Министерство образования и науки Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к магистерской диссертации

«Децентрализованный алгоритм управления конвейерной системой с использованием методов мультиагентного обучения с подкреплением»

Автор: Мухутдинов Дмитрий Вадимович _____

Направление подготовки (специальность): 01.04.02 Прикладная математика и информатика

Квалификация: Магистр

Руководитель: Фильченков А.А., канд. физ.-мат. наук _____

К защите допустить

Зав. кафедрой Васильев В.Н., докт. техн. наук, проф. _____

«___» _____ 20___ г.

Санкт-Петербург, 2019 г.

Студент Мухутдинов Д.В. **Группа** М4239 **Кафедра** компьютерных технологий **Факультет** информационных технологий и программирования

Направленность (профиль), специализация Технологии проектирования и разработки программного обеспечения

Консультанты:

а) Вяткин В.В., докт. техн. наук, проф. _____

Квалификационная работа выполнена с оценкой _____

Дата защиты «15» июня 2019 г.

Секретарь ГЭК *Павлова О.Н.* Принято: «___» _____ 20___ г.

Листов хранения _____

Демонстрационных материалов/Чертежей хранения _____

Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ»

УТВЕРЖДАЮ
Зав. каф. компьютерных технологий
докт. техн. наук, проф.
_____ Васильев В.Н.
«___» _____ 20___ г.

ЗАДАНИЕ
НА МАГИСТЕРСКУЮ ДИССЕРТАЦИЮ

Студент Мухутдинов Д.В. **Группа** М4239 **Кафедра** компьютерных технологий **Факультет** информационных технологий и программирования **Руководитель** Фильченков А.А., канд. физ.-мат. наук, кафедра КТ

1 Наименование темы: Децентрализованный алгоритм управления конвейерной системой с использованием методов мультиагентного обучения с подкреплением

Направление подготовки (специальность): 01.04.02 Прикладная математика и информатика

Направленность (профиль): Технологии проектирования и разработки программного обеспечения

Квалификация: Магистр

2 Срок сдачи студентом законченной работы: «31» мая 2019 г.

3 Техническое задание и исходные данные к работе.

Требуется разработать децентрализованный алгоритм управления конвейерной системы для транспортировки багажа. Алгоритм должен позволять контроллерам конвейерной сети динамически изменять свое поведение в целях адаптации под изменившиеся условия работы, такие как поломка одного из конвейеров или изменение интенсивности потока багажа. Алгоритм должен обеспечивать своевременную доставку багажных единиц до пунктов назначения, в то же время минимизируя энергопотребление всей системы в целом.

4 Содержание магистерской диссертации (перечень подлежащих разработке вопросов)

Пояснительная записка должна содержать обзор существующих результатов в сфере управления конвейерными системами, а также в сферах, имеющих непосредственное отношение к предложенному алгоритму (таких как обучение с подкреплением). Также записка должна содержать подробное изложение предложенного алгоритма и данные экспериментального сравнения его производительности с производительностью существующих методов управления конвейерной системой, проведенного с помощью виртуальной имитационной модели конвейерной сети.

5 Перечень графического материала (с указанием обязательного материала)

Не предусмотрено

6 Исходные материалы и пособия

а) Richard S. Sutton and Andrew G. Barto. Reinforcement Learning: An Introduction. The MIT Press, 2012

б) Mnih et al. Human-level control through deep reinforcement learning. Nature, 518(7540):529–533, 2015.

7 Календарный план

№№ пп.	Наименование этапов магистерской диссертации	Срок выполнения этапов работы	Отметка о выполнении, подпись руков.
1	Ознакомление с предметной областью	11.2017	
2	Чтение статей, посвященных алгоритмам маршрутизации	01.2018	
3	Чтение статей, посвященных задаче обучения с подкреплением	02.2018	
4	Чтение статей, посвященных задаче обучения с подкреплением	03.2018	
5	Чтение статей, посвященных задаче управления конвейерными системами	05.2018	
6	Разработка имитационной модели конвейерной сети	09.2018	
7	Реализация существующих алгоритмов управления	11.2018	
8	Разработка алгоритма маршрутизации, проведение экспериментов	03.2019	
9	Написание пояснительной записки	05.2019	

8 Дата выдачи задания: «01» сентября 2017 г.

Руководитель _____

Задание принял к исполнению _____ «01» сентября 2017 г.

Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ»

АННОТАЦИЯ
МАГИСТЕРСКОЙ ДИССЕРТАЦИИ

Студент: Мухутдинов Дмитрий Вадимович

Наименование темы работы: Децентрализованный алгоритм управления конвейерной системой с использованием методов мультиагентного обучения с подкреплением

Наименование организации, где выполнена работа: Университет ИТМО

ХАРАКТЕРИСТИКА МАГИСТЕРСКОЙ ДИССЕРТАЦИИ

1 Цель исследования: Разработка удобного стилевого файла LaTeXдля бакалавров и магистров кафедры компьютерных технологий.

2 Задачи, решаемые в работе:

а) соответствие титульной страницы, задания и аннотации шаблонам, принятым в настоящее время на кафедре;

б) соответствие содержательной части пояснительной записки требованиям ГОСТ 7.0.11-2011 «Диссертация и автореферат диссертации»;

в) относительное удобство в использовании — указание данных об авторе и научном руководителе один раз и в одном месте, автоматический подсчет числа тех или иных источников.

3 Число источников, использованных при составлении обзора: 3

4 Полное число источников, использованных в работе: 3

5 В том числе источников по годам

Отечественных			Иностранных		
Последние 5 лет	От 5 до 10 лет	Более 10 лет	Последние 5 лет	От 5 до 10 лет	Более 10 лет
0	0	0	0	1	2

6 Использование информационных ресурсов Internet: нет

7 Использование современных пакетов компьютерных программ и технологий: Программный код имитационной модели конвейерной сети и алгоритмов управления написан на языке Python 3.6. При разработке имитационной модели конвейерной системы была использована библиотека для дискретно-событийного моделирования SimPy. Программный код алгоритмов управления конвейерной системой использует библиотеки NetworkX, NumPy, SciPy, scikit-learn, PyTorch. Для проведения экспериментальных исследований использовалась система интерактивной разработки Jupyter Lab и библиотеки matplotlib, pandas, tqdm.

8 Краткая характеристика полученных результатов: Разработан алгоритм управления конвейерной системой на основе глубокого мультиагентного обучения с подкреплением. В ходе экспериментального исследования было установлено, что разработанный алгоритм превосходит существующие по качеству работы и способен адаптироваться к изменениям во внешней среде.

9 Гранты, полученные при выполнении работы: TBD: вписать грант?

10 Наличие публикаций и выступлений на конференциях по теме работы:

1 Multi-agent deep learning for simultaneous optimization for time and energy in distributed routing system / D. Mukhutdinov [и др.] // Future Generation Computer Systems. — 2019. — Т. 94. — С. 587–600.

Выпускник: Мухутдинов Д.В. _____

Руководитель: Фильченков А.А. _____

«___» _____ 20___ г.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В данном разделе размещается введение.

ГЛАВА 1. ПЕРВАЯ ГЛАВА

Пример ссылок в рамках обзора: [1–3]. Вне обзора: [**bellman**].

1.1. Таблицы

В качестве примера таблицы приведена таблица 1.

Таблица 1 – Таблица умножения (фрагмент)

–	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
2	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34
3	3	6	9	12	15	18	21	24	27	30	33	36	39	42	45	48	51
4	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64	68

Есть еще такое окружение `tabu`, его можно аккуратно растянуть на всю страницу. Приведем пример (таблица 2).

Таблица 2 – Таблица умножения с помощью `tabu` (фрагмент)

–	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
2	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34
3	3	6	9	12	15	18	21	24	27	30	33	36	39	42	45	48	51
4	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64	68

1.2. Рисунки

Пример рисунка (с помощью `TikZ`) приведен на рисунке 1. Под `pdflatex` можно также использовать `*.jpg`, `*.png` и даже `*.pdf`, под `latex` можно использовать Metapost. Последний можно использовать и под `pdflatex`, для чего в стилевике продекларированы номера картинок от 1 до 20.

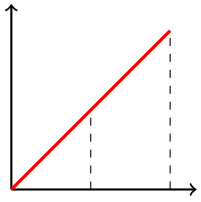

Рисунок 1 – Пример рисунка

1.3. Листинги

В работах студентов кафедры «Компьютерные технологии» часто встречаются листинги. Листинги бывают двух основных видов — исходный код и псевдокод. Первый оформляется с помощью окружения `lstlisting` из пакета `listings`, который уже включается в стилевике и немного настроен. Пример Hello World на Java приведен на листинге 1. Пример большого листинга — в приложении (листинг В.1).

Листинг 1 – Пример исходного кода на Java

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }
}
```

Псевдокод можно оформлять с помощью разных пакетов. В данном стилевике включается пакет `algorithmicx`. Сам по себе он не генерирует флоатов, поэтому для них используется пакет `algorithm`. Пример их совместного использования приведен на листинге 2.

Листинг 2 – Пример псевдокода

function IsPrime(N)
    for $t \leftarrow [2; \lfloor\sqrt{N}\rfloor]$ **do**
        if $N \bmod t = 0$ **then**
            return false
        end if
    end for
    return true
end function

Наконец, листинги из `listings` тоже можно подвешивать с помощью `algorithm`, пример на листинге 3.

Листинг 3 – Исходный код и флоат `algorithm`

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }
}
```

ГЛАВА 2. ПРОВЕРКА СКВОЗНОЙ НУМЕРАЦИИ

Листинг 4 должен иметь номер 4.

Листинг 4 – Исходный код и флоат `algorithm`

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }
}
```

Рисунок 2 должен иметь номер 2.

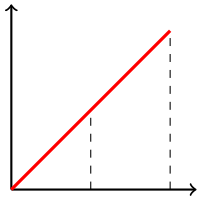

Рисунок 2 – Пример рисунка

Таблица 3 должна иметь номер 3.

Таблица 3 – Таблица умножения с помощью `tabu` (фрагмент)

–	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
2	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34
3	3	6	9	12	15	18	21	24	27	30	33	36	39	42	45	48	51
4	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64	68

Выводы по главе 2

В конце каждой главы желательно делать выводы. Вывод по данной главе — нумерация работает корректно, ура!

ЗАКЛЮЧЕНИЕ

В данном разделе размещается заключение.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 *Yan J.*, *Vyatkin V.* Distributed Software Architecture Enabling Peer to Peer Communicating Controllers // IEEE Transactions on Industrial Informatics. — 2013. — Vol. 9, no. 4. — P. 2200–2209.

2 *Boyan J. A.*, *Littman M. L.* Packet routing in dynamically changing networks: a reinforcement learning approach // Advances in Neural Information Processing Systems. — 1994. — No. 6. — P. 671–678.

3 *Tan M.* Multi-agent reinforcement learning: Independent vs. cooperative agents // Proceedings of the tenth international conference on machine learning. — 1993. — P. 330–337.

ПРИЛОЖЕНИЕ А. ПРИМЕР ПРИЛОЖЕНИЯ

В приложениях рисунки, таблицы и другие подобные элементы нумеруются по приложениям с соответствующим префиксом. Проверим это.

Листинг А.1 должен иметь номер А.1.

Листинг А.1 – Исходный код и флоат `algorithm`

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }
}
```

Рисунок А.1 должен иметь номер А.1.

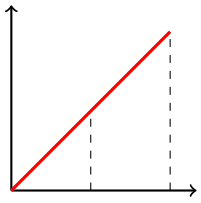

Рисунок А.1 – Пример рисунка

Таблица А.1 должна иметь номер А.1.

Таблица А.1 – Таблица умножения с помощью `tabu` (фрагмент)

–	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
2	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34
3	3	6	9	12	15	18	21	24	27	30	33	36	39	42	45	48	51
4	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64	68

Заодно проверим нумерованные и ненумерованные перечисления. Ненумерованные:

— пункт А;
— пункт Б;
— пункт В.

Нумерованные списки нескольких уровней:

а) первый элемент;
б) второй элемент с подэлементами:
    1) первый подэлемент;

2) второй подэлемент;
3) третий подэлемент.

в) третий элемент;
г) четвертый элемент;
д) пятый элемент;
е) шестой элемент;
ж) седьмой элемент;
и) восьмой элемент;
к) девятый элемент;
л) десятый элемент.

ПРИЛОЖЕНИЕ Б. ЕЩЕ ОДИН ПРИМЕР ПРИЛОЖЕНИЯ С НЕИМОВЕРНО ДЛИННЮЩИМ НАЗВАНИЕМ ДЛЯ ТЕСТИРОВАНИЯ ПЕРЕНОСОВ

Проверим на примере таблиц, что нумерация в приложениях — по приложениям. Таблица Б.1 должна иметь номер Б.1.

Таблица Б.1 – Таблица умножения с помощью `tabu` (фрагмент)

–	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
2	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34
3	3	6	9	12	15	18	21	24	27	30	33	36	39	42	45	48	51
4	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64	68

ПРИЛОЖЕНИЕ В. ПРИМЕР ОГРОМНОГО ЛИСТИНГА

Листинг В.1 – Пример большого листинга

```
import java.util.*;

public class Example {
    static int[] restoreOutgoing(int[] g, int[] outgoing,
                                 int vertex, int mask) {
        int[] rv = new int[1 + Integer.bitCount(mask)];
        int n = g.length;
        int current = rv.length - 1;
        while (true) {
            rv[current] = vertex;
            if (current == 0) {
                if (vertex != 0) {
                    throw new AssertionError();
                }
                return rv;
            }
            mask ^= 1 << (vertex - 1);
            int prevMask = outgoing[mask] & g[vertex];
            if (prevMask == 0) {
                throw new AssertionError();
            }
            vertex = Integer.numberOfTrailingZeros(prevMask);
            --current;
        }
    }

    static int[] restoreIncoming(int[] g, int[] incoming,
                                 int vertex, int mask) {
        int[] rv = new int[1 + Integer.bitCount(mask)];
        int n = g.length;
        int current = 0;
        while (true) {
            rv[current] = vertex;
            if (current == rv.length - 1) {
                if (vertex != 0) {
                    throw new AssertionError();
                }
                return rv;
            }
```

```
                mask ^= 1 << (vertex - 1);
                int nextMask = incoming[mask] & g[vertex];
                if (nextMask == 0) {
                    throw new AssertionError();
                }
                vertex = Integer.numberOfTrailingZeros(nextMask);
                ++current;
            }
        }
    }
```