ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра автоматизированных систем управления

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе по дисциплине

«Теория принятия решений»

Выполнил

ст. гр. ИС-20а \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Закотнюк Михаил Анатольевич

(подпись) (ФИО)

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ к.т.н., доц., Савкова Е.О.

(подпись) (ФИО)

Дата защиты \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Донецк 2023 г.

ГОУ ВПО «ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра автоматизированных систем управления

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

по дисциплине «Теория принятия решений»

Студенту группы ИС-20а Закотнюку Михаилу Анатольевичу

В карьере работают 6 экскаваторов - 3 первого типа и 3 второго. В течение каждой смены с вероятностью 0.1 каждый экскаватор может не работать. Грузовики вывозят руду из карьера в отвал.

Время погрузки подчиняется нормальному закону с параметрами: m1 =30 мин, σ1 =10 мин; m2=20 мин, σ2 =5 мин.

Во время погрузки каждого грузовика экскаватор может выходить из строя, время его ремонта подчинено равномерному закону (a=10мин, b=30мин). Если в это время есть свободные экскаваторы, то грузовик переезжает, но время его загрузки увеличивается на треть от основного времени, если свободных экскаваторов нет - то ожидает конца ремонта и догружается.

Время разгрузки в отвале имеет нормальный закон- m=5 мин, σ =1 мин и одновременно может разгружаться два грузовика.

Время движения подчинено экспоненциальному закону с mо→k =10 мин, mk→о =20 мин.

Во время движения грузовики могут выходить из строя: при движении в отвал Р=0.1; при движении назад Р=0.05. Поломки могут быть 2-х типов с

вероятностями 0.4 и 0.6. Время ремонта поломок первого типа имеет экспоненциальный закон с m=10 мин., второго - m=40 мин.

Сколько необходимо грузовиков, чтобы карьер работал синхронно.

Критерии оценки альтернатив: минимальная стоимость грузовиков, минимальное время простоя экскаваторов и минимальное время ожидания разгрузки.

Стоимость грузовика – 480 тыс. руб.

Средство решения – методы статистического имитационного моделирования и теории принятия решений.

Средство разработки моделирующей программы – язык программирования C++.

Дата выдачи задания: 01.02.2023

Срок выполнения:

Руководитель

(Подпись) (должность, ФИО)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| РЕФЕРАТ Курсовая работа: 53 с., 19 рис., 3 табл., 7 источников.  МОДЕЛИРОВАНИЕ СИСТЕМ, СИСТЕМА МАССОВОГО ОБСЛУЖИВАНИЯ, ТЕОРИЯ ПРИНЯТИЯ РЕШЕНИЙ, АЛЬТЕРНАТИВЫ, КРИТЕРИИ, C++  Цель курсовой работы – моделирование процессов и последующее применение методов для решение многокритериальных задач из теории принятия решений. Моделируемый объект – работа грузовых автомобилей. В ходе изучения индивидуального задания, были выделены следующие критерии:   * Минимальная стоимость грузовиков * Минимальное время простоя экскаваторов * Минимальное время ожидания разгрузки   Для определения лучшей альтернативы будут использованы следующие методы многокритериального решения задач:   * Аддитивная свёртка * Мультипликативная свёртка * Максиминный критерий * Главный критерий * Метод последовательных уступок * Целевое программирование | | | | | | | | | | |
|  |  |  |  |  | КР 09.03.02-20-262-001ПЗ | | | | | |
|  |  |  |  |  |
| Изм. | Лист | № документа | Подпись | Дата |
| Студент | | Закотнюк М. А. |  |  | Пояснительная записка | Лит | | | Лист | Листов |
| Рук. | | Савкова Е. О. |  |  |  | У |  | 4 | 53 |
| Консульт. | |  |  |  | ДонНТУ, ФИСТ,  Кафедра АСУ, гр. ИС-20а | | | | |
|  | |  |  |  |
| Н. контр. | |  |  |  |

СОДЕРЖАНИЕ

[РЕФЕРАТ 4](#_Toc137580326)

[ВВЕДЕНИЕ 6](#_Toc137580327)

[1. ПОСТАНОВКА ЗАДАЧИ МОДЕЛИРОВАНИЯ 7](#_Toc137580328)

[1.1. Формализованное описание объекта 7](#_Toc137580329)

[1.2. Выбор принципа построения моделирующего алгоритма 8](#_Toc137580330)

[1.3. Способы фиксации обработки результатов моделирования 10](#_Toc137580331)

[2. МОДЕЛИРОВАНИЕ СИСТЕМЫ 11](#_Toc137580332)

[2.1. Моделирование случайных факторов 11](#_Toc137580333)

[2.2. Разработка блок – схемы моделирующего алгоритма 15](#_Toc137580334)

[3. МЕТОДЫ РЕШЕНИЯ МНОГОКРИТЕРИАЛЬНЫХ ЗАДАЧ 23](#_Toc137580335)

[4. РЕЗУЛЬТАТ РЕАЛИЗАЦИИ МЕТОДОВ РЕШЕНИЯ МНОГОКРИТЕРИАЛЬНЫХ ЗАДАЧ 26](#_Toc137580336)

[5. ИНСТРУКЦИЯ ПОЛЬЗОВАТЕЛЯ 30](#_Toc137580337)

[ЗАКЛЮЧЕНИЕ 31](#_Toc137580338)

[СПИСОК ЛИТЕРАТУРЫ 32](#_Toc137580339)

[ПРИЛОЖЕНИЕ А. ПРОГРАММНЫЙ КОД И РЕЗУЛЬТАТЫ РЕАЛИЗАЦИИ РАБОТЫ СМО 33](#_Toc137580340)

**ВВЕДЕНИЕ**

При работе над сложными техническими системами, промышленными комплексами и экологическими проблемами, а также в других сферах жизни, человек сталкивается с задачами принятия правильных решений. Для облегчения этого процесса создаются и развиваются специальные математические модели и методы, объединенные в область исследований, называемую «Теория принятия решений». ТПР использует знания из областей математики, статистики, менеджмента, экономики и психологии для разработки оптимальных решений в разных областях деятельности.

Рациональный процесс решения задач и проблем включает в себя некоторые этапы, которые, при необходимости, могут выполнятся одновременно, параллельно или итерационно.

Из-за большого объема задач и сложности алгоритмов оптимизации, методы принятия оптимальных решений часто используются при помощи компьютеров.

Цель курсовой работы – провести анализ и разработать модель для реальной системы, чтобы найти оптимальное решение, используя навыки моделирования систем массового обслуживания и принятия решений.

1. **ПОСТАНОВКА ЗАДАЧИ МОДЕЛИРОВАНИЯ**
   1. **Формализованное описание объекта**

Для выполнения курсовой работы из математических схем моделирования была выбрана система массового обслуживания (далее – СМО), с помощью которой можно точно представить объект исходной задачи.

Системы массового обслуживания (СМО) представляют собой структуры, содержащие одно или несколько обслуживающих устройств – каналов обслуживания, на вход которых в случайные моменты времени поступают требования для обслуживания. В качестве каналов могут фигурировать технические устройства либо персонал, выполняющие функции обслуживания.

Каждая СМО предназначена для обслуживания какого-то потока заявок (или «требований»), поступающих на СМО в какие-то случайные моменты времени. Обслуживание поступившей заявки продолжается некоторое (вообще говоря, случайное) время, после чего канал освобождается и готов к принятию следующей заявки. Случайный характер потока заявок приводит к тому, что в какие-то промежутки времени на входе СМО скапливается излишне большое число заявок (они либо образуют очередь, либо покидают СМО необслуженными); в другие же периоды СМО будет работать с недогрузкой или вообще простаивать.

Предметом теории массового обслуживания является построение математических моделей, связывающих заданные условия работы СМО (число каналов, их производительность, характер потока заявок и т.п.) с показателями эффективности СМО, описывающими ее способность справляться с потоком заявок.

В качестве показателей эффективности СМО могут быть следующие параметры:

* среднее число заявок, обслуживаемых в единицу времени;
* среднее число заявок в очереди;
* вероятность отказа в обслуживании без ожидания;
* вероятность того, что число заявок в очереди превысит определенное значение и т.п.

В дальнейшем будет смоделирована замкнутая многоканальная СМО для последующего поиска оптимального решения. Каналами будут погрузочные экскаваторы в карьере.

В карьере осуществляется погрузка грузовиков экскаваторами. Погрузка осуществляется за разное время по нормальному закону распределения. Во время погрузки каждого грузовика экскаватор может выходить из строя, время его ремонта подчинено равномерному закону. Если в это время есть свободные экскаваторы, то грузовик переезжает, но время его загрузки увеличивается на треть от основного времени, если свободных экскаваторов нет - то ожидает конца ремонта и догружается.

Время разгрузки в отвале имеет нормальный закон и одновременно может разгружаться два грузовика.

Время движения подчинено экспоненциальному закону. Во время движения грузовики могут выходить из строя: при движении в отвал; при движении назад. Поломки могут быть 2-х типов с разными долями вероятности. Время ремонта поломок подчиняется экспоненциальному закону.

Обслуживанием заявки будет процесс погрузки грузовика (нормальный закон), его движение (экспоненциальный закон) и разгрузка (нормальный закон).

Заявкой будет являться грузовик, везущий руду из карьера в отвал.

* 1. **Выбор принципа построения моделирующего алгоритма**

В качестве главного принципа построения моделирующего алгоритма были выбраны принципы моделирования замкнутой СМО методом особых состояний. Грузовики не могут начать вывоз руды, не погрузив ее, и не могут покинуть место разгрузки, не разгрузив весь кузов. В такой системе будет отсутствовать моделирование постановки заявки в очередь и возможность отказа в обслуживании заявки.

Моделирование замкнутой многоканальной СМО с последовательной проводкой заявок содержит в себе некоторые фрагменты, при помощи которых моделируется одноканальные СМО.

Алгоритм моделирования замкнутой многоканальной СМО изображен на рисунке 1.1:

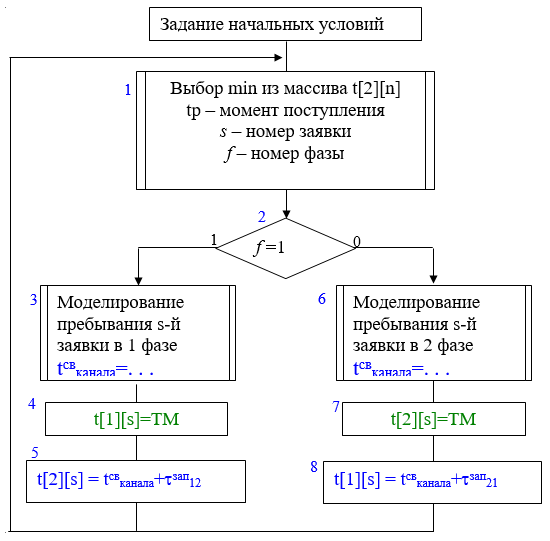


Рисунок 1.1 – Укрупненный моделирующий алгоритм замкнутой СМО

Блок 1 анализирует состояния таблицы заявок и определяет минимальный элемент в ней (min), номер его столбца и строки, тем самым определяя, когда наступит ближайший момент особого состояния, с какой заявкой по номеру -*s* и в какой из фаз - *f*.

Блок 2 разделяет алгоритм на две идентичных части – модель 1 фазы и модель второй фазы.

Блоки 3 и 6 соответственно моделируют пребывание выбранной заявки в соответствующей фазе.

Блоки 4 и 7 забивают выбранное минимальное значение достаточно большим, чтобы на следующем шаге цикла не выбрался тот же минимальный элемент.

Блоки 5 и 8 заносят в исходную таблицу новые момент поступления заявки в противоположную фазу, которые совпадают с моментами освобождения от обслуживания в текущей фазе.

Для поиска канала с минимальным временем освобождения используется простейший алгоритм поиска минимального элемента в массиве времён освобождения каждого канала.

* 1. **Способы фиксации обработки результатов моделирования**

При решении задачи в результате моделирования были определены параметры системы, которые удовлетворяют заданным критериям и ограничениям. В результате этого производится многократное моделирование системы, в котором подбираются значения входных параметров и производится анализ полученных результатов. После этого выбирается наилучший результат с помощью многокритериальных методов решения.

Из статистических входных данных присутствует математическое ожидание и среднее квадратичное отклонение, которые используются в законах распределения для формирования случайных величин. Также во входных данных присутствуют две величины a и b (начало и конец интервала), используемые для формирования псевдослучайной величины в диапазоне . Также, к входным параметрам относится количество экскаваторов.

К выходным параметрам относится количество использованных грузовиков, время простоя экскаваторов.

Время погрузки и разгрузки формируется по заданному нормальному закону.

Обработка результатов заключается в поиске оптимальной альтернативы с использованием методов решения многокритериальных задач.

1. **МОДЕЛИРОВАНИЕ СИСТЕМЫ**
   1. **Моделирование случайных факторов**

По индивидуальному заданию даны три закона формирования случайных чисел:

* Нормальный закон
* Экспоненциальный закон
* Равномерный закон

**Моделирование чисел с нормальным законом распределения**

Функция распределения вероятностей для нормального закона (формула 2.1, рис 2.1):

|  |  |
| --- | --- |
|  | (2.1) |

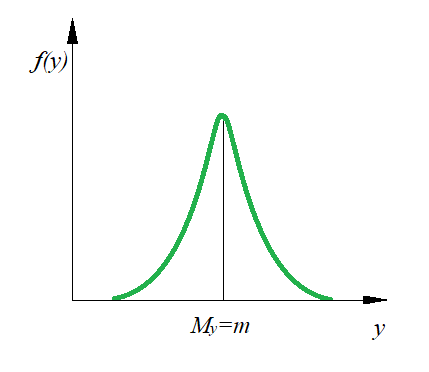


Рисунок 2.1 – Функция распределения для нормального закона

Для моделирования нормального распределения используется центральная граничная теорема: пусть дана некоторая последовательность случайных чисел с заданным законом распределения. Если из этой последовательности брать по n чисел и находить их сумму, то полученные в результате значения, при , будут подчинятся нормальному закону распределения.

Если математическое ожидание чисел *x* равно , то математическое ожидание результирующего числа *y* будет равняться . Аналогично для дисперсии и среднего квадратичного отклонения

Для получения выборки, которая будет удовлетворять нормальному закону распределения, достаточно сложить 8 – 12 чисел. Используя данную теорему, можно составить алгоритм для формирования чисел с нормальным законом распределения (рис. 2.2):



Рисунок 2.2 – Блок-схема нормального закона

где *a* и *b* считаются по формулам 2.2 – 2.3

|  |  |
| --- | --- |
|  | (2.2) |
|  | (2.3) |

**Моделирование чисел с экспоненциальным законом распределения**

Функция распределения вероятностей (формула 2.4, рис. 2.3)

|  |  |
| --- | --- |
|  | (2.4) |

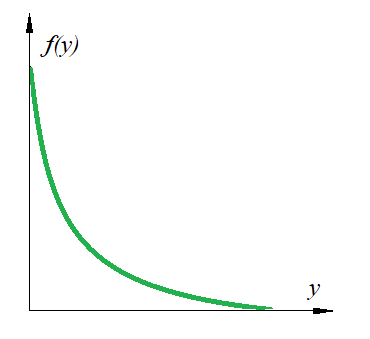


Рисунок 2.3 – Функция распределения вероятностей

Математическое ожидание связано с интенсивностью (формула 2.5):

|  |  |
| --- | --- |
|  | (2.5) |

Среднее квадратичное отклонение совпадает численно с математическим ожиданием (формула 2.6):

|  |  |
| --- | --- |
|  | (2.6) |

Выведем формулу для формирования случайных чисел с использованием экспоненциального закона распределения (формулы 2.7 – 2.9):

|  |  |  |
| --- | --- | --- |
|  | | (2.7) |
|  | (2.8) | |
|  | (2.9) | |

**Моделирование чисел с равномерным законом распределения**

Этот закон в природе встречается крайне редко, но является очень удобным при тестировании программ, используется для формирования других случайных факторов с последующим преобразованием и т. п.

Основные характеристики этого закона:

Функция распределения вероятностей  (2.10)

*y*

*b*

*f(y)*

*a*

Рисунок *2*.*4* – Функция распределения вероятностей

Математическое ожидание

|  |  |
| --- | --- |
|  | (2.11) |

Среднеквадратическое отклонение

|  |  |
| --- | --- |
|  | (2.12) |

Применим данный метод к равномерному закону

|  |  |
| --- | --- |
|  | (2.13) |

Конечная формула для использования

|  |  |
| --- | --- |
|  | (2.14) |

* 1. **Разработка блок – схемы моделирующего алгоритма**

Для реализации задачи будет удобно пользоваться вспомогательными алгоритмами поиска минимального элемента, например для поиска канала с минимальным временем освобождения. А также определение фазы. Блок-схемы этих алгоритмов показаны на рисунке 2.5.



Рисунок 2.5 – Блок-схемы вспомогательных модулей

Далее на рисунке 2.6 представлен алгоритм вспомогательной функции для расчета времени ремонта грузовика. Поломки могут быть 2-х типов с вероятностями 0.4 и 0.6, соответственно время ремонта поломок различается.



Рисунок 2.6 – Блок-схема расчета времени ремонта грузовика

Алгоритм моделирования замкнутой СМО представлен на рисунке 2.7, фазы представлены на рисунках 2.8 – 2.9.



Рисунок 2.7 – Блок-схема моделирующего алгоритма



Рисунок 2.8 – Блок-схема моделирования первой фазы



Рисунок 2.9 – Блок-схема моделирования второй фазы

Блок 1: цикл по количеству грузовиков.

Блок 2-4: цикл do-while до конца смены

Блок 3: определение фазы

**Начало 1-й фазы:**

Блок 5: выбор экскаватора с мин. временем погрузки

Блок 6: проверка на конец рабочего дня

Блок 7\*: если конец, то переход к следующей итерации по грузовикам.

Блок 8: проверка, занят ли выбранный экскаватор

Блок 9\*: если занят, время начала обслуживания считается как время освобождения обслуживания предыдущего камаза.

Блок 10: проверка на простой места

Блок 11: если место давно простаивает, увеличивается простой экскаватора

Блок 12: определяется тип экскаватора

Блок 13: расчет времени погрузки для первого типа экскаваторов

Блок 14: расчет времени погрузки для второго типа экскаваторов

Блок 15: моделируется вероятность поломки и время ремонта

Блок 16: проверка на выход экскаватора из строя

Блок 18\*: если не сломался, обновление времени освобождения

Блок 17\*: если сломался, осуществляется поиск резервного

Блок 19: проверка на наличие свободного резервного экскаватора

Блок 20\*: если есть свободный резервный экскаватор, то время погрузки увеличивается на треть от основного времени

Блок 21\*: если нет свободных, то время погрузки увеличивается на время ремонта.

Блок 22: время освобождения поломанного и резервного обновляются

Блок 23: моделируется время движения из карьера в отвал

Блок 24: формируется вероятность поломки грузовика в движении

Блок 25: проверка на выход грузовика из строя

Блок 26\*: если сломался, то время движения увеличивается на время ремонта

Блок 27: грузовик отправился в карьер.

**Начало 2-й фазы:**

Блок 28: выбор места с минимальным временем освобождения.

Блок 29: проверка, занято место или нет.

Блок 31\*: если место было занято, подсчет времени ожидания разгрузки как разницы между временем освобождения места и временем приезда на разгрузку грузовика.

Блок 32\*: если место было занято, грузовик обслуживается с момента его освобождения.

Блок 33: моделируется время разгрузки грузовика.

Блок 34: обновляется время освобождения текущего места разгрузки.

Блок 35: моделируется время движения грузовика в карьер на погрузку.

Блок 36: формируется вероятность поломки грузовика во время движения в карьер.

Блок 37: проверка на выход из строя.

Блок 38\*: если сломался, то время движения увеличивается на время ремонта поломки.

Блок 39: грузовик движется в карьер.

**Разработка программы для реализации моделирующего алгоритма**

Краткое описание переменные, использованных в программе, моделирующей СМО, приведено в таблице 2.1

Таблица 2.1 – Описание переменных

|  |  |
| --- | --- |
| **Переменная** | **Значение в программе** |
| file | Файл |
| COUNT\_GRUZ | Количество альтернатив (2 – 11) |
| f | фаза |

Продолжение таблицы 2.1

|  |  |
| --- | --- |
| **Переменная** | **Значение в программе** |
| tekGruz | Номер текущего обсуживаемого грузовика |
| tekExav | Номер текущего экскаватора (1-6) |
| rezervExav | Номер резервного экскаватора (1-6) |
| tekRazgr | Номер места разгрузки (1 или 2) |
| nGruz | Количество грузовиков в текущей итерации |
| prostoiEx | Суммарный простой экскаваторов в текущей итерации |
| ozidanieRazgr | Суммарный простой мест на разгрузке |
| ozidRazgr[nGruz] | Ожидание разгрузки по каждой альтернативе |
| prEx[nGruz] | Простой экскаваторов по каждой альтернативе |
| pogr\_osv[6] | Места погрузки |
| razgr\_osv[2] | Места разгрузки |
| r | Случайная величина |

Анализ результатов выполнения программы приведён в таблице 2.2.

Таблица 2.2 – Результаты

|  |  |  |
| --- | --- | --- |
| Входные данные | Выходные данные | |
| Кол-во аппаратов | Простой экскаваторов (мин.) | Ожидание разгрузки (мин.) |
| 2 | 2471,03 | 0,00 |
| 3 | 2298,03 | 7,98 |
| 4 | 2191,54 | 22,25 |
| 5 | 2070,44 | 53,41 |
| 6 | 1935,51 | 115,01 |
| 7 | 1827,74 | 209,37 |
| 8 | 1723,38 | 332,86 |
| 9 | 1632,55 | 519,45 |
| 10 | 1594,84 | 767,83 |
| 11 | 1535,68 | 1047,54 |

Полный листинг программы на языке программирования C++ и информация о работе СМО приведена в приложении А.

1. **МЕТОДЫ РЕШЕНИЯ МНОГОКРИТЕРИАЛЬНЫХ ЗАДАЧ**

Для выбора наилучшей альтернативы в многокритериальной задаче необходимо оценить каждое решение на основе нескольких критериев из множества доступных. Основной сложностью в данном процессе является неоднозначность выбора наилучшего решения.

Существует большое количество методов решения многокритериальных задач, часть из которых представлена на рисунке 3.1:



Рисунок 4.1 – Методы решения многокритериальных задач

При решении задач МКО приходится решать специфические вопросы, связанные с неопределенностью целей и несоизмеримостью критериев. Основные проблемы, возникающие при разработке методов МКО.

1. Проблема нормализации критериев, то есть приведение критериев к единому (безразмерному) масштабу измерения.
2. Проблема выбора принципа оптимальности, то есть установление, в каком смысле оптимальное решение лучше всех остальных решений.
3. Проблема учета приоритетов критериев, возникающая в тех случаях, когда из физического смысла ясно, что некоторые критерии имеют приоритет над другими.
4. Проблема вычисления оптимума задачи МКО. Т.е., как использовать методы линейной, нелинейной, дискретной оптимизации для вычисления оптимума задач с определенной спецификой.

Нормализация критериев – это приведение всех критериев к единому

масштабу и безразмерному виду.

Наиболее часто используется замена критериев их безразмерными

относительными величинами (формулы 3.1 и 3.2):

|  |  |
| --- | --- |
|  | (3.1) |
|  | (3.2) |

Нормализованные критерии обладают двумя важными свойствами:

* они являются безразмерными величинами;
* они удовлетворяют неравенству:

Эти свойства позволяют сравнивать критерии между собой.

Но, для некоторых методов решения многокритериальных задач необходимо, чтобы значения критериев были строго положительными и при их нормализации недопустимым является нулевое значение. В этом случае:

* увеличивают значения критериев на величину, позволяющую получить неотрицательные значения;
* выполняют нормализацию критериев по формулам 3.3 или 3.4:

|  |  |
| --- | --- |
|  | (3.3) |
|  | (3.4) |

Использование формул 3.3 и 3.4 приводит к тому, что

При решении многокритериальных задач предпочтение отдают методу аддитивной свёртки, если существенное значение для рассматриваемой задачи имеют абсолютные значения критериев для выбранного набора параметров.

Метод главного критерия позволяет выбрать один критерий в качестве главного, по мнению ЛПР, а остальные критерии сделать ограничениями.

Если необходимо решить задачу достижения равенства нормированных значений противоречивых частных критериев, то выбирают максиминный критерий.

Метод мультипликативной свёртки целесообразно использовать в тех случаях, когда существенную роль играет изменение абсолютных значений отдельных критериев при вариации искомого параметра.

При этом, метод целевого программирования является наиболее перспективным, так как он учитывает минимальные отклонения каждого критерия от исходного значения.

Однако, многокритериальные методы решения имеют некоторые недостатки:

* рост числа несравнимых альтернатив
* невозможность дать объяснения полученных результатов, при изменении методов, использующих свёртку критериев, поскольку по изменённым показателям трудно восстановить исходные данные
* большие трудозатраты ЛПР при построении функции полезности, определении весов критериев и сравнении альтернатив.

1. **РЕЗУЛЬТАТ РЕАЛИЗАЦИИ МЕТОДОВ РЕШЕНИЯ МНОГОКРИТЕРИАЛЬНЫХ ЗАДАЧ**

При выполнении курсовой работы для решения многокритериальной задачи были использованы следующие методы:

* Максиминный критерий;
* Аддитивная свёртка;
* Мультипликативная свёртка;
* Метод уступок;
* Главный критерий;
* Целевое программирование

Решение поставленной многокритериальной задачи производилось с использованием языка программирования C++. Код программы, для реализации всех используемых методов решения многокритериальных задач приведён в приложении А.

В исходной матрице для сравнения альтернатив имеется три критерия C1 (минимальная стоимость грузовиков), C2 (минимальный простой экскаваторов) и С3 (минимальное время ожидания разгрузки). После завершения работы программы, были получены 10 альтернативных решений(А1- 2 грузовика, А10 – 11 грузовиков), которые представлены на рисунке 4.1:

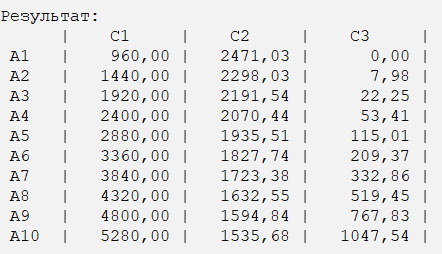


Рисунок 4.1 – Исходная матрица

Далее применим метод аддитивной свёртки. Для этого присваиваем коэффициентам важности α1, α2 и α3 значения, которые зависят от уровня важности критериев C1, С2 и C3 соответственно (0.6, 0.3, 0.1). Также, необходимо нормализовать матрицу при помощи формул 3.1 и 3.2. Результат показан на рисунке 4.2:



Рисунок 4.2 – Аддитивная свёртка

Мультипликативная свёртка выполняется аналогично аддитивной, но необходимо преобразовать исходную матрицу, избавившись от отрицательных или нулевых значений. Решение происходит так же, как и в аддитивной свёртке. Результат показан на рисунке 4.3:



Рисунок 4.3 – Мультипликативная свёртка

Максиминный метод применяется к нормализованной матрице, полученной в методе мультипликативной свёртки. Далее находятся минимальные значения по строкам. Среди полученных значений находится наибольшее значение. Результат показан на рисунке 4.4:



Рисунок 4.4 – Максиминный критерий

Метод главного критерия не требует нормализации исходной матрицы. В каждом столбце выбирается оптимальное значение в зависимости от того, куда стремится критерий: к минимуму или к максимуму. После этого выбираем главный критерий. В данной задаче в качестве главного критерия был выбран критерий C1 (стоимость грузовиков).

Затем, для остальных критериев устанавливаются граничные значения и находится альтернатива, которая удовлетворяет всем ограничениям. Решение показано на рисунке 4.5:

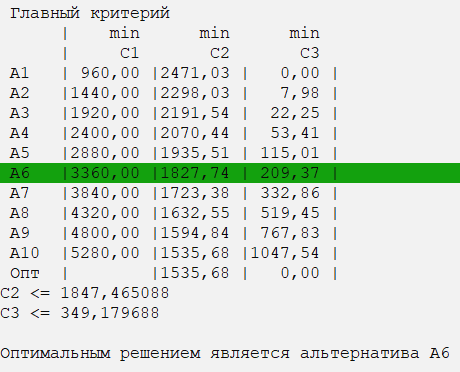


Рисунок 4.5 – Метод главного критерия

Метод уступок является разновидностью метода главного критерия и отличается тем, что лицо, принимающее решение подводится к выбору решения путём постепенного ослабления первоначальных требования (добавления уступок), как правило, одновременно невыполнимых. Решение показано на рисунке 4.6:

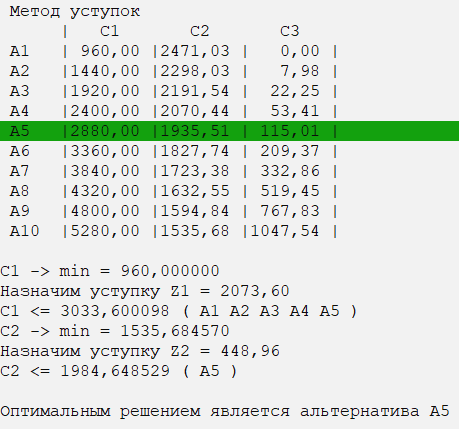


Рисунок 4.6 – Метод уступок

В методе целевого программирования необходимо нормализовать матрицу, в данном случае используются формула 3.4, так как оба критерия стремятся к минимуму. Затем находятся оптимальные значения в каждом столбце. В дополнительном столбце записываем значения, рассчитанные по формуле 4.1:

|  |  |
| --- | --- |
|  | (4.1) |

Решение показано на рисунке 4.7:



Рисунок 4.7 – Метод целевого программирования

1. **ИНСТРУКЦИЯ ПОЛЬЗОВАТЕЛЯ**

Для выполнения программы необходим компьютер с частотой процессора не менее 1,2 ГГц и объемом оперативной памяти 1 Гб. В составе компьютера обязательно должны содержаться: мышь, клавиатура и CD ROM. Запуск программы реализуется с помощью файла«KRabTPR.exe».

При запуске программы решается многокритериальная задача, результаты выводятся на экран и записываются в файл «file.txt», как показано на рисунке 5.1 и 5.2.

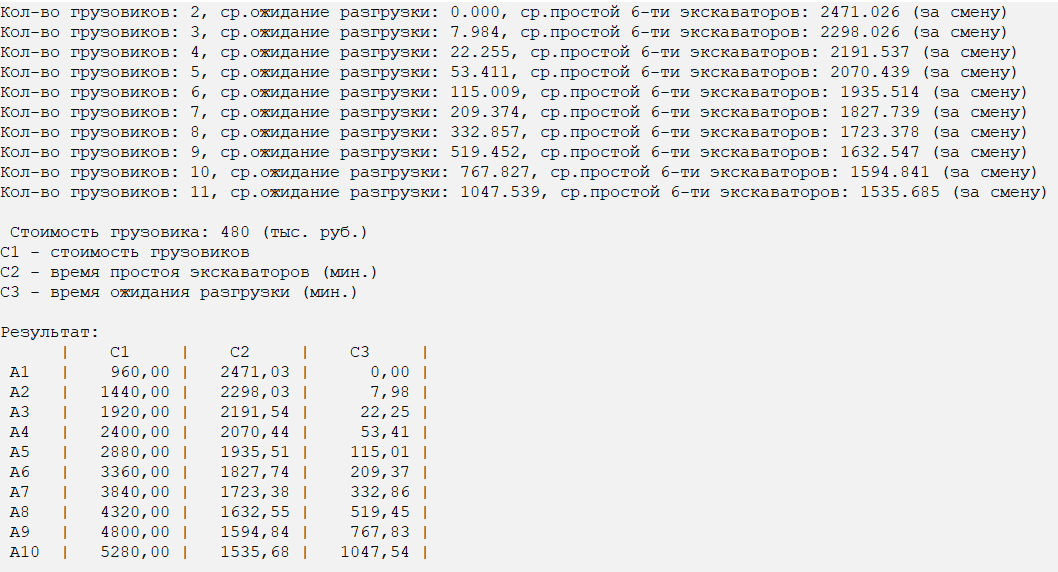


Рисунок 5.1 – Вывод в консоли

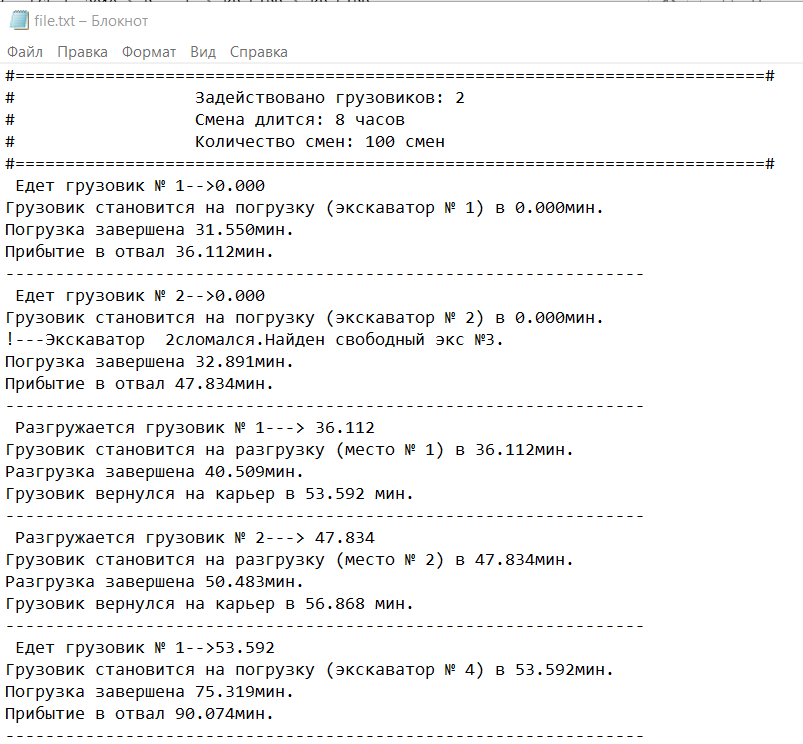


Рисунок 5.2 – Вывод в файле

Вывод всех данных на экран невозможен в связи с их большим объемом. Записав данные в файл, можно проверять работу программы пошагово, без потери информации.

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения данной курсовой работы была смоделирована замкнутая система массового обслуживания, которая реализует вывоз руды грузовиками из карьера в отвал. Были определены время минимальных простоев погрузочных экскаваторов, а также время минимальных простоев разгрузочных мест.

Для определения лучшей альтернативы из полученного в ходе работы моделирующего алгоритма в программном виде были реализованы методы для решения многокритериальных задач: метод главного критерия, метод уступок, метод целевого программирования, аддитивная свёртка, мультипликативная свёртка и максиминный критерий.

Результаты реализованных методов решения многокритериальных задач, полученные после окончания работы реализованной замкнутой системы массового обслуживания проанализированы.

Таким образом, можно сделать вывод о том, что полученные результаты являются прогнозируемыми и адекватными.

**СПИСОК ЛИТЕРАТУРЫ**

1. Агравал, Г.П. Системы автоматического управления: теория, применение, моделирование в MATLAB: Учебное пособие / Г.П. Агравал. - СПб.: Лань, 2013. - 208 c.
2. Афонин, В.В. Моделирование систем: Учебно-практическое пособие / В.В. Афонин. - М.: БИНОМ. ЛЗ, ИНТУИТ, 2012. - 231 c.
3. Косачев, Ю.В. Математическое моделирование интегрированных финансово-промышленных систем / Ю.В. Косачев. - М.: Логос, 2008. - 144 c.
4. Советов, Б.Я. Моделирование систем. практикум: Учебное пособие для бакалавров / Б.Я. Советов, С.А. Яковлев. - Люберцы: Юрайт, 2016. - 295 c.
5. Дорогов, В.Г. Введение в методы и алгоритмы принятия решений: Учебное пособие / В.Г. Дорогов, Я.О. Теплова. - М.: Форум, 2016. - 320 c.
6. Ефимова, М.Р. Методы и алгоритмы принятия решений в экономике: Учебное пособие / М.Р. Ефимова. - М.: Финансы и статистика, 2009. - 224 c.
7. Золотова, Т.В. Методы принятия управленческих решений (для бакалавров) / Т.В. Золотова. - М.: КноРус, 2018. - 381 c.

**ПРИЛОЖЕНИЕ А. ПРОГРАММНЫЙ КОД И РЕЗУЛЬТАТЫ РЕАЛИЗАЦИИ РАБОТЫ СМО**

// KRabTPR.cpp: определяет точку входа для консольного приложения.

//

#include "stdafx.h"

#include "stdafx.h"

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

#include <locale.h>

#include "conio.h"

#include <fstream>

#include <iostream>

#include <windows.h>

//const double eps = 1e-6;

const double PI = 3.141592653589793238463;

#pragma warning(disable : 4996)

using namespace std;

static DWORD a;

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

double NormalRaspr(float matOz, float SKO) {

double y = 0;

float a = (matOz - sqrtf(3) \* SKO) / 10;

float b = (matOz + sqrtf(3) \* SKO) / 10;

for (int i = 0; i < 10; i++) {

y += a + (b - a) \* rand() / RAND\_MAX;

}

return y;

}

double RavnRaspr(double a, double b) {

double f = a + ((rand()) / double(RAND\_MAX)) \* (b - a);//a+(b-a)\*ξ\_i

return f;

}

double ExpRaspr(float matOz) {

float r = (float)rand() / RAND\_MAX;

if (r < 0.00000000001) r = 0.00000000001;

double f = -matOz \* log(r);

return f;

}

double PolomkaGruz(){

//поломка грузовика при движении.

double timeRemont = 0;

double a = (double)rand() / RAND\_MAX;

int polomka = 0; //1 = 1й тип, 2 = 2й тип поломки

if (a < 0.4) { polomka = 1; /\*printf("поломка 1го типа \n");\*/ }

else if (a < 0.6 + 0.4) { polomka = 2; /\*printf("поломка 2го типа \n");\*/ }

if (polomka = 1){

timeRemont = ExpRaspr(10);

}else if (polomka = 2){

timeRemont = ExpRaspr(40);

}

//функция возвращает время ремонта грузовика, в зависимости от типа поломки.

return timeRemont;

}

int minTimeOsv(float\* mas, int n) {

int m = 0;

for (int i = 1; i < n; i++)

if (mas[m] > mas[i])

m = i;

return m;

}

void Faza(float\*\* mas, int n, int m, int\* tekc, int\* f) {

float min = 1000000;

for (int i = 0; i < n; i++) {

for (int j = 0; j < m; j++) {

if (mas[i][j] < min) {

min = mas[i][j];

\*f = i;

\*tekc = j;

}

}

}

}

int maxx(float\* alt, int n) {

int tmpI = 0;

for (int i = 1; i < n; i++)

if (alt[tmpI] < alt[i])

tmpI = i;

return tmpI;

}

int minni(float\* mas, int n) {

int tmpI = 0;

for (int i = 1; i < n; i++)

if (mas[tmpI] > mas[i])

tmpI = i;

return tmpI;

}

float\*\* Normaliz12(float\*\* alt, int nA, int nC, bool\* up) {

float\*\* norm = new float\* [nC];

for (int i = 0; i < nC; i++)

norm[i] = new float[nA]();

for (int i = 0; i < nC; i++) {

float maxA = alt[i][maxx(alt[i], nA)];

float minA = alt[i][minni(alt[i], nA)];

if (up[i])

for (int j = 0; j < nA; j++)

norm[i][j] = (alt[i][j] - minA) / (maxA - minA);

else

for (int j = 0; j < nA; j++)

norm[i][j] = (maxA - alt[i][j]) / (maxA - minA);

}

return norm;

}

float\*\* Normaliz34(float\*\* alt, int nA, int nC, bool\* up) {

float\*\* norm = new float\* [nC];

for (int i = 0; i < nC; i++)

norm[i] = new float[nA]();

for (int i = 0; i < nC; i++) {

if (up[i]) {

float maxA = alt[i][maxx(alt[i], nA)];

float minA = alt[i][minni(alt[i], nA)];

float delta = minA <= 0 ? abs(minA) + 1 : 0;

for (int j = 0; j < nA; j++)

norm[i][j] = (alt[i][j] + delta) / (maxA + delta);

}

else {

float minA = alt[i][minni(alt[i], nA)];

float delta = minA <= 0 ? abs(minA) + 1 : 0;

for (int j = 0; j < nA; j++) {

norm[i][j] = (minA + delta) / (alt[i][j] + delta);

}

}

}

return norm;

}

short getY() {

CONSOLE\_SCREEN\_BUFFER\_INFO csbi = { 0 };

GetConsoleScreenBufferInfo(hConsole, &csbi);

return csbi.dwCursorPosition.Y;

}

void MultiSvertka(float\*\* norm, int nA, int nC, float\* prior) {

short Y = getY();

float maxim = 0;

int maxI = 0;

printf(" Мультипликативная свертка\n |");

for (int i = 0; i < nC; i++) {

printf(" C%-2i |", i + 1);

}

printf(" sumA\n");

for (int i = 0; i < nA; i++) {

printf("A%-2i |", i + 1);

float sum = 1;

for (int j = 0; j < nC; j++) {

printf("% 3.2f |", norm[j][i]);

sum \*= pow(norm[j][i], prior[j]);

}

printf(" %3.2f\n", sum);

if (sum > maxim) {

maxim = sum;

maxI = i;

}

}

COORD invalid = { 0,(short)(maxI + 2 + Y) };

FillConsoleOutputAttribute(hConsole, BACKGROUND\_GREEN , 39, invalid , &a);

printf("Alfa |");

for (int j = 0; j < nC; j++)

printf(" %3.2f |", prior[j]);

printf("\n\nОптимальным решением является альтернатива А%i\n\n", maxI + 1);

}

void AdditivSvertka(float\*\* norm, int nA, int nC, float\* prior) {

short Y = getY();

float maxim = 0;

int maxI = 0;

printf(" Аддитивная свертка\n |");

for (int i = 0; i < nC; i++) {

printf(" C%-2i |", i + 1);

}

printf(" sumA\n");

for (int i = 0; i < nA; i++) {

printf("A%-2i |", i + 1);

float sum = 0;

for (int j = 0; j < nC; j++) {

printf("% 3.2f |", norm[j][i]);

sum += norm[j][i] \* prior[j];

}

printf(" %3.2f\n", sum);

if (sum > maxim) {

maxim = sum;

maxI = i;

}

}

COORD invalid = { 0, (short)(maxI + 2 + Y) };

FillConsoleOutputAttribute(hConsole, BACKGROUND\_GREEN , 39, invalid, &a);

printf("Alfa |");

for (int j = 0; j < nC; j++)

printf(" %3.2f |", prior[j]);

printf("\n\nОптимальным решением является альтернатива А%i\n\n", maxI + 1);

}

void MaxiMin(float\*\* norm, int nA, int nC) {

short Y = getY();

float maxim = 0;

int maxI = 0;

printf(" Максиминный метод\n |");

for (int i = 0; i < nC; i++) {

printf(" C%-2i |", i + 1);

}

printf(" min\n");

for (int i = 0; i < nA; i++) {

printf("A%-2i |", i + 1);

int minJ = 0;

for (int j = 0; j < nC; j++) {

printf("%3.2f |", norm[j][i]);

if (norm[minJ][i] > norm[j][i])

minJ = j;

}

COORD invalid = { (short)(6 + minJ \* 6),(short)(i + 2 + Y) };

FillConsoleOutputAttribute(hConsole, BACKGROUND\_GREEN , 5, invalid, &a);

printf("%3.2f\n", norm[minJ][i]);

if (maxim < norm[minJ][i]) {

maxim = norm[minJ][i];

maxI = i;

}

}

COORD invalid = { 0,(short)(maxI + 2 + Y) };

FillConsoleOutputAttribute(hConsole, BACKGROUND\_INTENSITY, 6 + nC \* 7, invalid , &a);

printf("\n\nОптимальным решением является альтернатива А%i\n\n", maxI + 1);

}

void CelProg(float\*\* norm, int nA, int nC, float p) {

short Y = getY();

float minim = 10;

int minI = 0;

float\* f = new float[nC]();

for (int i = 0; i < nC; i++) {

f[i] = norm[i][maxx(norm[i], nA)];

}

printf(" Целевое программирование\n |");

for (int i = 0; i < nC; i++) {

printf(" C%-2i |", i + 1);

}

printf(" sumA\n");

for (int i = 0; i < nA; i++) {

printf("A%-2i |", i + 1);

float sum = 0;

for (int j = 0; j < nC; j++) {

printf("% 3.2f |", norm[j][i]);

sum += pow(abs(norm[j][i] - f[j]), p);

}

sum = powf(sum, 1.0 / p);

printf(" %3.2f\n", sum);

if (sum < minim) {

minim = sum;

minI = i;

}

}

COORD invalid ={0,(short)(minI + 2 + Y)};

FillConsoleOutputAttribute(hConsole, BACKGROUND\_GREEN , 39, invalid, &a);

printf("\n\nОптимальным решением является альтернатива А%i\n\n", minI + 1);

delete[]f;

}

void GlavCrit(float\*\* alt, int nA, int nC, bool\* up, int mainC) {

short Y = getY();

printf(" Главный критерий\n |");

for (int i = 0; i < nC; i++)

up[i] ? printf(" max ") : printf(" min ");

printf("\n |");

for (int i = 0; i < nC; i++) {

printf(" C%i ", i + 1);

}

for (int i = 0; i < nA; i++) {

printf("\n A%-2i |", i + 1);

for (int j = 0; j < nC; j++)

printf("%7.2f |", alt[j][i]);

}

printf("\n Опт |");

float\* C = new float[nC - 1]();

for (int i = 0, j = 0; i < nC; i++) {

if (i != mainC - 1) {

float ma = alt[i][maxx(alt[i], nA)];

float mi = alt[i][minni(alt[i], nA)];

if (up[i]) {

printf("%7.2f |", ma);

C[j] = ma - (ma - mi) / 3;

}

else {

printf("%7.2f |", mi);

C[j] = mi + (ma - mi) / 3;

}

j++;

}

else printf(" |");

}

int res = -1;

while (res == -1) {

float glCr = up[mainC - 1] ? -999999999 : 999999999;

for (int i = 0; i < nA; i++) {

bool correct = true;

for (int j = 0, h = 0; j < nC && correct; j++) {

if (j != mainC - 1) {

if ((up[j] && C[h] > alt[j][i]) || (!up[j] && C[h] < alt[j][i]))

correct = false;

h++;

}

}

if (correct && ((up[mainC - 1] && glCr < alt[mainC - 1][i]) || (!up[mainC - 1] && glCr > alt[mainC - 1][i]))) {

glCr = alt[mainC - 1][i];

res = i;

}

}

for (int i = 0, j = 0; i < nC; i++) {

if (i != mainC - 1) {

if (up[i]) {

printf("\nC%i >= %f", i + 1, C[j]);

float mi = alt[i][minni(alt[i], nA)];

C[j] -= (C[j] - mi) / 3;

}

else {

printf("\nC%i <= %f", i + 1, C[j]);

float ma = alt[i][maxx(alt[i], nA)];

C[j] += (ma - C[j]) / 3;

}

j++;

}

}

printf("\n");

}

COORD invalid = {0,(short)(res + 3 + Y)};

FillConsoleOutputAttribute(hConsole, BACKGROUND\_GREEN , 43, invalid, &a);

printf("\nОптимальным решением является альтернатива А%i\n\n", res + 1);

}

void Ustupok(float\*\* alt, int nA, int nC, float\* prior, bool\* up) {

short Y = getY();

float\* tmp = new float[nC];

int\* im = new int[nC];

printf(" Метод уступок\n |");

for (int i = 0; i < nC; i++) {

printf(" C%-2i ", i + 1);

}

for (int i = 0; i < nA; i++) {

printf("\n A%-2i |", i + 1);

for (int j = 0; j < nC; j++)

printf("%7.2f |", alt[j][i]);

}

printf("\n\n");

for (int i = 0; i < nC; i++)

tmp[i] = prior[i];

for (int i = 0; i < nC; i++) {

int k = maxx(tmp, nC);

im[i] = k;

tmp[k] = -1;

}

delete[] tmp;

bool\* res = new bool[nA];

for (int i = 0; i < nA; i++)res[i] = true;

int r, otk;

for (int i = 0; i < nC; i++) {

int minI = minni(alt[im[i]], nA);

int maxI = maxx(alt[im[i]], nA);

if (up[im[i]])

printf("C%i -> max = %f\n", im[i] + 1, alt[im[i]][maxI]);

else

printf("C%i -> min = %f\n", im[i] + 1, alt[im[i]][minI]);

float ustupka = (alt[im[i]][maxI] - alt[im[i]][minI]) / 2.5;

do {

ustupka \*= 1.2;

printf("Назначим уступку Z%i = %.2f\n", i + 1, ustupka);

if (up[im[i]])

printf("C%i >= %f ( ", im[i] + 1, alt[im[i]][maxI] - ustupka);

else

printf("C%i <= %f ( ", im[i] + 1, alt[im[i]][minI] + ustupka);

otk = 0;

for (int j = 0; j < nA; j++) {

if (up[im[i]]) {

if (alt[im[i]][j] >= alt[im[i]][maxI] - ustupka && res[j]) {

printf("A%i ", j + 1);

r = j;

}

else otk++;

}

else {

if (alt[im[i]][j] <= alt[im[i]][minI] + ustupka && res[j]) {

printf("A%i ", j + 1);

r = j;

}

else otk++;

}

}

printf(")\n");

} while (otk == nA);

if (otk == nA - 1) {

printf("\nОптимальным решением является альтернатива А%i\n\n", r + 1);

COORD invalid = { 0,(short)(r + 2 + Y) };

FillConsoleOutputAttribute(hConsole, BACKGROUND\_GREEN , 43, invalid, &a);

break;

}

for (int j = 0; j < nA; j++) {

if (up[im[i]]) {

if (alt[im[i]][j] < alt[im[i]][maxI] - ustupka) res[j] = false;

}

else {

if (alt[im[i]][j] > alt[im[i]][minI] + ustupka) res[j] = false;

}

}

}

if (nA - otk >= 2) {

int zz = r;

for (int i = 0; i < nA; i++) {

if (up[im[0]]) {

if (res[i] && alt[im[0]][zz] < alt[im[0]][i]) zz = i;

}

else

if (res[i] && alt[im[0]][zz] > alt[im[0]][i]) zz = i;

}

printf("\nОптимальным решением является альтернатива А%i\n\n", zz + 1);

COORD invalid = { 0,(short)(zz + 2 + Y) };

FillConsoleOutputAttribute(hConsole, BACKGROUND\_GREEN , 43, invalid, &a);

}

delete[]res;

delete[]im;

}

void PrintResults(float\*\* alt, int nA, int nC) {

for (int i = 0; i < nC; i++) {

printf(" C%-4i |", i + 1);

}

for (int i = 0; i < nA; i++) {

printf(" \n A%-2i |", i + 1);

for (int j = 0; j < nC; j++)

printf("%10.2f |", alt[j][i]);

}

printf("\n\n");

}

void Obrabotka(float\*\* alt, int nA, int nC, bool\* up, float\* prior) {

printf("C1 - стоимость грузовиков\n");

printf("C2 - время простоя экскаваторов (мин.)\n");

printf("C3 - время ожидания разгрузки (мин.)\n");

printf("\nРезультат:\n |");

PrintResults(alt, nA, nC);

float\*\* norm12 = Normaliz12(alt, nA, nC, up);

float\*\* norm34 = Normaliz34(alt, nA, nC, up);

printf("Нормализация (1,2):\n |");

PrintResults(norm12, nA, nC);

printf("Нормализация (3,4):\n |");

PrintResults(norm34, nA, nC);

AdditivSvertka(norm12, nA, nC, prior);

MultiSvertka(norm34, nA, nC, prior);

MaxiMin(norm34, nA, nC);

CelProg(norm34, nA, nC, 2);

GlavCrit(alt, nA, nC, up, 1);

Ustupok(alt, nA, nC, prior, up);

for (int i = 0; i < nC; i++) {

delete[]norm12[i];

delete[]norm34[i];

}

delete[] norm12;

delete[] norm34;

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

setlocale(LC\_ALL, "Rus");

fstream file;

file.open("file.txt", ios::out);

streambuf\* stream\_buffer\_cout = cout.rdbuf();

streambuf\* stream\_buffer\_file = file.rdbuf();

cout.rdbuf(stream\_buffer\_file);

cout.precision(3);

cout.setf(ios::fixed);

const int COUNT\_GRUZ = 12;//до скольки грузовиков будем моделировать?

const int PRICE = 480; //цена одного грузовика

const int KOL\_SMEN = 100;

const int SMENA\_MINUTES = 480; // 480 - это количество минут в 8-ми часовой смене \* 100 <- моделируем для 100 смен

const float INFIN = 900000000;

int f=0, fl=0, nGruz, tekGruz = 0, tekRazgr = 0, tekExav = 0, rezervExav = -1;

float prostoiEx = 0, ozidanieRazgr = 0, razgr=0;

float pogr\_osv[6] = {}, razgr\_osv[2] = {};

float ozidRazgr[COUNT\_GRUZ] = { 90000000 }, prEx[COUNT\_GRUZ] = { 9000000 };

double verPolGruz = 0, verPolExkav = 0;

//цикл по количеству грузовых машин

for( nGruz = 2; nGruz < COUNT\_GRUZ; nGruz++ ){

f = 0;//фаза движения ( 0- погрузка+движение туда, 1 - разгрузка+движение назад )

float\*\* trucks = new float\* [2]; // массив на две строки: 1 - приезд на погрузку, 2 - приезд на разгрузку

// цикл по количеству поездок

for (int s = 0; s < 2; s++) {

trucks[s] = new float[nGruz];

}

for (int s = 0; s < nGruz; s++) {

trucks[1][s] = INFIN;

trucks[0][s] = 0;

}

tekExav = 0;//экскаватор

ozidanieRazgr = 0;//ожидание разгрузки

prostoiEx = 0; //простой экскаваторов

for(int i=0; i<6; i++)

pogr\_osv[i] = 0; //время освобождения мест на погрузку

for(int i=0; i<2; i++)

razgr\_osv[i] = 0; //время освобождения мест на разгрузку

cout << "#===========================================================================#\n";

cout << "# Задействовано грузовиков: " << nGruz << " \n";

cout << "# Смена длится: " << SMENA\_MINUTES/60 << " часов \n";

cout << "# Количество смен: " << KOL\_SMEN << " смен \n";

cout << "#===========================================================================#\n";

fl = 0;//конец смены 8 часов (480 минут)

do {

// выбираем фазу путем поиска минимального числа в массиве грузовиков

Faza(trucks, 2, nGruz, &tekGruz, &f); // нашли минимальный элемент, номер строки - фаза, номер столбца - номер грузовика

if (f == 0) {//если фаза 1я

cout << " Едет грузовик № " << tekGruz + 1 << "-->" << trucks[f][tekGruz]<< "";

cout << "\n";

tekExav = minTimeOsv(pogr\_osv, 6);//выбираем место погрузки (экскаватор)

rezervExav = -1;//резервный экскаватор не задан

// если место занято, то машина обслуживается с его освобождения

if (pogr\_osv[tekExav] > trucks[f][tekGruz]) {

trucks[f][tekGruz] = pogr\_osv[tekExav];

cout << "Экскаватор занят, грузовик загрузится с: " << trucks[f][tekGruz] << "\n";

}

// конец моделирования? стоп. нет? работаем дальше

if (trucks[f][tekGruz] > SMENA\_MINUTES\*KOL\_SMEN) { //кол-во минут в смене\*кол-во смен

fl = 1;

cout << "#============================ Смена завершена ==============================#\n\n";

break;

}

//если место было свободно, а машины не было долго, то увеличиваем простой экскаваторов

if (trucks[f][tekGruz] > pogr\_osv[tekExav] && pogr\_osv[tekExav] > 0)

prostoiEx += trucks[f][tekGruz] - pogr\_osv[tekExav];

cout << "Грузовик становится на погрузку (экскаватор № " << tekExav + 1 << ") в " << trucks[f][tekGruz] << "мин.\n";

if (tekExav < 3){//если экскаватор первого типа

trucks[f][tekGruz] += NormalRaspr(30, 10);// погрузка

}

else if (tekExav >= 3){//если экскаватор второго типа

trucks[f][tekGruz] += NormalRaspr(20, 5);// погрузка

}

//во время погрузки с вероятностью 0.1 экскаватор может выходить из строя

verPolExkav = (double)rand() / RAND\_MAX;

if(verPolExkav <= 0.1){//если поломался

float RavnRasp = RavnRaspr(10,30);//сформируем время по равномерному закону

for(int i=0; i<6; i++){//ищем свободный

if(pogr\_osv[i] < trucks[f][tekGruz])//если есть экскаватор свободный

{

rezervExav = i; //то он становится резервным

if(tekExav != rezervExav)//но исключаем то, что это тотже самый экскаватор

break;

}

}

if( rezervExav != -1 ){//есть свободные экскаваторы?

//замена экскаватора , время увеличивается на треть

if (rezervExav < 3){//если экскаватор первого типа

trucks[f][tekGruz] += NormalRaspr(30, 10)/3;// погрузка/3

}

else if (rezervExav >= 3){//если экскаватор второго типа

trucks[f][tekGruz] += NormalRaspr(20, 5)/3;// погрузка/3

}

cout << "!---Экскаватор " << tekExav+1 << "сломался." << "Найден свободный экс №" << rezervExav+1 << ".\n";

}

else{//нет свободных экскаваторов

trucks[f][tekGruz] += RavnRasp;//то + ремонт

}

// место погрузки резервного снова свободно с нового времени

pogr\_osv[rezervExav] = trucks[f][tekGruz];

//а поломанный сделают и освободят

pogr\_osv[tekExav] = trucks[f][tekGruz] + RavnRasp;

}

else {//если не ломался

// место погрузки снова свободно с нового времени

pogr\_osv[tekExav] = trucks[f][tekGruz];

}

cout << "Погрузка завершена " << trucks[f][tekGruz] << "мин.\n";

// движение (туда)

trucks[f][tekGruz] += ExpRaspr(30);

verPolGruz = (double)rand() / RAND\_MAX;

if(verPolGruz <= 0.1) { //если поломался

trucks[f][tekGruz] += PolomkaGruz(); //то + ремонт

}

cout << "Прибытие в отвал " << trucks[f][tekGruz] << "мин.\n";

trucks[1][tekGruz] = trucks[f][tekGruz];

trucks[f][tekGruz] = INFIN;

cout << "---------------------------------------------------------------- \n";

}

else if(f == 1) {//если фаза 2я

cout << " Разгружается грузовик № " << tekGruz + 1 << "---> " << trucks[f][tekGruz] << "\n";

//выбор места с мин. временем освобождения

tekRazgr = minTimeOsv(razgr\_osv, 2);//выбираем место разгрузки

// если место занято, то машина обслуживается с его освобождения

if (razgr\_osv[tekRazgr] > trucks[f][tekGruz]) {

ozidanieRazgr += razgr\_osv[tekRazgr] - trucks[f][tekGruz];//считаю ожидание

trucks[f][tekGruz] = razgr\_osv[tekRazgr];//грузовик разгрузится с нового времени

cout << "Место " << (tekRazgr+1) <<" занято, грузовик разгрузится с: " << trucks[f][tekGruz] << "\n";

}

cout << "Грузовик становится на разгрузку (место № " << tekRazgr + 1 << ") в " << trucks[f][tekGruz] << "мин.\n";

trucks[f][tekGruz] += ExpRaspr(20); // разгрузка

cout << "Разгрузка завершена " << trucks[f][tekGruz] << "мин.\n";

// место погрузки снова свободно с нового времени

razgr\_osv[tekRazgr] = trucks[f][tekGruz];

// машина сразу уезжает назад на погрузку

trucks[f][tekGruz] += ExpRaspr(20); // движение

verPolGruz = (double)rand() / RAND\_MAX;

if(verPolGruz <= 0.05) { //если поломался

trucks[f][tekGruz] += PolomkaGruz(); // то + ремонт

}

cout << "Грузовик вернулся на карьер в " << trucks[f][tekGruz] << " мин.\n";

trucks[0][tekGruz] = trucks[f][tekGruz];

trucks[f][tekGruz] = INFIN;

cout << "---------------------------------------------------------------- \n";

}

}while(fl == 0);//пока не конец смены - все работают

ozidRazgr[nGruz - 1] = ozidanieRazgr;//ожидание разгрузки

prEx[nGruz - 1] = prostoiEx;//простой экскаваторов

delete[] trucks;//очистка памяти

}

cout.rdbuf(stream\_buffer\_cout);

for (int d = 1; d < COUNT\_GRUZ - 1; d++) {

cout << "Кол-во грузовиков: " << d + 1 << ", ср.ожидание разгрузки: " << ozidRazgr[d]/KOL\_SMEN << ", ср.простой 6-ти экскаваторов: " << prEx[d]/KOL\_SMEN << " (за смену)\n";

}

//массив результаты (для обработки методами ТПР)

float\*\* rezultat = new float\* [3];

for (int i = 0; i < 3; i++)

rezultat[i] = new float[COUNT\_GRUZ - 1];

for (int j = 0; j < COUNT\_GRUZ - 2; j++) {

rezultat[0][j] = (j + 2) \* PRICE;//кол-во автомобилей

rezultat[1][j] = prEx[j + 1]/KOL\_SMEN;//время простоя экскаваторов

rezultat[2][j] = ozidRazgr[j + 1]/KOL\_SMEN;//время ожидания разгрузки на местах

}

bool v[3] = { false, false, false };

float prior[3] = { 0.6, 0.3, 0.1 };

cout << "\n Стоимость грузовика: " << PRICE <<" (тыс. руб.)" <<"\n";

Obrabotka(rezultat, COUNT\_GRUZ - 2, 3, v, prior);

file.close();

for (int i = 0; i < 3; i++) {

delete[] rezultat[i];

}

\_getch();

return 0;

}

Подробный результат работы на примере 2-х грузовых машин:

#===========================================================================#

# Задействовано грузовиков: 2

# Смена длится: 8 часов

# Количество смен: 100 смен

#===========================================================================#

Едет грузовик № 1-->0.000

Грузовик становится на погрузку (экскаватор № 1) в 0.000мин.

Погрузка завершена 31.550мин.

Прибытие в отвал 36.112мин.

----------------------------------------------------------------

Едет грузовик № 2-->0.000

Грузовик становится на погрузку (экскаватор № 2) в 0.000мин.

!---Экскаватор 2сломался.Найден свободный экс №3.

Погрузка завершена 32.891мин.

Прибытие в отвал 47.834мин.

----------------------------------------------------------------

Разгружается грузовик № 1---> 36.112

Грузовик становится на разгрузку (место № 1) в 36.112мин.

Разгрузка завершена 40.509мин.

Грузовик вернулся на карьер в 53.592 мин.

----------------------------------------------------------------

Разгружается грузовик № 2---> 47.834

Грузовик становится на разгрузку (место № 2) в 47.834мин.

Разгрузка завершена 50.483мин.

Грузовик вернулся на карьер в 56.868 мин.

----------------------------------------------------------------

Едет грузовик № 1-->53.592

Грузовик становится на погрузку (экскаватор № 4) в 53.592мин.

Погрузка завершена 75.319мин.

Прибытие в отвал 90.074мин.

----------------------------------------------------------------

Едет грузовик № 2-->56.868

Грузовик становится на погрузку (экскаватор № 5) в 56.868мин.

Погрузка завершена 74.368мин.

Прибытие в отвал 113.328мин.

----------------------------------------------------------------

Разгружается грузовик № 1---> 90.074

Грузовик становится на разгрузку (место № 1) в 90.074мин.

Разгрузка завершена 97.461мин.

Грузовик вернулся на карьер в 101.005 мин.

----------------------------------------------------------------

Едет грузовик № 1-->101.005

Грузовик становится на погрузку (экскаватор № 6) в 101.005мин.

Погрузка завершена 121.259мин.

Прибытие в отвал 136.139мин.

----------------------------------------------------------------

Разгружается грузовик № 2---> 113.328

Грузовик становится на разгрузку (место № 2) в 113.328мин.

Разгрузка завершена 133.687мин.

Грузовик вернулся на карьер в 171.423 мин.

----------------------------------------------------------------

Разгружается грузовик № 1---> 136.139

Грузовик становится на разгрузку (место № 1) в 136.139мин.

Разгрузка завершена 153.245мин.

Грузовик вернулся на карьер в 157.636 мин.

----------------------------------------------------------------

Едет грузовик № 1-->157.636

Грузовик становится на погрузку (экскаватор № 1) в 157.636мин.

Погрузка завершена 187.016мин.

Прибытие в отвал 245.663мин.

----------------------------------------------------------------

Едет грузовик № 2-->171.423

Грузовик становится на погрузку (экскаватор № 3) в 171.423мин.

Погрузка завершена 203.422мин.

Прибытие в отвал 255.219мин.

----------------------------------------------------------------

Разгружается грузовик № 1---> 245.663

Грузовик становится на разгрузку (место № 2) в 245.663мин.

Разгрузка завершена 260.541мин.

Грузовик вернулся на карьер в 297.756 мин.

----------------------------------------------------------------

Разгружается грузовик № 2---> 255.219

Грузовик становится на разгрузку (место № 1) в 255.219мин.

Разгрузка завершена 261.458мин.

Грузовик вернулся на карьер в 279.506 мин.

----------------------------------------------------------------

Едет грузовик № 2-->279.506

Грузовик становится на погрузку (экскаватор № 2) в 279.506мин.

Погрузка завершена 309.789мин.

Прибытие в отвал 335.373мин.

----------------------------------------------------------------

Едет грузовик № 1-->297.756

Грузовик становится на погрузку (экскаватор № 5) в 297.756мин.

Погрузка завершена 318.631мин.

Прибытие в отвал 354.971мин.

----------------------------------------------------------------

Разгружается грузовик № 2---> 335.373

Грузовик становится на разгрузку (место № 2) в 335.373мин.

Разгрузка завершена 346.711мин.

Грузовик вернулся на карьер в 379.305 мин.

----------------------------------------------------------------

Разгружается грузовик № 1---> 354.971

Грузовик становится на разгрузку (место № 1) в 354.971мин.

Разгрузка завершена 358.471мин.

Грузовик вернулся на карьер в 376.914 мин.

----------------------------------------------------------------

Едет грузовик № 1-->376.914

Грузовик становится на погрузку (экскаватор № 4) в 376.914мин.

Погрузка завершена 397.587мин.

Прибытие в отвал 402.740мин.

----------------------------------------------------------------

Едет грузовик № 2-->379.305

Грузовик становится на погрузку (экскаватор № 6) в 379.305мин.

Погрузка завершена 399.559мин.

Прибытие в отвал 408.301мин.

----------------------------------------------------------------

Разгружается грузовик № 1---> 402.740

Грузовик становится на разгрузку (место № 2) в 402.740мин.

Разгрузка завершена 441.433мин.

Грузовик вернулся на карьер в 569.971 мин.

----------------------------------------------------------------

Разгружается грузовик № 2---> 408.301

Грузовик становится на разгрузку (место № 1) в 408.301мин.

Разгрузка завершена 412.608мин.

Грузовик вернулся на карьер в 415.797 мин.

----------------------------------------------------------------

Едет грузовик № 2-->415.797

Грузовик становится на погрузку (экскаватор № 1) в 415.797мин.

Погрузка завершена 441.445мин.

Прибытие в отвал 452.308мин.

----------------------------------------------------------------

Разгружается грузовик № 2---> 452.308

Грузовик становится на разгрузку (место № 1) в 452.308мин.

Разгрузка завершена 468.879мин.

Грузовик вернулся на карьер в 498.062 мин.

----------------------------------------------------------------

Едет грузовик № 2-->498.062

Грузовик становится на погрузку (экскаватор № 3) в 498.062мин.

Погрузка завершена 528.455мин.

Прибытие в отвал 551.918мин.

----------------------------------------------------------------

Разгружается грузовик № 2---> 551.918

Грузовик становится на разгрузку (место № 2) в 551.918мин.

Разгрузка завершена 598.475мин.

Грузовик вернулся на карьер в 607.870 мин.

----------------------------------------------------------------

Едет грузовик № 1-->569.971

Грузовик становится на погрузку (экскаватор № 2) в 569.971мин.

Погрузка завершена 599.415мин.

Прибытие в отвал 623.263мин.

----------------------------------------------------------------

Едет грузовик № 2-->607.870

Грузовик становится на погрузку (экскаватор № 5) в 607.870мин.

Погрузка завершена 629.818мин.

Прибытие в отвал 632.810мин.

----------------------------------------------------------------

Разгружается грузовик № 1---> 623.263

Грузовик становится на разгрузку (место № 1) в 623.263мин.

Разгрузка завершена 656.589мин.

Грузовик вернулся на карьер в 666.649 мин.

----------------------------------------------------------------

Разгружается грузовик № 2---> 632.810

Грузовик становится на разгрузку (место № 2) в 632.810мин.

Разгрузка завершена 643.546мин.

Грузовик вернулся на карьер в 664.468 мин.

----------------------------------------------------------------

Едет грузовик № 2-->664.468

Грузовик становится на погрузку (экскаватор № 4) в 664.468мин.

Погрузка завершена 686.293мин.

Прибытие в отвал 696.067мин.

----------------------------------------------------------------

Едет грузовик № 1-->666.649

Грузовик становится на погрузку (экскаватор № 6) в 666.649мин.

!---Экскаватор 6сломался.Найден свободный экс №1.

Погрузка завершена 695.186мин.

Прибытие в отвал 704.300мин.

----------------------------------------------------------------

Разгружается грузовик № 2---> 696.067

Грузовик становится на разгрузку (место № 2) в 696.067мин.

Разгрузка завершена 699.874мин.

Грузовик вернулся на карьер в 700.710 мин.

----------------------------------------------------------------

Едет грузовик № 2-->700.710

Грузовик становится на погрузку (экскаватор № 3) в 700.710мин.

Погрузка завершена 731.089мин.

Прибытие в отвал 743.835мин.

----------------------------------------------------------------

Разгружается грузовик № 1---> 704.300

Грузовик становится на разгрузку (место № 1) в 704.300мин.

Разгрузка завершена 749.414мин.

Грузовик вернулся на карьер в 763.076 мин.

----------------------------------------------------------------

Разгружается грузовик № 2---> 743.835

Грузовик становится на разгрузку (место № 2) в 743.835мин.

Разгрузка завершена 768.573мин.

Грузовик вернулся на карьер в 770.241 мин.

----------------------------------------------------------------

Едет грузовик № 1-->763.076

Грузовик становится на погрузку (экскаватор № 2) в 763.076мин.

Погрузка завершена 793.253мин.

Прибытие в отвал 844.069мин.

----------------------------------------------------------------

Едет грузовик № 2-->770.241

Грузовик становится на погрузку (экскаватор № 5) в 770.241мин.

Погрузка завершена 792.411мин.

Прибытие в отвал 844.431мин.

----------------------------------------------------------------

Разгружается грузовик № 1---> 844.069

Грузовик становится на разгрузку (место № 1) в 844.069мин.

Разгрузка завершена 884.625мин.

Грузовик вернулся на карьер в 890.559 мин.

----------------------------------------------------------------

Разгружается грузовик № 2---> 844.431

Грузовик становится на разгрузку (место № 2) в 844.431мин.

Разгрузка завершена 854.251мин.

Грузовик вернулся на карьер в 925.882 мин.

----------------------------------------------------------------

Едет грузовик № 1-->890.559

Грузовик становится на погрузку (экскаватор № 4) в 890.559мин.

Погрузка завершена 910.302мин.

Прибытие в отвал 924.782мин.

----------------------------------------------------------------

Разгружается грузовик № 1---> 924.782

Грузовик становится на разгрузку (место № 2) в 924.782мин.

Разгрузка завершена 979.159мин.

Грузовик вернулся на карьер в 1014.725 мин.

----------------------------------------------------------------

Едет грузовик № 2-->925.882

Грузовик становится на погрузку (экскаватор № 1) в 925.882мин.

Погрузка завершена 959.813мин.

Прибытие в отвал 973.665мин.

----------------------------------------------------------------

Разгружается грузовик № 2---> 973.665

Грузовик становится на разгрузку (место № 1) в 973.665мин.

Разгрузка завершена 1001.097мин.

Грузовик вернулся на карьер в 1040.920 мин.

----------------------------------------------------------------

Едет грузовик № 1-->1014.725

Грузовик становится на погрузку (экскаватор № 6) в 1014.725мин.

Погрузка завершена 1032.141мин.

Прибытие в отвал 1039.284мин.

----------------------------------------------------------------

Разгружается грузовик № 1---> 1039.284

Грузовик становится на разгрузку (место № 2) в 1039.284мин.

Разгрузка завершена 1044.008мин.

Грузовик вернулся на карьер в 1049.928 мин.

----------------------------------------------------------------

Едет грузовик № 2-->1040.920

Грузовик становится на погрузку (экскаватор № 3) в 1040.920мин.

Погрузка завершена 1069.019мин.

Прибытие в отвал 1095.711мин.

----------------------------------------------------------------

Едет грузовик № 1-->1049.928

Грузовик становится на погрузку (экскаватор № 5) в 1049.928мин.

Погрузка завершена 1066.934мин.

Прибытие в отвал 1068.594мин.

----------------------------------------------------------------

Разгружается грузовик № 1---> 1068.594

Грузовик становится на разгрузку (место № 1) в 1068.594мин.

Разгрузка завершена 1081.133мин.

Грузовик вернулся на карьер в 1084.570 мин.

----------------------------------------------------------------

Едет грузовик № 1-->1084.570

Грузовик становится на погрузку (экскаватор № 2) в 1084.570мин.

Погрузка завершена 1112.167мин.

Прибытие в отвал 1137.104мин.

**………………………………**

Едет грузовик № 1-->47983.055

Грузовик становится на погрузку (экскаватор № 4) в 47983.055мин.

Погрузка завершена 48004.211мин.

Прибытие в отвал 48108.270мин.

----------------------------------------------------------------

Разгружается грузовик № 2---> 48015.355

Грузовик становится на разгрузку (место № 2) в 48015.355мин.

Разгрузка завершена 48016.941мин.

Грузовик вернулся на карьер в 48022.500 мин.

----------------------------------------------------------------

#============================ Смена завершена ==============================#