаны входные данные. Тестирование, созданного программного продукта, показало корректность работы программы.

Заключение

В выпускной квалификационной работе представлена разработка программного обеспечения для решения алгоритмических задач методом динамического программирования.

В первой главе описана сущность динамического программирования, была поставлена задача динамического программирования. Рассмотрели два основных подхода к решению задачи.

Приводится уравнение Беллмана, как основная концепция для решения алгоритмических задач методом динамического программирования. Разобрали несколько основных понятий уравнения Беллмана.

А так же рассмотрели особенности математической модели метода динамического программирования и перечислили свойства задач, допускающих применение метода динамического программирования.

Во второй главе рассмотрена математическая модель, где было выявлено, что при использовании динамического программирования многошаговая задача решается двумя разными способами: от конца к началу и от начала к концу.

А так же рассмотрели методы решения задач динамического программирования, где написали основное функциональное уравнение динамического программирования.

И было представлено решение задачи методом динамического программирования.

В третьей главе представлено проектирование для реализации программы, где была представлена блок-схема и описаны входные данные.

С помощью метода функционального тестирования мы выбрали несколько пунктов, по которым и протестировали программу. Проверили работоспособность и сделали вывод, что программа работает корректно, а значит успешно прошла тестирование.

Проделав работу, можно сказать, что актуальность исследованной темы полностью раскрылась. В настоящее время задачи принятия рациональных решений, оптимального управления и выбора наилучших вариантов решаются на предприятиях разных профилей и направлений работы. Решение алгоритмических задач с использованием методов динамического программирования является одним из наиболее эффективных методов оптимизации и занимает особое положение. Высокая эффективность и привлекательность этого метода достигается благодаря простоте основного принципа динамического программирования – принципа оптимальности.

Результаты работы могут быть рекомендованы для решения алгоритмических задач методом динамического программирования.

Список используемой литературы

1. А. Н. Чаплыгин. Учимся программировать вместе с Питоном. Учебник. – ревизия 226. – 135 с.
2. Бабаков Н.А. Теория линейных систем автоматического управления / Н.А. Бабаков, А.А. Воронов. Москва: Высшая школа, 1986. − 367 с.
3. Беллман Р. Динамическое программирование. – М.: Репринт оригинального издания иностранной литературы, 1960 год, 2012.
4. Гладких Б.А. Методы оптимизации и исследования операций. Часть Введение в исследование операций. Линейное программирование. – Томск: НТЛ, 2009. – 200 с.
5. Гома Х. UML-проектирование систем реального времени параллельных и распределенных приложений: [Пер. с англ.] / Х. Гома. 2-е изд. Москва: ДМК Пресс, 2011. − 704с.
6. Доусон М. Программируем на Python. – СПб.: Питер, 2012. –432 с.
7. Каллихман И.Л., Войтенко М.А. Динамическое программирование в примерах и задачах / И.Л. Каллихман, М.А. Войтенко. Москва: Высшая школа, 1979. − 124 с
8. Кормен, Т., Лейзерсон, Ч., Ривест, Р., Штайн, К. Глава 15. Динамическое программирование // Алгоритмы: построение и анализ = Introduction to Algorithms / Под ред. И. В. Красикова. – 2-е изд. – М.: Вильямс, 2005. – 1296 с.
9. Лежнев, А.В. Динамическое программирование в экономических задачах [Текст]: учеб. пособие / Лежнев А.В. - Москва: Бином, 2010. - 176 с.
10. Маккинли У. Python и анализ данных. – Перевод с английского. – М.: ДМК Пресс, 2015. – 482 с. – Sanjoy Dasgupta ,Christos H. Papadimitriou, Umesh Vazirani. Algorithms – Algorithms. – 1-е изд. – McGraw-Hill Science/Engineering/Math, 2006. – С. 336.

11. Марк Лутц. Программирование на Python / Пер. с англ. – 4-е изд. – СПб.: Символ-Плюс, 2011. – Т. I. – 992 с.

12. Марк Саммерфилд. Python на практике. – Перевод с английского. – М.: ДМК Пресс, 2014. – 338 с.

13. Некрасова М.Г. Методы оптимизации и теория управления: учебное пособие / М.Г. Некрасова. Комсомольск-на-Амуре: КНАГТУ, 2007. − 132 с.

14. Окулов С.М. Динамическое программирование. – М.:Бином. Лаборатория знаний, 2017. – 296 с.

15. Окулов С.М. Динамическое программирование / С.М. Окулов, О.А. Пестов. Москва: БИНОМ. Лаборатория знаний, 2012. − 260 с.

16. Самаров К.Л. Математика. Учебно-методическое пособие для студентов по разделу «Элементы теории графов. Динамическое программирование. Сетевое планирование» / К.Л. Самаров. Москва: Учебный центр «Резольвента», 2009. – 26с.

17. Фёдоров Д. Ю. Основы программирования на примере языка Python / Учебное пособие. – СПб.:Юрайт, 2018. – 167 с.

18. Шапкин А.С. Математические методы и модели исследования операций / А.С. Шапкин, В.А. Шапкин. Москва: Дашков и Ко, 2017. − 398 с.

19. Ширяев В.И. Исследование операций и численные методы оптимизации: учебное пособие / В.И. Ширяв. Москва: Ленанд, 2017. − 224 с.

20. [Электронный ресурс] Задачи ДП – Режим доступа: https://math.semestr.ru/dinam/dinam_manual.php.

21. Bertsekas, D. P. (2017), Dynamic Programming and Optimal Control (4th ed.), Athena Scientific.

22. Giegerich, R.; Meyer, C.; Steffen, P. (2004), "A Discipline of Dynamic Programming over Sequence Data" (PDF), Science of Computer Programming, 51 (3): 215–263, doi:10.1016/j.scico.2003.12.005.

23. Hamilton, Naomi (5 August 2008). "TheA-Zof Programming Languages: Python". Computer world. Archived from the original on 29 December

2008. Retrieved 31 March 2010.

24. Lutz, Mark (2013). Learning Python (5thed.). O'Reilly Media.

25. Meyn, Sean (2007), Control Techniques for Complex Networks, Cambridge University Press, archived from the original on 2010-06-19.