

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ Федеральное государственное автономное образовательное
учреждение высшего образования
«Национальный исследовательский
Нижегородский государственный университет им. Н.И. Лобачевского»
(ННГУ)

Институт информационных технологий, математики и механики

Кафедра: Алгебры, геометрии и дискретной математики

Направление подготовки: «Прикладная математика и информатика»
Профиль подготовки: «Прикладная математика и информатика (общий
профиль)»

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
БАКАЛАВРА**

Тема:
**«Минимизация неявно заданной функции, применение метода
имитации отжига»**

Выполнил:
студент группы 381903_3
Розанов Дмитрий Игоревич

подпись

Научный руководитель:
доцент, кандидат физико-
математических наук
Чирков Александр Юрьевич

подпись

Нижний Новгород
2023

Введение:

Задачи оптимизации представляют собой один из самых важных и распространенных классов задач в практическом применении. Задачи оптимизации возникают в различных сферах жизни и деятельности, где необходимо найти оптимальное решение для достижения определенной цели. Например, задачи оптимизации возникают:

- в экономике и бизнесе при планировании производства, оптимизации затрат и максимизации прибыли;
- в инженерии и технологии при проектировании и оптимизации процессов, создании новых материалов и технологий;
- В науке и исследованиях при поиске оптимальных решений в различных областях, таких как физика, биология, медицина и др.;

Существует большое количество задач, которые можно оптимизировать с помощью простых и не очень трудоемких методов, например, таких как:

- метод градиентного спуска;
- метод Ньютона (использует аппроксимацию функции вторым порядком для нахождения минимума функции) - может сходиться быстрее, чем метод градиентного спуска

Кроме того, имеются задачи, для которых нет известного эффективного алгоритма, который бы находил точное решение за разумное время. Эти задачи называются неразрешимыми. Такие задачи обычно характеризуются высоким ростом вычислительной сложности (например, факториальным ростом или экспоненциальным) и в этих задачах возможно отыскать только приближенное решения с некоторой точностью. Эти задачи называются NP-полными задачами оптимизации.

Так же, например, целевая функция, которую необходимо минимизировать может быть задана неявно. Алгоритмов для нахождения минимума неявно заданной функции множество: эвристические и точные. Эвристический алгоритм не гарантирует, что найдено оптимальное решение, но оно является достаточно точным для выбранной задачи. Существует большое количество эвристических методов, которые могут быть использованы для решения NP-полных задач оптимизации. Некоторые из наиболее распространенных методов, например:

- методы муравьиной колонии (Ant Colony Optimization) - используют аналогию с поведением муравьев, чтобы найти оптимальные пути и решения

- генетические алгоритмы (Genetic Algorithms) - имитируют процесс эволюции, используя генетические операторы, такие как скрещивание и мутация, чтобы создать новые поколения решений.
- методы имитации отжига (Simulated Annealing) - эти методы основаны на физическом процессе отжига металлов и используют случайный поиск с постепенным уменьшением температуры, чтобы найти оптимальные решения.

В данной работе будет рассмотрено применение метода отжига - алгоритма для решения оптимизационной задачи. Так же будет исследовано, как метод имитации отжига используется для нахождения оптимального решения в сложных, NP-полных, задачах.

Выбранная мной тема актуальная, так как задачи оптимизации являются одним из наиболее важных и широко распространенных классов задач в практическом применении. Однако, при использовании эвристических алгоритмов невозможно гарантировать нахождение оптимального решения, поэтому необходимо оценить достоверность полученного решения. Оценка погрешности решения и его точности относительно конкретной задачи представляет важность для решения задач оптимизации.

Для неявно заданных функций не всегда легко отыскать его минимум. Если даже удалось найти точное решение, возможно, для ускорения процесса стоит использовать приближенный алгоритм (который, вероятно, проще в реализации или является менее трудоемким). Тогда необходимо знать так же с какой точностью алгоритм приближено решает задачу. Если мы обладаем информацией, чему равно точное решение задачи, то рассмотрев отношение приближенного решения к точному, предварительно найдя его (приближенное решение) - таким образом, мы получим разницу между приближенным и точным решениями. То есть, пусть было получено точное решение $\alpha(a, b, c)$ и приближенное $\beta(a, b, c)$ - эти функции заданы неявно. Узнав $\inf \left| \frac{\alpha(a, b, c)}{\beta(a, b, c)} \right|$, определим, с какой точностью приближенный алгоритм близок к оптимальному. Имея информацию о погрешности приближенного решения для некоторого списка алгоритмов - не трудно выбрать тот, решение которого получается с нужной точностью. Таким образом, используя неточный алгоритм – возможно получить решение, удовлетворяющее параметрам точности, и для получения решения нет необходимости использовать точный алгоритм, который может быть сложнее в реализации и более трудоемкий.

Объектом исследования данной курсовой работы являются неявно заданные функции. Предмет исследования – минимизация таких функций эвристическими алгоритмами. В связи с поставленной целью исследования необходимо:

- 1) изучить научные материалы на данную тему;
- 2) проанализировать имеющуюся информацию;
- 3) сравнить полученные мной результаты с теми, что были изучены;
- 4) сделать выводы, определить дальнейшие перспективы для изучения темы исследования и перспективы для применения полученных результатов.

Для изучения данной темы, использовалась статья А.Ю. Чирков, В.Н. Шевченко “О приближении оптимального решения целочисленной задачи о ранце оптимальными решениями целочисленной задачи о ранце с ограничением на мощность” и работа Лопатина А.С. “Метод отжига,

Содержание

Введение:	2
Содержание	5
Глава 1. Предмет исследования.....	7
1.1. Основные определения:.....	7
1.2. Постановка задачи.....	9
Глава 2. Описание метода отжига.....	11
Глава 3. Методы Решения.....	14
3.1. Метод Method_dynamic_programming.....	14
3.1.1. Пример нахождения точного решения (метод динамического программирования)	15
3.2. Метод Approximate_to_k_elements.....	20
3.3. Метод Anneal – метод отжига.....	20
Глава 4. Программная реализация.....	23
4.1. Используемые типы данных.....	23
4.2. Описание класса BackPackItem.....	23
4.3. Вспомогательные функции.....	25
4.4. Описание вывода программы.....	25
Глава 5. Исследования.....	28
5.1. Исследование $\alpha_{1,n}$ ($n = 3, 5$).....	28
5.1.1. Исследование $\alpha_{1,3}$ ($n = 3$, и $k = 1$).....	28
5.1.2. Исследование $\alpha_{1,4}$ ($n = 4$, и $k = 1$).....	29
5.1.3. Исследование $\alpha_{1,5}$ ($n = 5$, и $k = 1$).....	30
5.1.3. Вывод для $\alpha_{1,n}$ ($n = 3, 5$).....	30
5.2. Исследование $\alpha_{n-1,n}$ ($n = 3, 6$).....	32
5.2.1. Исследование $\alpha_{2,3}$ ($n = 3$, и $k = 2$).....	32
5.2.2. Исследование $\alpha_{3,4}$ ($n = 4$, и $k = 3$).....	33
5.2.3. Исследование $\alpha_{4,5}$ ($n = 5$, $k = 4$).....	35
5.2.4. Исследование $\alpha_{5,6}$ ($n = 6$, $k = 5$).....	36
5.2.5. Вывод для $\alpha_{n-1,n}$ ($n = 3, 6$).....	37
5.3. Исследование $\alpha_{2,n}$ ($n = 0, 10$).....	38
5.3.1. Исследование $\alpha_{2,4}$ ($n = 4$, и $k = 2$).....	39
5.3.2. Исследование $\alpha_{2,5}$ ($n = 5$, и $k = 2$).....	42
5.3.3. Исследование $\alpha_{2,6}$ ($n = 6$, и $k = 2$).....	44
5.3.4. Исследование $\alpha_{2,7}$ ($n = 7$, и $k = 2$).....	44
5.3.5. Исследование $\alpha_{2,8}$ ($n = 8$, и $k = 2$).....	45
5.3.6. Исследование $\alpha_{2,9}$ ($n = 9$, и $k = 2$).....	46

5.3.7. Исследование $\alpha_{2,10}$ ($n = 10$, и $k = 2$).....	46
5.3.8. Итог исследования $\alpha_{2,n}$ ($n = 4, 10$).....	47
5.4. Исследование $\alpha_{3,5}$ ($n = 5$, $k = 3$)	49
Глава 6. Решение задачи с дополнительными ограничениями	52
6.1. Описание ограничения	52
6.2. Метод решения	53
6.3. Описание программной реализации.....	55
6.4. Описание вывода программы.....	55
6.5. Исследование	56
6.5.1. Исследование $\alpha_{2,4}$ ($n = 4$, и $k = 2$).....	56
6.5.2. Исследование $\alpha_{2,5}$ ($n = 5$, и $k = 2$).....	59
6.5.3. Итог исследования:	61
Глава 7. Использование предыдущих результатов для отыскания минимума $\alpha_{2,n}$ ($n > 4$)....	62
7.1. Идея вычисления $\alpha_{2,5}$ на основе полученных результатов $\alpha_{2,4}$	62
7.2. Исследование $\alpha_{2,5}$ с использованием результатов $\alpha_{2,4}$	63
7.3. Исследование $\alpha_{2,5}$ с использованием наилучшего решения той же задачи	65
7.4. Исследование $\alpha_{2,5}$ с использованием наилучшего решения той же задачи с ограничением на сумму компонент цен, равным 1000000.....	67
7.5. Исследование $\alpha_{2,5}$ с использованием наилучшего решения той же задачи с ограничением на сумму компонент цен, но без ограничения на вектор весов.	68
7.6. Итог исследования $\alpha_{2,5}$ с использованием наилучшего решения той же задачи с ограничением на сумму компонент цен.....	69
Глава 8. Решение задачи с понижением размерности	70
8.1. Описание идеи	70
8.2. Описание программной реализации.....	71
8.3. Описание вывода программы.....	71
8.4. Исследование $\alpha_{2,5}$	73
Заключение	77
Список литературы	80
Приложение А (метод Отжига)	81
Приложение В (ограничения на сумму компонент вектора цен и веса)	85
Приложение С (метод отжига с пониженной размерностью)	87

Глава 1. Предмет исследования

1.1. Основные определения:

Неявно заданная функция.

Функция y называется неявной относительно независимой переменной x , если функция двух переменных $F(x, y) = 0$ неразрешима относительно x .

Аналогичное определение дается в случае $n = 3, 4, \dots$ переменных.

Функция задана в явном виде, если она задана в виде $y = f(x)$ и разрешается относительно переменной y .

- Если функция задана в виде $y = f(x)$, то ее можно записать как неявно заданную: $f(x) - y = 0$ (наоборот нельзя)
- Не всегда возможно разрешить уравнение относительно y . Например:
$$x + \cos(y) + e^y = 0$$

NP-полная задача.

NP-полная задача - это класс задач в вычислительной теории, для которых не существует эффективного решения, которое бы работало за полиномиальное время от объема входных данных (т.е. решение такой задачи возможно лишь за экспоненциальное время). Для NP-полных задач существуют различные приближенные методы, которые могут дать достаточно точный результат за разумное время, но эффективных решений не существует.

- Простые вычислительные задачи могут решаться за время, которое зависит от степени полинома (т.е. время поиска точного решения поставленной задачи или количество итераций для нахождения ее решения полиномиально пропорциональны числу наблюдений начальных/исходных данных). Алгоритмы, которые имеют экспоненциальную трудоемкость при решении более сложных задач, считаются неэффективными, потому что с увеличением объема входных данных время их выполнения быстро возрастает.
- К классу NP-полных задач относятся, например: задача о рюкзаке, задача о клике, задача раскраски графа, задача коммивояжера и т.д.

Свойство NP-полной задачи:

1. Если задача из класса NP имеет полиномиальный алгоритм решения - она может быть решена за полиномиальное время.
2. Любая NP-полная задача сводится к другой задаче из этого же класса (т.е. если существует эффективный алгоритм для решения NP-полной задачи - также существует эффективный алгоритм для решения любой задачи из этого же класса.

Точная нижняя грань (infimum, inf.).

Множество M ограничено снизу, если: $\exists C > 0: \forall x \in M \ x < C$.

Точная нижняя грань (inf) – наибольшая нижняя граница.

Свойства точной нижней грани $\beta = \inf x_n$:

3. $\forall x_n: x_n > \beta$ (из определения)
4. $\forall \varepsilon > 0 \ \exists x_n: x_n < \beta + \varepsilon$.
5. Всегда существует и может быть равна одному из элементов множества либо равна числу, которое не принадлежит множеству (минимум - это наименьший элемент в множестве, но он может не существовать, если множество неограниченно снизу или пусто). Например: множество $M = \{x_n = \frac{1}{n}: n \in N\}$: $\inf_{x_n \in M} x_n = 0$, но $0 \notin M$ и минимум на этом множестве не достигается.

1.2. Постановка задачи

Поставлена целочисленная задача о ранце на максимизацию:

$$\begin{cases} \max_{x_1, \dots, x_n} (c_1 x_1 + c_2 x_2 + \dots + c_n x_n) = \max_{x_1, \dots, x_n} (\sum_{i=1}^n c_i x_i) \\ a_1 x_1 + a_2 x_2 + \dots + a_n x_n = \sum_{i=1}^n a_i x_i \leq b \end{cases} \quad (1), \text{ где:}$$

c_i – цена i -ого предмета ($i = 1, \dots, n$),

a_i – вес i -ого предмета ($i = 1, \dots, n$),

b – максимальный вес, который может быть в рюкзаке

$x_i \in Z_+, i = \overline{1, n}$

Или перепишем задачу в виде:

$$\begin{cases} \sum_{i=1}^n c_i x_i \rightarrow \max_{x_1, \dots, x_n} \\ \sum_{i=1}^n a_i x_i \leq b \end{cases} \quad (2)$$

Или, если обозначить:

$c = (c_1, c_2, \dots, c_n)^T$ – вектор цен предметов

$a = (a_1, a_2, \dots, a_n)^T$ – вектор весов предметов

$x = (x_1, x_2, \dots, x_n)^T$ – вектор, где x_i количество предметов i -ого типа.

Тогда задача перепишется в виде: $\begin{cases} \max_x cx \\ ax \leq b \end{cases} \quad (3).$

Пусть найдено оптимальное решение целочисленной задачи о рюкзаке. Точка $p \in Z_+$, на которой достигается максимум целевой функции и удовлетворяющая ограничению $\sum_{i=1}^n a_i x_i \leq b$, называется оптимальным решением целочисленной задачи о ранце. Значение функции в этой точке называется оптимальным значением задачи (1) и равно: $\lambda(a, b, c) = cp$.

Введем множество Z_k^n – множество из наборов размерности n , у которых k компонент отличны от нуля (т.е. $n-k$ компоненты равны нулю).

Точка $v \in Z_k^n$, на которой достигается максимум целевой функции и удовлетворяющая ограничению $\sum_{i=1}^n a_i x_i \leq b$, называется k -оптимальным решением целочисленной задачи о ранце (1). Тогда значение функции в этой точке называется k -оптимальным значением задачи (1) и равно: $\lambda_k(a, b, c) = cv$.

Для сравнения оптимального значения целочисленной задачи о ранце и k -оптимального значения введём следующее отношение: $\alpha_k = \frac{\lambda_k(a, b, c)}{\lambda(a, b, c)}$. При этом: если

$\lambda(a, b, c) = 0$, то будем считать, что $\alpha_k(a, b, c) = 1$. Такое отношение (величину) назовем точностью k -оптимального решения целочисленной задачи о рюкзаке (1).

Для α_k верна следующая оценка: $\forall a, c \in Z_+^n, b \in N \Rightarrow 0 \leq \left| \frac{\lambda_k(a, b, c)}{\lambda(a, b, c)} \right| \leq 1$

Тогда введем понятие гарантированной точностью k -оптимального решения целочисленной задачи о ранце (1).

Гарантированная точность k -оптимального решения целочисленной задачи о ранце (1) – это точная нижнюю грань точности k -оптимального решения этой же задачи и ее будем обозначать: $\alpha_{k,n} = \inf_{a,b,c} \left\{ \left| \frac{\lambda_k(a,b,c)}{\lambda(a,b,c)} \right| : a, c \in Z_+^n, b \in N \right\}$. Т.е. нижний индекс n обозначает, что задача о целочисленном рюкзаке поставлена для вектора C (вектор цен) и A (вектор весов), состоящего из n компонент. И точное решение – решение исходной задачи (1). Но при этом k -оптимальное значение для задачи о целочисленном рюкзаке получено, когда вектор $x = (x_1, x_2, \dots, x_n)^T \in Z_+^n$ имеет не более k компонент отличных от нуля (т.е. $n - k$ компонент равны нулю), при чем элементы зануляются таким образом, чтобы набор вносил наибольший вклад в решение задачи. Число $b \in N$ при поиске k -оптимального значения остается таким же, как и в исходной задаче.

Пусть найдено оптимальное значение целочисленной задачи о рюкзаке (точное решение): $\lambda(a, b, c)$. Пусть найдено k -оптимальное значение задачи (1): $\lambda_k(a, b, c)$.

Необходимо, используя различные методы для минимизации функции, найти:

$\inf_{a,b,c} \left| \frac{\lambda_k(a,b,c)}{\lambda(a,b,c)} \right|$, если $a, c \in Z_+^n, b \in N$. В дальнейшем, для нахождения точного решения используется метод динамического программирования (который заключается в разделении задачи на более простые подзадачи и последующем объединении их решений для получения наилучшего оптимального результата), а для минимизации функции $\alpha_k = \frac{\lambda_k(a,b,c)}{\lambda(a,b,c)}$ и нахождения значения $\alpha_{k,n} = \inf_{a,b,c} \left\{ \left| \frac{\lambda_k(a,b,c)}{\lambda(a,b,c)} \right| \right\}$ используется метод имитации отжига (simulated annealing), использующий случайный поиск с постепенным уменьшением температуры, чтобы найти оптимальные решения.

Глава 2. Описание метода отжига

Основа метода “отжига” - процесс кристаллизации (используется для повышения однородности металла).

Каждый металл имеет свою кристаллическую решетку – она определяется расположением атомов. Взаимное расположение атомов и их совокупность позиций определяет состояние системы. Качество решетки зависит от уровня энергии каждого состояния: чем он ниже, тем более идеальна решетка, меньше она имеет дефектов и прочнее металл. Метод имитации отжига использует процесс охлаждения металла для оптимизации состояния системы, перемещая ее случайным образом по пространству состояний и оценивая уровень энергии каждого нового состояния. Принятие нового состояния зависит от его уровня энергии и текущей температуры системы. Цель термической обработки методом отжига заключается в достижении минимальной энергии системы.

В процессе «отжига» металл нагревают до высокой температуры (атомы кристаллической решетки покидают свои позиции). Далее тело медленно охлаждается, и атомы стараются занять состояние с меньшей энергией, но существует определенная вероятность того, что атомы могут перейти в состояние с более высокой энергией (с уменьшением температуры вероятность перехода атомов в состояние с более высокой энергией так же уменьшается). Переход в худшее состояние помогает отыскать состояние с меньшей энергией, чем начальная - таким образом, используя этот метод, можно найти состояние с минимальной энергией, что соответствует идеальной кристаллической решетке металла. Процесс термической обработки методом отжига заканчивается, когда температура достигает заранее заданного значения, при этом происходила минимизация энергии системы с целью привести ее в состояние с наименьшей энергией, где кристаллическая решетка металла содержит меньше дефектов, в следствие чего металл становится прочнее.

Для метода имитации отжига введем следующие функции и обозначения для параметров алгоритма:

1. Множества:

1.1. S – множество всех состояний (решений) задачи.

2. Параметры:

2.1. $s_i \in S$ – состояние на i -ом шаге алгоритма.

2.2. $t_i \in R$ – температура на i -ом шаге.

3. Функции:

- 3.1. $E: S \rightarrow R$ – функция энергии (то, что необходимо оптимизировать). Функция E каждому решению ставит в соответствие число, полученное по какому-то правилу, которое зависит от конкретной решаемой задачи, а именно от оптимизируемого параметра.
- 3.2. $T: N \rightarrow R$ – функциональная связь между температурой и временем. Обязательно необходимо, чтобы функция была убывающая. Функция T ставит номеру итерации $i \in N$ в соответствие температуру $t_i \in R$. Функция T также определяет время, которое понадобится для выполнения алгоритма. Если T линейная функция, то Время, необходимое для завершения алгоритма, будет достаточно большим. В случае же степенной функции, например $T(i) = \frac{t_1}{a^i}$, алгоритм закончит свою работу очень быстро (но не факт, что таким образом метод “не застрянет” в низменности некоторого интервала области значения функции, не добравшись до “лучшего”, оптимального, значения)
- 3.3. Функция $F: S \rightarrow S$ – функция, порождающую новое состояние. Функция F создает новое состояние-кандидата на основе предыдущего состояния, и система может либо перейти в это новое состояние, либо отвергнуть его. (обозначим s_c). Способ получения кандидата так же, как и E , зависит от рассматриваемой задачи.

Разберем алгоритм метода имитации отжига (псевдокод):

На входе: t_{max} – начальная температура и t_{min} – минимальная температура, до которой необходимо уменьшать начальную температуру.

Решение:

1. Задаём произвольное первое начальное состояние s_1
2. $t_1 = t_{max}$
3. Пока не достигли минимальной температуры: $t_i > t_{min}$
 - 3.1. Получаем новое состояние: $s_c = F(s_{i-1})$
 - 3.2. Подсчитываем разницу энергий прошлого состояния и полученного:
$$\Delta E = E(s_c) - E(s_{i-1})$$
 - 3.3. Если энергия «кандидата» меньше, то он становится новым состоянием: если $\Delta E \leq 0$, то $s_i = s_c$.

Иначе (если $\Delta E > 0$), то переход осуществляется с вероятностью

$$P(\Delta E) = e^{-\frac{\Delta E}{t_i}} \text{ (такой переход называется вероятностным).}$$

3.4. Понижаем температуру: $t_{i+1} = T(i)$

На первых итерациях метода имитации отжига получаемое решение “плохое”, сильно отличающееся от желаемого оптимального, и тогда при высокой температуре можно позволить перейти в состояние хуже. Требуется, чтобы температура была высокой в начале процесса имитации отжига и постепенно уменьшалась к концу работы алгоритма. В конце работы метода наоборот — решение почти оптимальное, близкое к действительности, и не продуктивно, если выбрать решение хуже, полученного. Если новое состояние, полученное на текущей итерации, хуже предыдущего, вероятность выбора его должна быть меньше. Также вероятность перехода к худшему решению должна быть выше при более высокой температуре. Важно, чтобы температура монотонно убывала к нулю. Необходимо умножать на каждом шаге температуру на некоторый коэффициент чуть меньший единицы.

Глава 3. Методы Решения

Для решения данной задачи реализован класс `BackPackItem`.

Класс `BackPackItem` хранит следующие данные:

1. количество предметов (n)
2. вектор цен (c)
3. вектор весов (a)

В этом же классе для решения задачи о рюкзаке реализован точный метод для нахождения решения поставленной задачи – метод отжига (`Method_anneal`), метод динамического программирования (`Method_dynamic_programming`), приближенный метод (`Approximate_to_k_elements`). Рассмотрим подробнее, как работают методы

3.1. Метод `Method_dynamic_programming`

Этот метод находит точное решение задачи о рюкзаке (оптимальное значение целочисленной задачи о рюкзаке $\lambda(a, b, c)$). Метод работает следующим образом:

1. Внутри метода создаётся вектор $d[i][j]$ – максимальная стоимость любого количества вещей типов от 1 до i с суммарным весом j (до j включительно),
2. $d[0][j]$ заполняется нулями, т.к. стоимость вещей из $i = 0$ типов равно нулю.
3. Перебирая $i = \overline{0, n}$ (n – количество типов предметов), высчитываем $d[i][j]$, где $j = \overline{0, \max_weight}$ ($\max_weight = b$ – максимальная вместимость рюкзака) получим:

$$d[i][j] = \begin{cases} d[i-1][j], & \text{для } j = \overline{0, \text{weight}[i] - 1} \\ \max\{d[i-1][j]; d[i][j - \text{weight}[i]] + \text{price}[i]\}, & \text{для } j = \overline{\text{weight}[i], \max_weight} \end{cases}$$

$\text{prices}[i]$ – цена i -ого предмета.

$\text{weights}[i]$ – вес i -ого предмета.

4. После заполнения вектора $d[i][j]$ по всем i и j в $d[n][\max_weight]$ будет лежать максимальная стоимость предметов, помещающихся в рюкзак. То есть по окончании работы метода на выходе получим, что оптимальное значение целочисленной задачи о рюкзаке $\lambda(a, b, c) = d[N][\max_weight]$, где:
 n – количество различных типов предметов,
 $b = \max_weight$ – вместимость рюкзака.

3.1.1. Пример нахождения точного решения (метод динамического программирования)

Задача поставлена в следующем виде:
$$\begin{cases} \sum_{i=1}^3 c_i x_i \rightarrow \max_{x_1, \dots, x_3} \\ \sum_{i=1}^3 a_i x_i \leq b \end{cases}$$

$$c_1, c_2, c_3 = 4, 9, 11;$$

$$a_1, a_2, a_3 = 2, 3, 5;$$

$$b = 15.$$

Введем управление: u_k – количество предмета k -ого типа, взятого в рюкзак

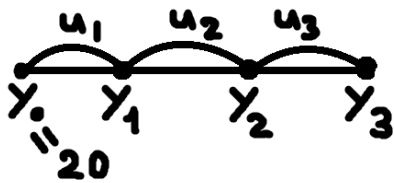
Состояние: y_k – оставшееся количества веса для загрузки в рюкзак после того, как в него загрузили k типов предметов

$$y_0 = b = 10$$

$$y_{k+1} = y_k - u_{k+1} a_{k+1}$$

$$0 \leq u_{k+1} \leq \left\lfloor \frac{y_k}{a_{k+1}} \right\rfloor$$

Стоимость от k -ого шага: $q(u_{k+1}, k+1) = u_{k+1} c_{k+1}$



Начнем решать с конца: пусть мы распределили предмет типа 1 и 2, находимся в состоянии y_2 и необходимо определить u_3 . Т.е. ищем максимальную стоимость предметов 3-го типа в рюкзаке, который можно поместить, чтобы в сумме вес не переполнил значение $b = 20$

$$(S_1(y_2) = \max_{0 \leq u_3 \leq \left\lfloor \frac{y_2}{a_3} \right\rfloor} u_3 c_3)$$

Тогда возможны следующие случаи (таблица 1):

y_2	$S_1(y_2)$	u_3
15	33	3
13	22	2
12	22	2
11	22	2
10	22	2

9	11	1
8	11	1
7	11	1
6	11	1
5	11	1
4	0	0
3	0	0
2	0	0
1	0	0

Таблица 1. Значение $S_1(y_2)$

Пусть теперь находимся в состоянии y_1 и необходимо определить u_2 . Т.е. распределили только предмет первого типа.

$$S_2(y_1) = \max_{0 \leq u_2 \leq \lfloor \frac{y_1}{a_2} \rfloor} \{u_2 c_2 + S_1(y_2)\} = \max_{0 \leq u_2 \leq \lfloor \frac{y_1}{a_2} \rfloor} \{u_2 c_2 + S_1(y_1 - u_2 a_2)\}.$$

Тогда возможны следующие случаи:

$$S_2(15) = \max_{0 \leq u_2 \leq \lfloor \frac{15}{a_2} \rfloor = 5} \begin{cases} 0 + S_1(15) \\ 9 \cdot 1 + S_1(12) \\ 9 \cdot 2 + S_1(9) \\ 9 \cdot 3 + S_1(6) \\ 9 \cdot 4 + S_1(3) \\ 9 \cdot 5 + S_1(0) \end{cases} = \max_{0 \leq u_2 \leq \lfloor \frac{15}{a_2} \rfloor = 5} \begin{cases} 0 + 33 \\ 9 \cdot 1 + 22 \\ 9 \cdot 2 + 11 \\ 9 \cdot 3 + 11 \\ 9 \cdot 4 + 0 \\ 9 \cdot 5 + 0 \end{cases} = 45, \text{ при } u_2^* = 5$$

$$S_2(13) = \max_{0 \leq u_2 \leq \lfloor \frac{13}{a_2} \rfloor = 4} \begin{cases} 0 + S_1(13) \\ 9 \cdot 1 + S_1(10) \\ 9 \cdot 2 + S_1(7) \\ 9 \cdot 3 + S_1(4) \\ 9 \cdot 4 + S_1(1) \end{cases} = \max_{0 \leq u_2 \leq \lfloor \frac{13}{a_2} \rfloor = 4} \begin{cases} 22 \\ 9 \cdot 1 + 22 \\ 9 \cdot 2 + 11 \\ 9 \cdot 3 + 0 \\ 9 \cdot 4 + 0 \end{cases} = 36, \text{ при } u_2^* = 4$$

$$S_2(11) = \max_{0 \leq u_2 \leq \lfloor \frac{11}{a_2} \rfloor = 3} \begin{cases} 0 + S_1(11) \\ 9 \cdot 1 + S_1(8) \\ 9 \cdot 2 + S_1(5) \\ 9 \cdot 3 + S_1(2) \end{cases} = \max_{0 \leq u_2 \leq \lfloor \frac{11}{a_2} \rfloor = 3} \begin{cases} 0 + 22 \\ 9 \cdot 1 + 11 \\ 9 \cdot 2 + 11 \\ 9 \cdot 3 + 0 \end{cases} = 29, \text{ при } u_2^* = 2$$

$$S_2(9) = \max_{0 \leq u_2 \leq \lfloor \frac{y_1}{a_2} \rfloor = 3} \begin{cases} 0 + S_1(9) \\ 9 \cdot 1 + S_1(6) \\ 9 \cdot 2 + S_1(3) \\ 9 \cdot 3 + S_1(0) \end{cases} = \max_{0 \leq u_2 \leq \lfloor \frac{y_1}{a_2} \rfloor = 3} \begin{cases} 0 + 11 \\ 9 \cdot 1 + 11 \\ 9 \cdot 2 + 0 \\ 9 \cdot 3 + 0 \end{cases} = 27, \text{ при } u_2^* = 3$$

$$S_2(7) = \max_{0 \leq u_2 \leq \lfloor \frac{y_1}{a_2} \rfloor = 2} \begin{cases} 0 + S_1(7) \\ 9 \cdot 1 + S_1(4) \\ 9 \cdot 2 + S_1(1) \end{cases} = \max_{0 \leq u_2 \leq \lfloor \frac{y_1}{a_2} \rfloor = 2} \begin{cases} 0 + 11 \\ 9 \cdot 1 + 0 \\ 9 \cdot 2 + 0 \end{cases} = 18, \text{ при } u_2^* = 2$$

$$S_2(5) = \max_{0 \leq u_2 \leq \lfloor \frac{y_1}{a_2} \rfloor = 1} \begin{cases} 0 + S_1(5) \\ 9 \cdot 1 + S_1(4) \end{cases} = \max_{0 \leq u_2 \leq \lfloor \frac{y_1}{a_2} \rfloor = 1} \begin{cases} 0 + 11 \\ 9 \cdot 1 + 0 \end{cases} = 11, \text{ при } u_2^* = 0$$

$$S_2(3) = 0$$

Запишем в таблицу (таблица 2):

y_1	$S_2(y_1)$	u_2
15	45	5
13	36	4
11	29	2
9	27	3
7	18	2
5	11	0
3	0	0 или 1
1	0	0

Таблица 2. Значения $d[i][j]$.

Осталось проверить состояние y_0 и определить u_1 . Т.е. распределяем только предмет первого типа, когда рюкзак еще не заполнен и $y_0 = 15$ (таблица 3):

y_0	$S_3(y_0)$	u_1
15	45	0

Таблица 3. значение $S_3(y_0)$.

$$S_3(15) = \max_{0 \leq u_1 \leq \left\lfloor \frac{y_0}{a_1} \right\rfloor = 7} \begin{cases} 0 + S_2(15) \\ 4 \cdot 1 + S_2(13) \\ 4 \cdot 2 + S_2(11) \\ 4 \cdot 3 + S_2(9) \\ 4 \cdot 4 + S_2(7) \\ 4 \cdot 5 + S_2(5) \\ 4 \cdot 6 + S_2(3) \\ 4 \cdot 7 + S_2(1) \end{cases} = \max_{0 \leq u_1 \leq \left\lfloor \frac{y_0}{a_1} \right\rfloor = 7} \begin{cases} 0 + 45 \\ 4 \cdot 1 + 36 \\ 4 \cdot 2 + 29 \\ 4 \cdot 3 + 27 \\ 4 \cdot 4 + S_2(7) \\ 4 \cdot 5 + S_2(5) \\ 4 \cdot 6 + S_2(3) \\ 4 \cdot 7 + 0 \end{cases} = 45, \text{ при } u_1^* = 0$$

Получаем:

- Ответ задачи: $\lambda(a, b, c) = \max_{x_1, \dots, x_3} \sum_{i=1}^3 c_i x_i = 45$

- $u_1^* = 0 \Rightarrow y_1 = 15 \Rightarrow u_2^* = 5 \Rightarrow y_2 = 0 \Rightarrow u_3^* = 0$

Т.е., чтобы $\sum_{i=1}^3 c_i x_i$ была максимальна, нужно в рюкзак положить 5 предметов второго типа ($c_2, a_2 = 9, 3$)

При этом: $\sum_{i=1}^3 a_i x_i = 15 \leq b$

- Или вектор $d[i][j]$, который является максимальной стоимостью любого количества вещей типов от 1 до i с суммарным весом j (до j включительно) имеет следующий вид (таблица 4):

	0	1	2	3
0	0	0	0	0
1	0	0	0	0
2	0	4	0	0
3	0	4	9	0
4	0	8	9	0
5	0	8	9	11
6	0	12	18	18
7	0	12	18	18
8	0	16	18	18
9	0	16	27	27
10	0	20	27	27
11	0	20	27	27
12	0	24	36	36
13	0	24	36	36
14	0	28	36	36
15	0	28	45	45

Таблица 4. Значения $d[i][j]$.

3.2. Метод `Approximate_to_k_elements`

Этот метод находит приближенное решение (k-оптимальное значение $\lambda_k(a, b, c)$) целочисленной задачи о рюкзаке (1)).

Так как точка v (k-оптимальное решение целочисленной задачи о ранце (1)) из множества Z_k^n , то в ней k компонент отличны от нуля: $n - k$ компонент нули. К-оптимальное значение задачи (1) – это $\lambda_k(a, b, c) = cv$. Т.е. решая приближённо задачу (1), необходимо найти k шт. компонент для точки v , отличных от нуля или, что можно свести к задаче – удалить из векторов цен и весов $a, c \in Z_+^n$ $n-k$ компонент, которые равны нулю. И тогда размерность векторов уменьшится до k: $\tilde{v}, \tilde{a}, \tilde{c} \in Z_+^k$. Все компоненты векторов $\tilde{a}, \tilde{c} \in Z_+^k$ отличны от нуля и при подстановке точки \tilde{v} в целевую функцию k-оптимальное значение целочисленной задачи о рюкзаке не изменится и будет равно: $\lambda_k(a, b, c) = \tilde{c}\tilde{v}$

Метод зануляет $n - k$ переменных (т.е. остается набор из k предметов), приносящих максимальный вклад в полученный результирующий ответ. Метод работает следующим образом:

1. Выбираются k элементов.
2. Для них высчитывается результирующая цена.
3. В дальнейшем полученная цена сравнивается с другой, полученной из следующего набора из k предметов, и выбирается максимальная. Полученный набор из k предметов, для которого цена максимальна, запоминается для дальнейших вычислений.

3.3. Метод `Anneal` – метод отжига.

С помощью данного метода находится минимальное отношение приближенного и точного решения целочисленной задачи о рюкзаке: $\alpha_{k,n} = \inf_{a,b,c} \left\{ \left| \frac{\lambda_k(a,b,c)}{\lambda(a,b,c)} \right| : a, c \in Z_+^n, b \in N \right\}$.

Описание работы метода (алгоритм):

0. Инициализация:

Для начала предполагается, что $\alpha_{k,n} = \inf_{a,b,c} \left\{ \left| \frac{\lambda_k(a,b,c)}{\lambda(a,b,c)} \right| \right\} = 1$, где $\lambda(a, b, c)$: оптимальное

значение задачи (1), $\lambda_k(a, b, c)$: k – оптимальное значение задачи (1), минимум ищется по переменным $a, c \in Z_+^n, b \in N$.

Задано начальное состояние системы: вектора $a, c \in Z_+^n$ и $b \in N$

Задана начальная температура: $t_0 = 100$

Задан минимум для температуры $t_{min} = 0.00001$.

Температура на i -ой итерации вычисляется по формуле: $t_i = \frac{t_0}{i+1} \cdot 0.1$, где i – номер итерации.

1. Далее в цикле, пока температура на i -ой итерации t_i больше, чем t_{min} (пока $t_i > t_{min} = 0.00001$):

1.1. вычисляется $\frac{\lambda_k(a,b,c)}{\lambda(a,b,c)}$ для конкретных $a, c \in N_+^n$ (либо заданы в начале работы программы равными единице, либо получены из пункта 1.4., как новое состояние метода имитации отжига)

1.2. Если $\frac{\lambda_k(a,b,c)}{\lambda(a,b,c)} \leq \alpha_{k,n}$, то $\alpha_{k,n} = \frac{\lambda_k(a,b,c)}{\lambda(a,b,c)}$

1.3. Если $\frac{\lambda_k(a,b,c)}{\lambda(a,b,c)} > \alpha_{k,n}$:

1.3.1. Вычисляется вероятность $P = e^{-\left(\alpha_{k,n} - \frac{\lambda_k(a,b,c)}{\lambda(a,b,c)}\right) \cdot t_i}$, и с вероятностью P принимаем полученное значение $\alpha_{k,n} = \frac{\lambda_k(a,b,c)}{\lambda(a,b,c)}$. Иначе отвергаем - т.е. оставляем раннее найденное.

1.4. Находим новое состояние – изменяем случайным образом вектор цен и весов $a, c \in N_+^n$ и получаем новые их компоненты.

1.5. Далее переходим к следующей итерации цикла (п.1.)

Рассмотрим более подробно, как в методе имитации отжига получить новое состояние для поставленной задачи о целочисленном рюкзаке (1). Для получения нового состояния $a, c \in Z_+^n$, в котором на следующей итерации будет посчитано отношение $\frac{\lambda_k(a,b,c)}{\lambda(a,b,c)}$: функцией $\text{rand}()$ выбираются номера компонент этих векторов, которые необходимо изменить.

Цена и вес соответствующих компонент вычисляются по следующим формулам:

- цены изменяются по формуле:

$$prices_j = \left(prices_j + (-1)^{\text{rand}() \% 2} \cdot (\text{rand}() \% 2000 + 1) \right)$$

И чтобы $prices_j$ находилось в некотором отрезке, например, $prices_j \in [0; \beta]$ – применим к $prices_j$ операцию “остаток от деления” с операндом β и прибавим 1, чтобы цена не была равна 0 в случае, если $prices_j = \beta$: $prices_j = prices_j \% \beta + 1$.

- веса изменяются по формуле:

$$weights_j = (-1)^{rand() \% 2} \cdot (rand() \% 2000 + 1)$$

И чтобы $weights_j$ находилось в некотором отрезке, например, так же

$$weights_j \in [0; \beta]: weights_j = weights_j \% \beta + 1.$$

Если в процессе вычисления нового значения для j -ой цены в $prices_j$ присвоено сосчитанное

отрицательное значение, т.е. $prices_j = (prices_j + (-1)^{rand() \% 2} \cdot (rand() \% 2000 +$

$+1)) < 0$, то $prices_j = (prices_j \cdot (-1) + 1)$, и тогда чтобы $prices_j$ находилось в некотором

заданном отрезке $[0; \beta]: prices_j = prices_j \% \beta + 1$.

Аналогично и с весом: если в процессе вычисления нового значения для j -ого веса в

$weights_j$ присвоено сосчитанное отрицательное значение $(-1)^{rand() \% 2} (rand() \% 2000 +$

$+1) < 0$, то $weights_j = (weights_j \cdot (-1) + 1)$, и тогда $weights_j = weights_j \% \beta + 1$.

При этом:

- если получили, что меняется цена j -ого предмета — не следует, что будет и изменен вес этого же предмета.
- максимальная вместимость рюкзака также меняется с каждой итерацией. Вместимость рюкзака ($b = \max_weight$) изменяется вместе и с компонентами вектора веса ($weights$), как $b = \max_weight = \sum_{i=1} weights_i$

Глава 4. Программная реализация

Программная реализация для поставленной задачи была выполнена на языке C++ в среде Microsoft Visual Studio 2019. Реализация перечисленных ниже основных методов представлена в [Приложение А (метод Отжига)]. Реализованный проект содержит с++-файл файл “Source.cpp” и один header-файл “BackPackItem.h”.

4.1. Используемые типы данных

В программной реализации вводятся следующие типы данных:

1. `BackPackItem` – класс для реализации рюкзака, содержащий точный метод решения задачи о рюкзаке, приближенный метод решения и метод имитации отжига. Реализован в файле “BackPackItem.h”.
2. `std::vector<size_t>` – вектор с компонентами типа данных `size_t` для хранения вектора цен и весов $a, c \in \mathbb{N}_+^n$.

4.2. Описание класса `BackPackItem`

Поля класса (private):

1. `std::vector<size_t> prices` – вектор цен предметов (вектор c)
2. `std::vector<size_t> weights` – вектор весов предметов (вектор a)
3. `size_t count_zero_elements` – количество различных предметов, которые можно положить в рюкзак.

Методы класса (public):

1. `int Method_dynamic_programming(int max_weight)` – метод для получения точного решения, полученного методом динамического программирования для целочисленной задачи о рюкзаке (1) вместимостью `max_weight`, в который можно положить `count_elements` различных типов предметов весом `weights` и с ценой `prices`.

Метод `Method_dynamic_programming` принимает в качестве входных параметров ограничение по весу (вместимость рюкзака): `int max_weight`.

Возвращает метод отношение приближенного решения к точному типу `float`.

2. `int Approximate_to_k_elements(float _MaxWeight, std::vector<int>& res_vec_prices, std::vector<int>& res_vec_weights)` – метод зануляет $n - k$ компонент векторов $a, c \in Z_+^n$ (т.е. остается набор из k элементов), и ищет решение задачи о рюкзаке для векторов $\tilde{a}, \tilde{c} \in Z_+^k$, состоящего уже из k ненулевых компонент. При этом вместимость рюкзака b в задаче для векторов \tilde{a}, \tilde{c} не меняется.

Метод принимает в качестве входных параметров: `int max_weight` – ограничение по весу (вместимость рюкзака b), `vector& res_vec_prices` – ссылка на результирующий вектор цен, состоящий из k компонент (т.е. вектор \tilde{c}), `vector& res_vec_weights` – ссылка на вектор весов, состоящий из k компонент (вектор \tilde{a}).

Возвращает метод полученную максимальную стоимость предметов, для целочисленной задачи о рюкзаке с набором предметов из k различных типов для векторов $\tilde{a}, \tilde{c} \in Z_+^k$ – т.е. k -оптимальное значение поставленной задачи. После работы метода вместимость рюкзака остается та же, и оставшиеся типы предметов имеют ту же стоимость и вес. Т.к. в метод были переданы ссылки на результирующие вектора цен и веса после аппроксимации, состоящие из k компонент – на выходе так же имеем информацию о предметах, которые вошли в набор из k предметов (известны их вес, цена, и количество предметов в наборе – k предметов)

3. `float Method_anneal(float _MaxWeight)` – метод имитации отжига.

Метод принимает `float _MaxWeight` – ограничение по весу ($b \in N$).

Возвращает метод отношение приближенного решения к точному типу `float`.

Для этого класса написан конструктор по умолчанию и конструктор-инициализатор:

1. конструктор по умолчанию создает `count_elements = 10` предметов различных типов, которые можно положить в рюкзак, и вектора цен и весов `prices` и `weights` заполняются покомпонентно функцией `rand()`;
2. конструктор-инициализатор принимает на вход количество различных предметов, которые можно поместить в рюкзак (`count_elements`) и вектор цен и вектор весов `prices` и `weights` (вектора уже заполнены конкретными числами типа данных `size_t`).

4.3. Вспомогательные функции

Так же написаны следующие вспомогательные функции:

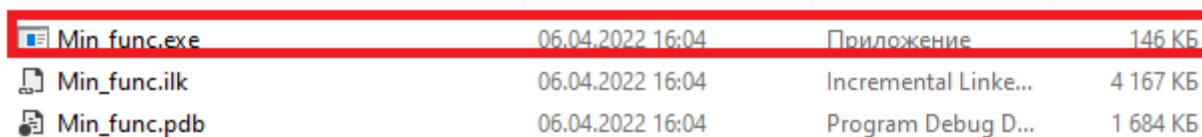
1. `bool NextSet(std::vector<size_t>& _set, int n, int m)` – функция перебирает всевозможные размещения из n по m , и используется, чтобы выбрать набор предметов из k элементов в методе `Approximate_to_k_elements` с наибольшей результирующей ценой `res_price`.

На вход функция принимает ссылку на набор чисел `_set`, число элементов в начальном наборе n , число элементов в конечном наборе m ($m < k$).

2. `void sort_price_with_weight(std::vector<int>& prices, std::vector<int>& weights)` – функция сортирует цены из вектора `prices`, полученного на вход, по возрастанию, а веса из вектора `weights` меняются местами в соответствии с изменением цен в векторе `prices` (функция принимает на вход ссылку на вектор цен и ссылку на вектор весов).
3. `void swap(int& a, int& b).`
4. `void print() const.`

4.4. Описание вывода программы

Для использования вышеописанной программной реализации по минимизации неявно заданной функции $\alpha_k = \frac{\lambda(a,b,c)}{\lambda_k(a,b,c)}$ по параметрам $a, c \in Z_+^n$, $b \in N$ в поставленной целочисленной задаче о рюкзаке(1) необходимо открыть файл `Min_func.exe` (рис. 1):





 Min_func.exe	06.04.2022 16:04	Приложение	146 КБ
 Min_func.ilk	06.04.2022 16:04	Incremental Linke...	4 167 КБ
 Min_func.pdb	06.04.2022 16:04	Program Debug D...	1 684 КБ

Рис. 1. Скриншот нахождения программы “Min_func.exe”.

При открытии программы откроется консоль.

Пусть необходимо исследовать $\alpha_{2,4}$ (т.е. $k = 2$, $n = 4$) для задачи о целочисленном рюкзаке:
$$\begin{cases} \max_x cx \\ ax \leq b \end{cases}$$
$$c = (c_1, c_2, \dots, c_n)^T$$
 – вектор цен предметов

$a = (a_1, a_2, \dots, a_n)^T$ – вектор весов предметов

$x = (x_1, x_2, \dots, x_n)^T$ – вектор, где x_i количество предметов i -ого типа.

Полученные значения $\alpha_{2,4}$, вычисленные в программе выводятся следующим образом (рис. 2):

```

Выбрать C:\Users\Rozanov\source\repos\Project3\x64\Debug\Min_func.exe
Price  Weight  SpecValue
8       9       0.888889
7       11      0.636364
3        4       0.75
6       12       0.5
des: 32
approx: 32

Prices: 1, 2, 2, 15,
Weights: 2, 3, 3, 16,
Appr prices : 2, 15,
Appr weights: 3, 16,
MaxWeight: 24
0.95 19 20

Prices: 7, 9, 12, 14,
Weights: 16, 22, 28, 27,
Appr prices : 9, 14,
Appr weights: 22, 27,
MaxWeight: 93
0.954545 42 44

```

Рис. 2. Вывод программы минимизации неявно заданной функции для $\alpha_{2,4}$

Рассмотрим, какую информацию выводит программа:

1. Prices – итоговые компоненты вектора цен $c = (c_1, c_2, \dots, c_n)^T$, при котором получилось найденное минимальное значение $\alpha_k = \frac{\lambda_k(a,b,c)}{\lambda(a,b,c)}$ методом имитации отжига.
2. Weights – итоговые компоненты вектора весов $a = (a_1, a_2, \dots, a_n)^T$, при котором получилось минимальное минимальное значение $\alpha_k = \frac{\lambda_k(a,b,c)}{\lambda(a,b,c)}$ методом имитации отжига.
3. Appr prices - итоговые компоненты вектора цен размерности k ($\tilde{c} \in Z_+^k$), при котором получилось решение задачи наибольшим, чем для других наборов из k компонент (т.е. при приближенном решении поставленной задачи).
4. Appr weights - итоговые компоненты вектора веса размерности k ($\tilde{a} \in Z_+^k$), при котором получилось решение задачи наибольшим, чем для других наборов из k компонент (т.е. при приближенном решении поставленной задачи).
5. MaxWeight - вместимость рюкзака.

6. Далее написано найденное минимальное отношение приближенного и точного решения $\alpha_k = \frac{\lambda_k(a,b,c)}{\lambda(a,b,c)}$ (минимизированная точность k-оптимального решения целочисленной задачи о ранце, равная в данном примере 0.95), приближенное решение (k-оптимальное значение целочисленной задачи о рюкзаке $\lambda_k(a,b,c)$) и точное решение (оптимальное значение целочисленной задачи о рюкзаке $\lambda(a,b,c)$).

При этом значения вектора `prices` выводятся в порядке возрастания (т.е. отсортированы) и компоненты вектора `weights` переставлены соответствующе сортировке компонент вектора `prices`.

На примере рассмотрим, какую информацию выводит программа когда $k = 3$, $n = 5$ ($\alpha_{3,5}$):

```

C:\Users\Rozanov\source\repos\Project3\x64\Debug\Min_func.exe
Price Weight SpecValue
8      12      0.666667
5      12      0.416667
6      12      0.5
8      9       0.888889
6      7       0.857143
des: 46
approx: 46

Prices: 1, 2, 2, 8, 17,
Weights: 2, 3, 3, 9, 18,
Appr prices : 1, 8, 2,
Appr weights: 2, 9, 3,
MaxWeight: 35
0.966667 29 30

```

Рис. 3. Вывод программы минимизации неявно заданной функции для $\alpha_{3,5}$

Вектор `Prices` и `Weights` состоит из $n = 5$ компонент

Вектор `Appr prices` и `Appr weights` состоит из $k = 3$ компонент.

`MaxWeight` равен 35.

Отношение приближенного решения к точному: $\frac{\lambda_3(a,b,c)}{\lambda(a,b,c)} = 0.966667$.

Глава 5. Исследования

Из статьи А.Ю. Чиркова, В.Н. Шевченко “О приближении оптимального решения целочисленной задачи о ранце оптимальными решениями целочисленной задачи о ранце с ограничением на мощность” гарантированная точность k -оптимального решения целочисленной задачи о ранце лежит в следующем отрезке: $\frac{2^n - 2^{n-k}}{2^n - 1} \leq \alpha_{k,n} \leq \frac{2^{k+1} - 2}{2^{k+1} - 1}$, где n – количество элементов, k – количество незануленных элементов для поиска приближенного k -оптимального решения целочисленной задачи о ранце (1) ($k = \overline{2, n-2}$).

Так же для задач, когда $k = 1$ и $k = n-1$ при любом n были получены конкретные значения гарантированной точности k -оптимального решения $\alpha_{k,n} = \inf_{a,b,c} \left\{ \left| \frac{\lambda_k(a,b,c)}{\lambda(a,b,c)} \right| : a, c \in Z_+^n, b \in N \right\}$

В статье доказано, что:

1. $\forall n$ и $k = 1$: $\alpha_{k,n}$ не меньше, чем 0,59135549205,
2. $\forall n$ и $k = n-1$: $\alpha_{n-1,n} = \frac{2^n - 2}{2^n - 1}$.

Рассмотрим все вышеописанные задачи и исследуем эффективность метода имитации отжига при минимизации точности k оптимального решения целочисленной задачи о рюкзаке (неявно заданная функция) – и сравним полученные результаты с точной нижней гранью точности k -оптимального решения этой же задачи или ее нижней оценкой.

5.1. Исследование $\alpha_{1,n}$ ($n = \overline{3, 5}$)

Из статьи А.Ю. Чиркова, В.Н. Шевченко [1]: $\forall n \alpha_{1,n} \geq 0,59135549205$. Рассмотрим $n = \overline{3, 5}$ и сравним полученные результаты: теоретические (из статьи [1]), и полученные из программы минимизацией методом отжига. Значения компонент вектора цен (prices) и весов (weights) будем искать на отрезке $1 \leq prices[i], weights[i] \leq 1001$.

5.1.1. Исследование $\alpha_{1,3}$ ($n = 3$, и $k = 1$)

Запустим программу, и сравним полученные минимизированные значения $\alpha_1 = \left| \frac{\lambda_1(a,b,c)}{\lambda(a,b,c)} \right|$ с теоретической оценкой $\alpha_{1,n} > 0,59135549205$ (рис. 4):

```

Price  Weight  SpecValue
2      3      0.666667
5      8      0.625
5      10     0.5
des: 14
approx: 14

Prices: 117, 420, 991,
Weights: 290, 658, 992,
Appr prices : 991,
Appr weights: 992,
MaxWeight: 1940
0.64856 991 1528

Prices: 92, 394, 834,
Weights: 228, 570, 910,
Appr prices : 834,
Appr weights: 910,
MaxWeight: 1708
0.631818 834 1320

Prices: 68, 252, 507,
Weights: 168, 413, 640,
Appr prices : 507,
Appr weights: 640,
MaxWeight: 1221
0.613059 507 827

```

Рис. 4. Полученные программой значения $\alpha_{1,3}$

Получили, что минимизированные программой значения $\left| \frac{\lambda_1(a,b,c)}{\lambda(a,b,c)} \right|$ по компонентам $a, c \in Z_+^3, b \in N$ удовлетворяют нижней оценке 0,59135549205 гарантированной точности k -оптимального решения (т.е. $\alpha_{1,3} = \inf_{a, c \in Z_+^n, b \in N} \left\{ \left| \frac{\lambda_k(a,b,c)}{\lambda(a,b,c)} \right| > 0,59135549205 \right\}$).

5.1.2. Исследование $\alpha_{1,4}$ ($n = 4$, и $k = 1$)

Проверим: удовлетворяют ли минимизированные методом отжига значения $\alpha_1 = \left| \frac{\lambda_1(a,b,c)}{\lambda(a,b,c)} \right|$ теоретической нижней оценке гарантированной точности k -оптимального решения, полученного из статьи [1]: $\alpha_{1,4} > 0,59135549205$ (рис. 5)

```

Prices: 1, 124, 170, 734,
Weights: 2, 308, 424, 735,
Appr prices : 734,
Appr weights: 735,
MaxWeight: 1469
0.666667 734 1101

Prices: 1, 55, 402, 808,
Weights: 2, 137, 487, 809,
Appr prices : 808,
Appr weights: 809,
MaxWeight: 1435
0.631744 808 1279

Prices: 1, 90, 226, 818,
Weights: 2, 224, 563, 819,
Appr prices : 818,
Appr weights: 819,
MaxWeight: 1608
0.674918 818 1212

Prices: 34, 61, 174, 582,
Weights: 84, 151, 433, 826,
Appr prices : 582,
Appr weights: 826,
MaxWeight: 1494
0.683901 582 851

```

Рис. 5. Полученные программой значения $\alpha_{1,4}$

Получили, что минимизированные программой значения $\left| \frac{\lambda_1(a,b,c)}{\lambda(a,b,c)} \right|$ по параметрам $a, c \in Z_+^4, b \in N$ так же удовлетворяют нижней оценке 0,59135549205 для гарантированной точности k-оптимального решения (1).

5.1.3. Исследование $\alpha_{1,5}$ ($n = 5$, и $k = 1$)

Рассмотрим задачу для $n = 5$ и так же проверим: удовлетворяют ли минимизированные методом отжига значениям $\alpha_1 = \left| \frac{\lambda_1(a,b,c)}{\lambda(a,b,c)} \right|$ теоретической нижней оценке гарантированной точности k-оптимального решения $\alpha_{1,5}$, полученной в статье [1] (рис. 6):

```
Prices: 2, 182, 182, 385, 935,
Weights: 3, 453, 453, 961, 936,
Appr prices : 935,
Appr weights: 936,
MaxWeight: 2806
0.750401 1870 2492

Prices: 1, 2, 2, 138, 354,
Weights: 2, 4, 4, 343, 355,
Appr prices : 2,
Appr weights: 4,
MaxWeight: 708
0.667925 354 530

Prices: 1, 1, 121, 183, 761,
Weights: 2, 2, 301, 455, 762,
Appr prices : 1,
Appr weights: 2,
MaxWeight: 1522
0.666959 761 1141
```

Рис. 6. Полученные программой значения $\alpha_{1,5}$

Получили, что минимизированные программой значения $\alpha_1 = \left| \frac{\lambda_1(a,b,c)}{\lambda(a,b,c)} \right|$ так же удовлетворяют оценке 0,59135549205.

5.1.3. Вывод для $\alpha_{1,n}$ ($n = \overline{3,5}$)

Реализованная программа минимизации неявно заданной функции $\left| \frac{\lambda_k(a,b,c)}{\lambda(a,b,c)} \right|$ была запущена для задачи о рюкзаке с $n = \overline{3,5}$ различных типов предметов, которые можно в него положить. При этом в приближенной задачи k-оптимальное решение целочисленной задачи о ранце искалась на множестве Z_1^5 ($v \in Z_1^5$) – т.е. вектор $x = (x_1, x_2, \dots, x_5)^T \in Z_+^5$ имеет не более $k = 1$ компонент отличных от нуля.

При запуске программы для $n = \overline{3,5}$ для нашей задачи получили следующие значения – выпишем минимальные и максимальные минимизированные $\left| \frac{\lambda_1(a,b,c)}{\lambda(a,b,c)} \right|$ в таблицу (таблица 5):

$\alpha_{1,n}$	$\alpha_{1,3}$	$\alpha_{1,4}$	$\alpha_{1,5}$
$\min_{a,c \in \mathbb{Z}_+^n, b \in N} \left \frac{\lambda_1(a, b, c)}{\lambda(a, b, c)} \right $	0.613059	0.631744	0.66959

Таблица 5. Значения, полученные из запуска программы для $\alpha_{1,n}$

Все минимизированные полученные значения $\alpha_1 = \left| \frac{\lambda_1(a,b,c)}{\lambda(a,b,c)} \right|$ из запуска программ удовлетворяют нижней оценке для гарантированной точности k -оптимального решения целочисленной задачи о ранце $\alpha_{1,n} > 0.59135549205$, полученной в статье А.Ю. Чиркова, В.Н. Шевченко [1].

5.2. Исследование $\alpha_{n-1,n}$ ($n = \overline{3,6}$)

Из статьи А.Ю. Чиркова, В.Н. Шевченко [1] доказано, что гарантированная точность k -оптимального решения $\forall n \alpha_{n-1,n} = \frac{2^n - 2}{2^n - 1}$. Сравним при $n = \overline{3,6}$ значения точности k -оптимального решения, полученного минимизацией методом имитации отжига, со значением точной нижней грани точности k -оптимального решения этой же задачи, полученным и доказанным из теории. Подставим $n = \overline{3,6}$ и сосчитаем, какие значения должны получиться минимизацией $\left| \frac{\lambda_k(a,b,c)}{\lambda(a,b,c)} \right|$ по переменным $a, c \in Z_+^n, b \in N$ - запишем их в таблицу (таблица 6):

$$\alpha_{2,3} = 0.857142,$$

$$\alpha_{3,4} = 0.933333,$$

$$\alpha_{4,5} = 0.9677419,$$

$$\alpha_{5,6} = 0.984126.$$

$\alpha_{2,3}$	$\alpha_{3,4}$	$\alpha_{4,5}$	$\alpha_{5,6}$
0.857142	0.933333	0.96774193	0.984126

Таблица 6. Теоретические оценки для $\alpha_{n-1,n}$

5.2.1. Исследование $\alpha_{2,3}$ ($n = 3$, и $k = 2$)

Запустим программу, и сравним полученные минимизированные значения точности k -оптимального решения $\alpha_2 = \left| \frac{\lambda_2(a,b,c)}{\lambda(a,b,c)} \right|$ с гарантированной точностью k -оптимального решения $\alpha_{2,3} = \inf_{a,b,c} \left\{ \left| \frac{\lambda_2(a,b,c)}{\lambda(a,b,c)} \right| : a, c \in Z_+^3, b \in N \right\} = 0.857142$ для этой же задачи. Значения компонент вектора цен (prices) и весов (weights) будем искать на $prices[i], weights[i] \in [1; 101]$. Программы вывела следующий результат (рис. 7):

```
Prices: 2, 4, 8,
Weights: 3, 5, 9,
Appr prices : 2, 4,
Appr weights: 3, 5,
MaxWeight: 17
0.857143 12 14
```

Рис. 7. Полученные программой значения $\alpha_{2,3}$.

Минимизацией точности k-оптимального решения при $n = 3$ и $k = 2$ методом имитации отжига получили значение, как и в теории. В этой задаче теоретические и вычисленные программой значения совпадают – т.е. методом имитации отжига удалось минимизировать $\left| \frac{\lambda_2(a,b,c)}{\lambda(a,b,c)} \right|$ до значения $\alpha_{2,3}$, полученного в статье [1].

Вектор цен (Prices)	(2, 4, 8)
Вектор весов (Weights)	(3, 5, 9)
Максимальная вместимость рюкзака	17
Вектор цен для приближенного решения из набора k предметов (Appr prices)	(2, 4)
Вектор весов для приближенного решения из набора k предметов (Appr weights)	(3, 5)
Точное решение задачи	14
Приближенное решение	12
Отношение приближенного решения к точному	0. 857142

Таблица 7. Результаты полученного решения для $\alpha_{2,3}$.

Изменяя параметры цены и веса четырех предметов во время работы метода отжига, $\alpha_{2,3} = 0.857142$ достигается при следующих параметрах (0):

Максимальная вместимость рюкзака $\max_weight = \sum weight[i] = 3 + 5 + 9 = 17$.

Точное решение задачи - оно достигается при следующем наборе предметов: каждый предмет в набор входит один раз ($2 + 4 + 8 = 14$).

Приближенное решение – оно достигается при следующем наборе предметов: три предмета ценой 4 ($3 \cdot 4 = 12$).

И тогда: $\frac{12}{14} = 0.857142$.

5.2.2. Исследование $\alpha_{3,4}$ ($n = 4$, и $k = 3$)

Для случая $n = 4$, $k = 3$ сравним полученные минимизированные значения точности k-оптимального решения $\left| \frac{\lambda_3(a,b,c)}{\lambda(a,b,c)} \right|$ с гарантированной точностью k-оптимального решения

$\alpha_{3,4} = \inf_{a,b,c} \left\{ \left| \frac{\lambda_3(a,b,c)}{\lambda(a,b,c)} \right| : a, c \in Z_+^4, b \in N \right\} = 0.933333$. Значения компонент вектора цен (prices) и

весов (weights) будем искать на отрезке $1 \leq prices[i], weights[i] \leq 100$. Программа вывела следующий результат (рис. 8):

```
Prices: 3, 6, 12, 24,
Weights: 4, 7, 13, 25,
Appr prices : 24, 3, 12,
Appr weights: 25, 4, 13,
MaxWeight: 49
0.933333 42 45
```

Рис. 8. Полученные программой значения $\alpha_{2,3}$

Минимизацией $\left| \frac{\lambda_3(a,b,c)}{\lambda(a,b,c)} \right|$ по переменным $a, c \in Z_+^4, b \in Z_+$ методом имитации отжига удалось получить значение $\alpha_{3,4} = \inf_{a,c \in Z_+^n, b \in N} \left\{ \left| \frac{\lambda_3(a,b,c)}{\lambda(a,b,c)} \right| \right\} = 0.933333$, как и в теории. В случае $n = 4$, $k = 3$ теоретические и вычисленные программой значения совпадают.

Изменяя параметры цены и веса четырех предметов во время работы метода отжига, $\alpha_{3,4} = 0.933333$ достигается при следующих параметрах (таблица 8):

Вектор цен (Prices)	(2, 4, 8)
Вектор весов (Weights)	(3, 5, 9)
Максимальная вместимость рюкзака	17
Вектор цен для приближенного решения из набора k предметов (Appr prices)	(2, 4)
Вектор весов для приближенного решения из набора k предметов (Appr weights)	(3, 5)
Точное решение задачи	14
Приближенное решение	12
Отношение приближенного решения к точном	0. 857142

Таблица 8. Результаты полученного решения для $\alpha_{2,3}$

Максимальная вместимость рюкзака $\max_weight = \sum weight[i] = 3 + 5 + 9 = 17$

Точное решение задачи - оно достигается при следующем наборе предметов: каждый предмет в набор входит один раз ($3 + 6 + 12 + 24 = 45$).

Приближенное решение - оно достигается при следующем наборе предметов: два предмета ценой три и три предмета ценой двенадцать ($2 \cdot 3 + 3 \cdot 12 = 42$).

5.2.3. Исследование $\alpha_{4,5}$ ($n = 5$, $k = 4$)

Теперь запустим программу для $n = 5$, $k = 4$ и сравним полученные минимизированные значения $\left| \frac{\lambda_4(a,b,c)}{\lambda(a,b,c)} \right|$ с теоретической оценкой $\alpha_{4,5} = 0.96774193$. Значения компонент вектора цен (prices) и весов (weights) будем так же искать на $prices[i], weights[i] \in [1; 101]$.

Программа вывела следующий результат (рис. 9):

```
Prices: 1, 2, 5, 11, 23,
Weights: 2, 3, 6, 12, 24,
Appr prices : 1, 5, 23, 2,
Appr weights: 2, 6, 24, 3,
MaxWeight: 47
0.97619 41 42
```

Рис. 9. Полученные программой значения $\alpha_{4,5}$

Минимизацией точности k -оптимального решения при $n = 5$ и $k = 4$ методом имитации отжига не удалось получить значение, как и в теории. Из программы получили: $\frac{\lambda_4(a,b,c)}{\lambda(a,b,c)} = 0.97619 > \alpha_{4,5} = 0.96774193$, т.е. получили значение чуть большее, чем $\alpha_{4,5}$ из статьи [1].

Изменяя параметры цены и веса четырех предметов во время работы метода отжига, $\alpha_{4,5} = 0.97619$ достигается при следующих параметрах (таблица 9):

Вектор цен (Prices)	(1, 2, 5, 11, 23),
Вектор весов (Weights)	(2, 3, 6, 12, 24)
Максимальная вместимость рюкзака	47
Вектор цен для приближенного решения из набора k предметов (Appr prices)	(1, 5, 23, 2)
Вектор весов для приближенного решения из набора k предметов (Appr weights)	(2, 6, 24, 3)
Точное решение задачи	24
Приближенное решение	41
Отношение приближенного решения к точному	0.97619

Таблица 9. результаты полученного решения для $\alpha_{2,3}$

Максимальная вместимость рюкзака $\sum weight[i] = 2 + 3 + 6 + 12 + 24 = 47$.

Точное решение задачи - оно достигается при следующем наборе предметов: каждый предмет в набор входит один раз ($1 + 2 + 5 + 11 + 23 = 42$).

Приближенное решение - оно достигается при следующем наборе предметов: каждый предмет взят 1 раз, и предмет ценой пять взят три раза ($1 + 5 \cdot 3 + 23 + 2 = 41$).

И тогда: $\frac{42}{45} = 0.933333$.

5.2.4. Исследование $\alpha_{5,6}$ ($n = 6, k = 5$)

Запустим программу, и сравним полученные минимизированные значения $\left| \frac{\lambda_5(a,b,c)}{\lambda(a,b,c)} \right|$ с теоретической оценкой $\alpha_{5,6} = 0.984126$. Но значения компонент вектора цен (prices) и весов (weights) будем искать на отрезке $1 \leq prices[i], weights[i] \leq 300$ - увеличим отрезок поиска компонент векторов $a, c \in Z_+^5$, т.к. в предыдущем случае с $\alpha_{4,5}$ не удалось достичь нужного значения, в отличие от $\alpha_{2,3}$ и $\alpha_{3,4}$. Программы вывела следующий результат (рис. 10):

```

C:\Users\Rozanov\source\repos\Project3\64\Debug\Min_func.exe
Price  Weight  SpecValue
5      12      0.416667
2       4       0.5
5       6      0.833333
3       5       0.6
4       9      0.444444
2       4       0.5
des: 32
approx: 32

Prices: 1, 3, 10, 21, 100, 222,
Weights: 2, 4, 11, 23, 101, 223,
Appr prices : 222, 21, 1, 10, 100,
Appr weights: 223, 23, 2, 11, 101,
MaxWeight: 364
0.997199 356 357

```

Рис. 10. Полученные программой значения $\alpha_{5,6}$

Из запуска программы получили: $\frac{\lambda_5(a,b,c)}{\lambda(a,b,c)} = 0.997199$. Минимизацией $\left| \frac{\lambda_5(a,b,c)}{\lambda(a,b,c)} \right|$ по переменным $a, c \in Z_+^n, b \in N$ методом имитации отжига не получили в значение в точности, как и в теории. Т.е. получили значение чуть большее, чем нижняя оценка из теории для $\alpha_{5,6}$. При увеличении n размерность задачи увеличивается, из-за чего и не получается при $n \geq 4$ с данной реализацией имитацией метода отжига достичь точной нижней грани точности k -оптимального решения, полученного в статье [1] при поиске компонент вектора цен (prices) и веса (weights) на отрезке $[1; 101]$ или на $[1; 301]$.

5.2.5. Вывод для $\alpha_{n-1,n}$ ($n = \overline{3,6}$)

При запуске реализованной программы минимизации неявно заданной функции $\left| \frac{\lambda_k(a,b,c)}{\lambda(a,b,c)} \right|$ для задачи о рюкзаке с $n = \overline{3,6}$ с приближенной задачей, где k -оптимальное решение целочисленной задачи о ранце искалась на множестве Z_{n-1}^n ($v \in Z_{n-1}^n$), удалось достичь значений $\alpha_{n-1,n}$ из теории (из статьи А.Ю. Чиркова, В.Н. Шевченко [1]) при $n = 3, 4$. В случае, когда $n = 5, 6$ получили минимизированное значение $\left| \frac{\lambda_k(a,b,c)}{\lambda(a,b,c)} \right|$ по переменным $a, c \in Z_+^n, b \in N$ чуть больше, чем точная нижняя грань точности k -оптимального решения, полученного в статье.

Выпишем в таблицу полученные результаты - минимальные значения $\left| \frac{\lambda_k(a,b,c)}{\lambda(a,b,c)} \right|$ для каждой задачи, где $n = \overline{3,6}$ (таблица 10):

	$\alpha_{2,3}$	$\alpha_{3,4}$	$\alpha_{4,5}$	$\alpha_{5,6}$
$\min_{a,c \in Z_+^n, b \in N} \left \frac{\lambda_{n-1}(a,b,c)}{\lambda(a,b,c)} \right $	0.857142	0.933333	0.97619	0.997199
$\alpha_{n-1,n}$	0.857142	0.933333	0.9677419	0.984126

Таблица 10. Значения, полученные из запуска программы для $\alpha_{n-1,n}$

При минимизации методом отжига k -оптимального решения $\left| \frac{\lambda_k(a,b,c)}{\lambda(a,b,c)} \right|$ по переменным $a, c \in Z_+^n, b \in N$ для достижения значений гарантированной точности k -оптимального решения целочисленной задачи о ранце $\alpha_{k-1,n} = \inf_{a,b,c} \left\{ \left| \frac{\lambda_k(a,b,c)}{\lambda(a,b,c)} \right| : a, c \in Z_+^n, b \in N \right\}$

при $n \geq 4$ необходимо либо модифицировать метод отжига для уменьшения размерности задачи, либо увеличить отрезок поиска компонент вектора цен и весов при получении нового состояния на каждой итерации метода отжига. Такое значение точной нижней грани k -оптимального значения может достигаться, например, когда одна из компонент вектора цены или веса больше, чем на искомом отрезке.

5.3. Исследование $\alpha_{2,n}$ ($n = \overline{0,10}$)

Рассмотрим значения при $k = 2, n = \overline{4,10}$. Из статьи А.Ю. Чиркова, В.Н. Шевченко должны получиться следующие значения:

$$\frac{2^4 - 2^2}{2^4 - 1} = 0.8 \leq \alpha_{2,4} \leq \frac{2^3 - 2}{2^3 - 1} = 0.857143,$$

$$\frac{2^5 - 2^3}{2^5 - 1} = 0.774194 \leq \alpha_{2,5} \leq \frac{2^3 - 2}{2^3 - 1} = 0.857143,$$

$$\frac{2^6 - 2^4}{2^6 - 1} = 0.761905 \leq \alpha_{2,6} \leq \frac{2^3 - 2}{2^3 - 1} = 0.857143$$

$$\frac{2^7 - 2^5}{2^7 - 1} = 0.755906 \leq \alpha_{2,7} \leq \frac{2^3 - 2}{2^3 - 1} = 0.857143,$$

$$\frac{2^8 - 2^6}{2^8 - 1} = 0.752941 \leq \alpha_{2,8} \leq \frac{2^3 - 2}{2^3 - 1} = 0.857143,$$

$$\frac{2^8 - 2^7}{2^8 - 1} = 0.751468 \leq \alpha_{2,9} \leq \frac{2^3 - 2}{2^3 - 1} = 0.857143,$$

$$\frac{2^8 - 2^8}{2^8 - 1} = 0.750733 \leq \alpha_{2,10} \leq \frac{2^3 - 2}{2^3 - 1} = 0.857143.$$

Выпишем таблицу с оценками, полученными из теории для $\alpha_{k,n}$ ($k = 2$ и $n = \overline{4,10}$) (таблица 11):

$\alpha_{k,n}$	$\alpha_{2,4}$	$\alpha_{2,5}$	$\alpha_{2,6}$	$\alpha_{2,7}$	$\alpha_{2,8}$	$\alpha_{2,9}$	$\alpha_{2,10}$
нижняя оценка $\alpha_{2,n}$	0.8	0.774194	0.761905	0.755906	0.752941	0.751468	0.750733
верхняя оценка $\alpha_{2,n}$	0.857143						

Таблица 11. Теоретические оценки для $\alpha_{k,n}$

Исследуем $\alpha_{2,4}, \alpha_{2,5}, \dots, \alpha_{2,10}$ при запуске программы. Будем минимизировать $\alpha_{k,n} = \alpha_{2,n}$ ($k = 2$ и $n = \overline{4,10}$) методом имитации отжига, изменяя компоненты цены ($prices[i]$, $i = 0, \dots, n - 1$) и компоненты веса ($weights[i]$, $i = 0, \dots, n - 1$) – состояние системы. При этом максимальная вместимость рюкзака (max_weight) равна сумме весов предметов в состоянии на текущей итерации ($max_weights = \sum_{i=0}^{n-1} weights[i]$). В статье [1], в отличие

от предыдущих вышеописанных случаев, не было получено конкретных значений для точности k-оптимального решения. Необходимо минимизировать k-оптимальное решения $\left| \frac{\lambda_k(a,b,c)}{\lambda(a,b,c)} \right|$ по переменным $a, c \in Z_+^n, b \in N$ и сравнить с нижней оценкой для гарантированной точности k – оптимального решения (т.е. с точной нижней гранью точности k-оптимального решения).

5.3.1. Исследование $\alpha_{2,4}$ (n = 4, и k = 2)

Пусть задано $n = 4$, и $k = 2$ ($n - k = 2$ - количество нулевых компонент в векторе решений $x = (x_1, x_2, \dots, x_4)^T$ для приближенной задачи). Тогда гарантированная точность k-оптимального решения лежит в отрезке: $\frac{2^4 - 2^2}{2^4 - 1} \leq \alpha_{2,4} \leq \frac{2^3 - 2}{2^3 - 1}$, т.е. $0.8 \leq \alpha_{2,4} \leq 0.857143$. Пусть компоненты вектора цен (prices) и вектора веса (weights) будем искать в отрезке $[1; 101]$. Программа вывела следующий результат (рис. 11):

```
Prices: 2, 9, 13, 43,
Weights: 3, 20, 14, 44,
Appr prices : 13, 2,
Appr weights: 14, 3,
MaxWeight: 81
0.946667 71 75

Prices: 1, 5, 26, 42,
Weights: 2, 7, 27, 43,
Appr prices : 42, 26,
Appr weights: 43, 27,
MaxWeight: 79
0.918919 68 74

Prices: 1, 2, 11, 35,
Weights: 2, 3, 12, 36,
Appr prices : 2, 11,
Appr weights: 3, 12,
MaxWeight: 53
0.938776 46 49

Prices: 6, 13, 13, 52,
Weights: 12, 15, 14, 53,
Appr prices : 52, 13,
Appr weights: 53, 15,
MaxWeight: 94
0.928571 78 84

Prices: 2, 2, 3, 12,
Weights: 3, 4, 4, 13,
Appr prices : 12, 3,
Appr weights: 13, 4,
MaxWeight: 24
0.9 18 20
```

Рис. 11. Результаты при $n = 4, k = 2$ на отрезке $1 \leq prices[i], weights[i] \leq 101$

Запишем в таблицу полученные минимизированные значения точности k-оптимального решения $\alpha_2 = \frac{\lambda_2(a,b,c)}{\lambda(a,b,c)}$ по параметрам $a, c \in Z_+^4, b \in N$ при $k = 2$ и $n = 4$ на отрезке $[1; 101]$ в порядке возрастания (таблица 12):

0.9
0.91819
0.928571
0.938776
0.946667

Таблица 12. Полученные значение $\alpha_{2,4}$ при $1 \leq prices[i], weights[i] \leq 101$

Получили значения, превышающие верхнюю оценку для $\alpha_{2,4}$ из теории, равную 0.857143 (нижняя оценка равна 0.8). Из запуска программы на $1 \leq prices[i], weights[i] \leq 101$ минимальное значение $\alpha_{2,4} = 0.9$.

Значение $\alpha_{2,4} = 0.9$ получилось при следующем наборе параметров (таблица 13):

Вектор цен (Prices)	(2, 2, 3, 12)
Вектор весов (Weights)	(3, 4, 4, 13),
Максимальная вместимость рюкзака	24
Вектор цен для приближенного решения из набора k предметов (Appr prices)	(12, 3)
Вектор весов для приближенного решения из набора k предметов (Appr weights)	(13, 4)
Точное решение задачи	20
Приближенное решение	18

Таблица 13. Результаты полученного решения для $\alpha_{2,4}$

Максимальная вместимость рюкзака $\sum weight[i] = 3 + 4 + 4 + 13 = 24$

Точное решение задачи - оно достигается при следующем наборе предметов: первый предмет взят один раз, третий – два раза, четвертый - один раз ($2 + 3 * 2 + 12 = 20$).

Приближенное решение - оно достигается при следующем наборе предметов: предмет ценностью 3 взят шесть раз ($18 = 3 * 6$)

И тогда: $\frac{\lambda_2(a,b,c)}{\lambda(a,b,c)} = \frac{18}{20} = 0.9$.

Методом отжига для неявно заданной функции $\alpha_{2,4}$ на отрезке $prices[i], weights[i] \in [1; 101]$ не получилось минимизировать точность k -оптимального решения $\alpha_2 = \frac{\lambda_2(a,b,c)}{\lambda(a,b,c)}$ до отрезка $\frac{2^4 - 2^2}{2^4 - 1} = 0.8 \leq \alpha_{2,4} \leq \frac{2^3 - 2}{2^3 - 1} = 0.857143$ – не удалось попасть в заданный промежуток значений из теории (статьи [1]). Тогда увеличим отрезок поиска компонент вектора цен (prices) и вектора веса (weights). Такие значения точной нижней грани k -оптимального решения может достигаться, например, когда одна из компонент вектора цены или веса больше, чем на искомом отрезке (т.е. больше, чем 101).

Рассмотрим отрезок $[1, 2001]$ для значений вектора цен (prices) и вектора веса (weights). Запустим программу, получим следующие значения (рис. 12)

```
Prices: 46, 129, 345, 929,
Weights: 114, 321, 464, 930,
Appr prices : 345, 929,
Appr weights: 464, 930,
MaxWeight: 1829
0.879227 1274 1449

Prices: 1, 245, 466, 955,
Weights: 2, 308, 561, 956,
Appr prices : 1, 466,
Appr weights: 2, 561,
MaxWeight: 1827
0.881824 1470 1667

Prices: 23, 123, 314, 778,
Weights: 56, 305, 440, 897,
Appr prices : 23, 778,
Appr weights: 56, 897,
MaxWeight: 1698
0.88853 1100 1238

Prices: 5, 145, 290, 657,
Weights: 10, 218, 414, 658,
Appr prices : 5, 657,
Appr weights: 10, 658,
MaxWeight: 1300
0.890611 977 1097
```

```
Prices: 14, 217, 417, 1049,
Weights: 33, 413, 506, 1050,
Appr prices : 1049, 217,
Appr weights: 1050, 413,
MaxWeight: 2002
0.873895 1483 1697
```

```
Prices: 46, 327, 679, 1732,
Weights: 113, 516, 915, 1889,
Appr prices : 679, 1732,
Appr weights: 915, 1889,
MaxWeight: 3433
0.86602 2411 2784
```

Рис. 12. полученные значение $\alpha_{2,4}$ при $1 \leq prices[i], weights[i] \leq 2001$

Запишем в таблицу полученные значения $\alpha_{k,n}$ при $k = 2$ и $n = 4$ на отрезке:

$1 \leq prices[i], weights[i] \leq 2001$ в порядке возрастания (таблица 14):

0.86602
0.873895
0.879227
0.881824
0.88853
0.890611

Таблица 14. Полученные значение $\alpha_{2,4}$ при $1 \leq prices[i], weights[i] \leq 2101$

На таком отрезке поиска состояния системы на каждой итерации - методом имитации отжига удалось минимизировать точность k -оптимального решения целочисленной задачи о резке до меньших значений, чем в предыдущем случае, когда отрезок поиска компонент векторов цен и веса ($prices$ и $weights$) был $[1;101]$. Но полученные значения все же получились больше, чем верхняя оценка из теории для $\alpha_{2,4}$ (из статьи [1]). Из программы было получено, что минимальное достигнутое значение, полученное методом имитации отжига при изменении параметров $a, c \in Z_+^4, b \in N$ равняется 0.86602, а в теории верхняя оценка для $\alpha_{2,4}$: 0.857143 (нижняя оценка 0.8). Запуская программу на других отрезках для значений $prices[i]$ и $weights[i]$ – методом имитации отжига не получается минимизировать до меньших значений. В таком случае необходимо модифицировать метод имитации отжига или уменьшить размерность задачи.

5.3.2. Исследование $\alpha_{2,5}$ ($n = 5$, и $k = 2$)

Пусть задано $n = 5$, и $k = 2$ ($n - k = 3$ - количество ненулевых компонент в векторе решений $x = (x_1, x_2, \dots, x_5)^T$ для приближенной задачи). Тогда гарантированная точность k -оптимального решения лежит в отрезке: $\frac{2^5 - 2^3}{2^5 - 1} \leq \alpha_{2,5} \leq \frac{2^3 - 2}{2^3 - 1}$, т.е: $0.774194 \leq \alpha_{2,5} \leq 0.857143$. Запустим программу для этой задачи, получим следующие значения (рис. 13):

```

Prices: 1, 1, 2, 3, 11,
Weights: 2, 2, 3, 4, 12,
MaxWeight: 23
0.894737 17 19

Prices: 1, 2, 5, 5, 17,
Weights: 2, 3, 6, 6, 18,
MaxWeight: 35
0.9 27 30

Prices: 1, 1, 2, 3, 11,
Weights: 2, 2, 3, 4, 12,
MaxWeight: 23
0.894737 17 19

Prices: 1, 1, 2, 3, 11,
Weights: 2, 2, 3, 4, 12,
MaxWeight: 23
0.894737 17 19

Prices: 1, 1, 2, 3, 11,
Weights: 2, 2, 3, 4, 12,
MaxWeight: 23
0.894737 17 19

```

Рис. 13. Результат при $n = 5$, $k = 2$.

При $n = 5$ и $k = 2$ минимизированные значения точности k -оптимального решения в программной реализации методом имитации отжига получились больше, чем верхняя оценка для $\alpha_{2,5}$ из теории (статья [1]) – т.е. минимизировать до заданного отрезка возможных значений гарантированной точности k -оптимального решения методом имитации отжига не удалось. Из запуска программы получили, что минимизированные значения точности k -оптимального решения лежат в отрезке $[0.89473; 0.9]$. При чем, увеличивая время работы программы (в методе отжига увеличиваем начальную температуру), получаем чаще значения равные 0.89473.

Изменяя параметры цены и веса пяти предметов во время работы метода отжига, $\alpha_{2,5} = 0.89473$ достигается при следующих параметрах (таблица 15):

Вектор цен (Prices)	(1, 1, 2, 3, 11)
Вектор весов (Weights)	(2, 2, 3, 4, 12),
Максимальная вместимость рюкзака	23
Точное решение задачи	19
Приближенное решение	17
Отношение приближенного решения к точному	0.89473

Таблица 15. Результаты полученного решения для $\alpha_{2,5}$

Точное решение задачи - оно достигается при следующем наборе предметов: один предмет с ценой 2, три предмета с ценой 3 и один предмет с ценой 11 ($2 + 3 * 2 + 11 = 19$).

Приближенное решение: 17.

И тогда: $\alpha_{2,5} = \frac{17}{19} = 0.89473$.

5.3.3. Исследование $\alpha_{2,6}$ (n = 6, и k = 2)

Пусть задано n = 6 и k = 2, тогда $n - k = 4$ количество нулевых компонент в векторе решений $x = (x_1, x_2, \dots, x_6)^T$ для приближенной задачи. Тогда: $\frac{2^6 - 2^4}{2^6 - 1} \leq \alpha_{2,6} \leq \frac{2^3 - 2}{2^3 - 1}$, т.е: $0.761905 \leq \alpha_{2,6} \leq 0.857143$. Запустим программу, получим следующие значения (рис. 14)

```
Price  Weight  SpecValue
5      6      0.833333
6      11     0.545455
2      3      0.666667
5      10     0.5
6      7      0.857143
3      5      0.6
des: 36
approx: 36

Prices: 1, 2, 2, 2, 5, 17,
Weights: 2, 3, 3, 3, 6, 18,
MaxWeight: 35
0.9 27 30

Prices: 1, 2, 2, 2, 5, 17,
Weights: 2, 3, 3, 3, 6, 18,
MaxWeight: 35
0.9 27 30
```

Рис. 14. Результат при n = 6, k = 2.

При n = 6, k = 2 получили аналогичное предыдущему случаю: не удалось минимизировать точность k-оптимального решения методом имитации отжига на выбранном для поиска отрезке компонент вектора цен и весов до верхней оценки для его точной нижней грани. Из запуска программы получили, что результат работы метода имитации отжига равен 0.9, когда верхняя оценка для задачи $\alpha_{2,6}$ равна 0.857143. ($\alpha_{2,6} \leq 0.857143$).

5.3.4. Исследование $\alpha_{2,7}$ (n = 7, и k = 2)

Пусть задано n = 7 и k = 2, тогда $n - k = 5$ количество нулевых компонент в векторе решений $x = (x_1, x_2, \dots, x_7)^T$ для приближенной задачи. Тогда: $\frac{2^5 - 2^3}{2^5 - 1} \leq \alpha_{2,7} \leq \frac{2^3 - 2}{2^3 - 1}$, т.е. $0.75591 \leq \alpha_{2,7} \leq 0.8571$. Запустим программу, получим следующие значения (рис. 15):

```

Price Weight SpecValue
3      4      0.75
3      4      0.75
6      7      0.857143
5      6      0.833333
3      7      0.428571
4      6      0.666667
3      6      0.5
des: 34
approx: 34

Prices: 2, 2, 2, 3, 3, 6, 12,
Weights: 3, 3, 3, 4, 4, 7, 13,
MaxWeight: 37
0.909091 30 33

Prices: 2, 2, 2, 3, 3, 6, 12,
Weights: 3, 3, 3, 4, 4, 7, 13,
MaxWeight: 37
0.909091 30 33

```

Рис. 15. Результат при $n = 7$, $k = 2$.

Из запуска программы, получили, что минимизированное отношение приближенного решения к точному равняется 0.909091.

5.3.5. Исследование $\alpha_{2,8}$ ($n = 8$, и $k = 2$)

Пусть задано $n = 8$ и $k = 2$, тогда $n - k = 6$ количество нулевых компонент в векторе решений $x = (x_1, x_2, \dots, x_8)^T$ для приближенной задачи. Тогда: $\frac{2^8 - 2^6}{2^8 - 1} \leq \alpha_{2,8} \leq \frac{2^3 - 2}{2^3 - 1}$, т.е. $0.7529 \leq \alpha_{2,8} \leq 0.857143$. Запустим программу, получим следующие значения (рис. 16):

```

Price Weight SpecValue
4      7      0.571429
8     11      0.727273
9     12      0.75
9     10      0.9
7      8      0.875
8     12      0.666667
3      4      0.75
7     10      0.7
des: 66
approx: 66

Prices: 2, 2, 2, 4, 4, 4, 8, 18,
Weights: 3, 3, 3, 5, 5, 5, 9, 19,
MaxWeight: 52
0.916667 44 48

```

Рис. 16. Результат при $n = 8$, $k = 2$.

Из запуска программы, получили, что отношение приближенного решения к точному: 0.916667.

5.3.6. Исследование $\alpha_{2,9}$ ($n = 9$, и $k = 2$)

Пусть задано $n = 9$ и $k = 2$, тогда $n - k = 7$ количество ненулевых компонент в векторе решений $x = (x_1, x_2, \dots, x_9)^T$ для приближенной задачи. Тогда: $\frac{2^9 - 2^2}{2^7 - 1} \leq \alpha_{2,9} \leq \frac{2^3 - 2}{2^3 - 1}$, т.е.: $0.75147 \leq \alpha_{2,9} \leq 0.8571$. Запустим программу, получим следующие значения (рис. 17):

```
Price Weight SpecValue
2 3 0.666667
5 6 0.833333
5 12 0.416667
8 9 0.888889
4 8 0.5
5 11 0.454545
5 11 0.454545
7 11 0.636364
2 3 0.666667
des: 64
approx: 64

Prices: 2, 2, 2, 3, 4, 6, 6, 6, 19,
Weights: 3, 3, 3, 4, 5, 7, 7, 7, 20,
MaxWeight: 59
0.925926 50 54

Prices: 2, 3, 3, 3, 4, 4, 6, 6, 19,
Weights: 3, 4, 4, 4, 5, 5, 7, 7, 20,
MaxWeight: 59
0.925926 50 54
```

Рис. 17. Результат при $n = 9$, $k = 2$.

Из запуска программы, получили, что минимизированное отношение приближенного решения к точному равняется 0.925926.

5.3.7. Исследование $\alpha_{2,10}$ ($n = 10$, и $k = 2$)

Пусть задано $n = 10$ и $k = 2$, тогда $n - k = 8$ количество нулевых компонент в векторе решений $x = (x_1, x_2, \dots, x_{10})^T$ для приближенной задачи. Тогда: $\frac{2^{10} - 2^2}{2^{10} - 1} \leq \alpha_{2,10} \leq \frac{2^3 - 2}{2^3 - 1}$, т.е. должны получить следующую оценку: $0.750733 \leq \alpha_{2,10} \leq 0.857143$. Запустим программу, получим следующие значения (рис. 18):

```
Price Weight SpecValue
3 4 0.75
5 10 0.5
10 11 0.909091
7 8 0.875
5 12 0.416667
4 9 0.444444
3 4 0.75
10 12 0.833333
4 9 0.444444
4 5 0.8
des: 75
approx: 75

Prices: 2, 2, 2, 2, 3, 3, 4, 6, 6, 19,
Weights: 3, 3, 3, 3, 4, 4, 5, 7, 7, 20,
MaxWeight: 59
0.925926 50 54

Prices: 1, 2, 2, 2, 2, 3, 4, 6, 6, 18,
Weights: 2, 3, 3, 3, 3, 4, 5, 7, 7, 19,
MaxWeight: 56
0.941176 48 51
```

Рис. 18. Результат при $n = 10$, $k = 2$

5.3.8. Итог исследования $\alpha_{2,n}$ ($n = \overline{4, 10}$)

При запуске реализованной программы минимизации неявно заданной функции $\left| \frac{\lambda_k(a,b,c)}{\lambda(a,b,c)} \right|$ для задачи о рюкзаке с $n = \overline{4, 10}$ с приближенной задачей, где k -оптимальное решение целочисленной задачи о ранце искалась на множестве $Z_{k=2}^n$ ($v \in Z_{k=2}^n$), не удалось достичь интервальной оценки значений $\alpha_{k,n}$ из теории (из статьи А.Ю. Чиркова, В.Н. Шевченко [1]). Из статьи [1] в задаче о ранце с ограничением на мощность, гарантированная точность k -оптимального решения лежит в отрезке:

$\frac{2^n - 2^{n-k}}{2^n - 1} \leq \alpha_{k,n} \leq \frac{2^{k+1} - 2}{2^{k+1} - 1}$, где: n – количество элементов, k – количество незануленных элементов для поиска приближенного k -оптимального решения целочисленной задачи о ранце (1) ($k = \overline{2, n-2}$).

В случае, когда $n = 4, 5, 6$ на выходе получали значения близкие к верхней оценке для гарантированной точности k -оптимального решения, но все же минимизировать $\alpha_2 = \frac{\lambda(a,b,c)}{\lambda_2(a,b,c)}$ по параметрам $a, c \in Z_+^n, b \in N$ до этих значений методом имитации отжига не удалось – получались значения чуть больше, чем верхняя оценка для точной нижней грани точности k -оптимального решения.

В задаче при $n = 4$ и $k = 2$ программная реализация была запущена при поиске состояний системы на отрезке $[1; 101]$ и на $[1; 2001]$. Увеличив отрезок поиска компонент вектора цен и весов, удалось уменьшить отношение приближенного решения к точному. Но все же, достичь интервальной оценки для $\alpha_{2,n}$ не получились.

Выпишем в таблицу полученные результаты – минимальные значения $\left| \frac{\lambda_k(a,b,c)}{\lambda(a,b,c)} \right|$ для каждой задачи, где $n = \overline{4, 10}$ при $k = 2$ (таблица 16):

$\alpha_{k,n}$	$\alpha_{2,4}$	$\alpha_{2,5}$	$\alpha_{2,6}$	$\alpha_{2,7}$	$\alpha_{2,8}$	$\alpha_{2,9}$	$\alpha_{2,10}$
$\min_{a,c \in Z_+^n, b \in N} \left \frac{\lambda_{n-1}(a, b, c)}{\lambda(a, b, c)} \right $	0.86602	0.89473.	0.9	0.909091	0.916667	0.925926	0.941176

Таблица 16. Минимальные вычисленные значения $\left| \frac{\lambda_k(a,b,c)}{\lambda(a,b,c)} \right|$

Для сравнения выпишем таблицу с оценками, полученными из теории для $\alpha_{k,n}$ ($k = 2$ и $n = 4, 10$) (таблица 17):

$\alpha_{2,n}$	$\alpha_{2,4}$	$\alpha_{2,5}$	$\alpha_{2,6}$	$\alpha_{2,7}$	$\alpha_{2,8}$	$\alpha_{2,9}$	$\alpha_{2,10}$
нижняя оценка $\alpha_{2,n}$	0.8	0.774194	0.761905	0.755906	0.752941	0.751468	0.750733
верхняя оценка $\alpha_{2,n}$	0.857143						

Таблица 17. Теоретические оценки для $\alpha_{k,n}$

При минимизации методом отжига точности k -оптимального решения $\left| \frac{\lambda_k(a,b,c)}{\lambda(a,b,c)} \right|$ по переменным $a, c \in Z_+^n, b \in N$ для достижения значений гарантированной точности k -оптимального решения целочисленной задачи о ранце $\alpha_{k=2,n} = \inf_{a,c \in Z_+^n, b \in N} \left\{ \left| \frac{\lambda_k(a,b,c)}{\lambda(a,b,c)} \right| \right\}$ при $n \geq 4$ необходимо либо модифицировать метод отжига для уменьшения размерности задачи, либо увеличить отрезок поиска компонент вектора цен и весов при получении нового состояния на каждой итерации метода отжига. Но для случая, когда $n = 4$, увеличение отрезка поиска состояний системы (компонент вектора цен и весов) более, чем [1; 2001] не приводит к улучшению результата. Для других задачах, когда $n \geq 5$ такое значение точной нижней грани k -оптимального значения, принадлежащее отрезку: $\left[\frac{2^n - 2^{n-k}}{2^n - 1}; \frac{2^{k+1} - 2}{2^{k+1} - 1} \right]$, может достигаться, например, когда одна из компонент вектора цены или веса больше, чем на искомом отрезке. Изменяя $2n$ параметров (метод отжига изменял n цен предметов и n весов), не получилось минимизировать функцию $\alpha_{k,n}$ до нужных нам теоретических значений.

В случае $n = 4, k = 2$ и $n = 5, k = 2$ получили, что минимизированное значение отношения приближенного решения к точному методом имитации отжига близки к теоретической оценке из статьи А.Ю. Чиркова, В.Н. Шевченко [1].

5.4. Исследование $\alpha_{3,5}$ ($n = 5, k = 3$)

Рассмотрев задачи по минимизации точности k -оптимального решения при $k = 2$ и $n = 4, 5$ до известной из статьи [1] интервальной оценки гарантированной точности k -оптимального решения (т.е. $\frac{2^n - 2^{n-k}}{2^n - 1} \leq \alpha_{k,n} \leq \frac{2^{k+1} - 2}{2^{k+1} - 1}$), получили, что минимизированные $\alpha_{k=2} = \frac{\lambda_{k=2}(a,b,c)}{\lambda(a,b,c)}$ по переменным $a, c \in Z_+^n, b \in N$ близки к верхней известной оценке 0.857143. Т.е. методом имитации отжига почти удалось минимизировать отношение приближенного решения к точному до интервальной оценки $\alpha_{k=2,n}$.

Теперь же рассмотрим близкую задачу к $\alpha_{2,4}$ или к $\alpha_{2,5}$ - пусть $n = 5, k = 3$. Из статьи А.Ю. Чиркова, В.Н. Шевченко [1] для гарантированной точности k -оптимального решения верна та же оценка: $\frac{2^n - 2^{n-k}}{2^n - 1} \leq \alpha_{k,n} \leq \frac{2^{k+1} - 2}{2^{k+1} - 1}$, где n – количество элементов, k – количество ненулевых элементов для поиска приближенного решения ($n-k$ элементов в векторе решений $x = (x_1, x_2, \dots, x_n)^T$). Тогда для этой задачи гарантированная точность k -оптимального решения лежит в следующем отрезке: $0.903226 = \frac{2^5 - 2^2}{2^5 - 1} \leq \alpha_{3,5} \leq \frac{2^4 - 2}{2^4 - 1} = 0.933333$. Запустим программу и сравним полученные минимизированные значения $\alpha_3 = \frac{\lambda_3(a,b,c)}{\lambda(a,b,c)}$ с интервальной оценкой для $\alpha_{3,5}$. Из запуска программы получили следующие значения (рис. 19):

```
Price Weight SpecValue
3 5 0.6
3 4 0.75
2 3 0.666667
2 3 0.666667
3 6 0.5
Des: 15
approx: 15

Prices: 1, 2, 11, 17, 47,
Weights: 2, 3, 12, 18, 48,
Appr prices : 1, 2, 17,
Appr weights: 2, 3, 18,
MaxWeight: 83
0.961538 75 78

Prices: 1, 1, 4, 14, 33,
Weights: 2, 2, 5, 15, 34,
Appr prices : 1, 14, 33,
Appr weights: 2, 15, 34,
MaxWeight: 58
0.962264 51 53

Prices: 1, 2, 5, 7, 14,
Weights: 2, 3, 6, 15, 15,
Appr prices : 14, 1, 5,
Appr weights: 15, 2, 6,
MaxWeight: 41
0.972222 35 36

Prices: 1, 2, 5, 13, 13,
Weights: 2, 3, 6, 14, 14,
Appr prices : 5, 1, 13,
Appr weights: 6, 2, 14,
MaxWeight: 39
0.970588 33 34
```

Рис. 19. Значения $\alpha_{3,5}$ из программы

Получили, что минимальное значение из выведенных в программе: 0.961538. Данное значение близко верхней границе оценки, полученной из теории для $\alpha_{3,5}$, но все же отличается на $|0.933333 - 0.961538| = 0.028205$.

Значение $\frac{\lambda_3(a,b,c)}{\lambda(a,b,c)} = 0.961538$ получилось при следующем наборе параметров (таблица 18):

Вектор цен (Prices)	(1, 2, 11, 17, 47)
Вектор весов (Weights)	(2, 3, 12, 18, 48)
Максимальная вместимость рюкзака	83
Вектор цен для приближенного решения из набора k предметов (Appr prices)	(1, 2, 17)
Вектор весов для приближенного решения из набора k предметов (Appr weights)	(2, 3, 18)
Точное решение задачи	78
Приближенное решение	75

Таблица 18. Теоретические оценки для $\alpha_{3,5}$

Максимальная вместимость рюкзака $\sum weight[i] = 2 + 3 + 12 + 18 + 48 = 83$.

Точное решение задачи – оно достигается, когда каждый предмет взят по одному разу.

Приближенное решение: 17.

Таким образом, для задачи по поиску гарантированной точности k-оптимального решения целочисленной задачи о рюкзаке при $n = 5$, $k = 3$, близкой к задаче по поиску $\alpha_{2,4}$ и $\alpha_{2,5}$, так же не получилось минимизировать методом имитации отжига с приведённой реализацией до интервальной оценки, указанной в статье [1]. Метод отжига – алгоритм, который дает различные результаты при каждом запуске в зависимости от начальных условий и параметров. Для минимизации функции методом имитации отжига (в нашем случае функции $\alpha_3 = \frac{\lambda_3(a,b,c)}{\lambda(a,b,c)}$ по переменным $a, c \in Z_+^n, b \in N$) - важно задавать оптимальные н.у и параметры метода, необходимо проводить достаточное количество итераций. Поэтому алгоритм нужно запускать несколько раз, тогда из всех результатов не трудно выбрать наименьший. Но метод отжига не гарантирует нахождение глобального минимума, хотя может быть полезен для нахождения приближенного решения задачи. Необходимо либо модифицировать метод отжига для уменьшения размерности задачи, либо увеличить отрезок поиска компонент вектора цен и весов при получении нового состояния на каждой итерации

метода отжига, либо изменить время работы программы (увеличить/ уменьшить конечную температуру).

Глава 6. Решение задачи с дополнительными ограничениями

6.1. Описание ограничения

Так как не получилось добиться теоретических нижних оценок для функции $\alpha_{2,n}$ из статьи А.Ю. Чиркова, В.Н. Шевченко [1] – введем дополнительные ограничения в программную реализацию, упрощающие поиск состояния системы (компоненты вектора цен c , вектора весов a и вместимость рюкзака b) для каждой итерации метода имитации отжига.

Напомним задачу:
$$\begin{cases} \max_x cx \\ ax \leq b \end{cases}$$

$c = (c_1, c_2, \dots, c_n)^T$ – вектор цен предметов

$a = (a_1, a_2, \dots, a_n)^T$ – вектор весов предметов

$x = (x_1, x_2, \dots, x_n)^T$ – вектор, где x_i количество предметов i -ого типа.

$\lambda(a, b, c)$ – оптимальное значение задачи.

$\lambda_k(a, b, c)$ – k -оптимальное значение задачи.

$\alpha_k = \frac{\lambda_k(a, b, c)}{\lambda(a, b, c)}$ – точность k -оптимального значения задачи.

$\alpha_{k,n} = \inf_{a, b, c} \left| \frac{\lambda_k(a, b, c)}{\lambda(a, b, c)} \right| : a, c \in Z_+^n, b \in N$ – гарантированная точность k -оптимального значения задачи.

Необходимо, используя различные методы для нахождения минимума

функции, найти: $\min_{a, b, c} \left| \frac{\lambda_k(a, b, c)}{\lambda(a, b, c)} \right|$, если $a, c \in Z_+^n, b \in N$. Для нахождения точного решения используется метод динамического программирования, а для нахождения значения

$\alpha_{k,n} = \inf_{a, b, c} \left\{ \left| \frac{\lambda_k(a, b, c)}{\lambda(a, b, c)} \right| \right\}$ используется метод имитации отжига (simulated annealing).

Введем следующее ограничение: $\sum_{i=1}^n c_i = C, \sum_{i=1}^n a_i = A$, где C и A – константы из Z_+ .

Ранее было введено, что $b = \sum_{i=1}^n a_i$ (вместимость рюкзака). Тогда теперь введенная константа A будет равна вместимости рюкзака. Такое ограничение вводится, чтобы ограничить область поиска решения поставленной задачи. Таким образом, при сужении области поиска решения мы исключаем из рассмотрения некоторое количество совпадающих решений, но имеющие разные цены c_i и веса a_i .

6.2. Метод решения

Для введения ограничения $\sum_{i=1}^n c_i = C$ и $\sum_{i=1}^n a_i = A$ написаны две функции:

1. `change_prices_to_limit`
2. `change_weights_to_limit`

Эти функции принимают на вход вектор цен и вектор весов соответственно. Функция `change_prices_to_limit` возвращает сумму полученного вектора цен, а функция `change_weights_to_limit` возвращает сумму полученного вектора весов. Разберем более подробно `change_prices_to_limit` (функция для ограничения весов работает аналогично).

Функция `change_prices_to_limit` получает на вход вектор цен c . Далее подсчитывается сумма всех компонент вектора цен $tmp_sum_prices = \sum c_i$. Если сумма не равна заданному ограничению (для цен ограничение равно A , т.е. если $tmp_sum_prices \neq A$), то попадаем в цикл `while`, пока tmp_sum_prices не станет равна A . Каждую итерацию цикла `while` генерируется индекс (`index`) и значение (`value`), на которое будет изменена компонента вектора цен c_{index} . Далее, проверяется условие: суммарная цена всех компонент вектора c больше введенного ограничения или меньше ($tmp_sum_prices > A$ или $tmp_sum_prices < A$). В случае, если сумма цен меньше ограничения, то к компоненте c_{index} прибавляем сгенерированное значение `value`. Если же сумма цен больше ограничения, то проверяем не больше ли сгенерированное значение `value` (которое нужно вычесть из компоненты вектора c), чем цена с этим индексом c_{index} . В случае если $value > c_{index}$, то ничего не происходит и цикл переходит на следующую итерацию. Если же $value < c_{index}$, то из c_{index} вычитается значение `value` и происходит переход на следующую итерацию с дальнейшей проверкой, не стала ли сумма компонент вектора цен равна введенному ограничению A .

Далее разберем псевдокод этой функции (`change_prices_to_limit`):

```
func change_prices_to_limit (vec& prices):
```

1. $tmp_sum_prices = \sum c_i$
2. Пока: $tmp_sum_prices \neq A$
 - 2.1. `index = generation(index)`
 - 2.2. `value = generation(value)`

```
2.3. Если: tmp_sum_prices < A
    2.3.1. c_index += value
    2.3.2. tmp_sum_prices += value
2.4. Если: tmp_sum_prices > A
    2.4.1. Если: tmp_sum_prices > value)
    2.4.2. c_index = c_index - value
    2.4.3. tmp_sum_prices = tmp_sum_prices - value
3. return tmp_sum_price
```

6.3. Описание программной реализации

Программная реализация для работы с поставленной задачей была продолжена на языке C++ в среде Microsoft Visual Studio 2019. Полная реализация перечисленных ниже методов и классов представлена в [Приложение В]. К проекту добавился один .h файл “ Restrictions.h”, в котором реализованы следующие функции:

```
1. int change_prices_to_limit(std::vector<int>& prices.
```

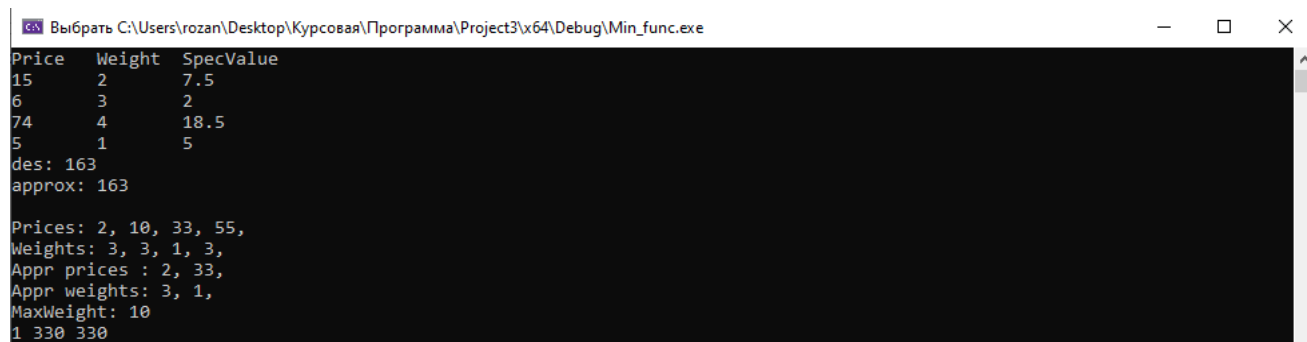
На вход функция принимает ссылку на вектор цен и возвращает значение `int` - сумму компонент вектора цен. Эта функция изменяет вектор цен, полученный на входе так, чтобы сумма всех компонент равнялась заранее заданному ограничению.

```
2. int change_weights_to_limit(std::vector<int>&weights.
```

На вход принимает ссылку на вектор весов и возвращает значение `int` - сумму компонент вектора весов. Эта функция изменяет вектор весов, полученный на входе так, чтобы сумма всех компонент равнялась заранее заданному ограничению.

6.4. Описание вывода программы

Формат вывода программы не изменился, т.к. к программной реализации добавилось только ограничение. Выводимые цены и веса равны заданному ограничению (рис. 20)



```
Выбор C:\Users\rozan\Desktop\Курсовая\Программа\Project3\x64\Debug\Min_func.exe
Price  Weight  SpecValue
15     2        7.5
6      3        2
74     4       18.5
5      1        5
des: 163
approx: 163

Prices: 2, 10, 33, 55,
Weights: 3, 3, 1, 3,
Appr prices : 2, 33,
Appr weights: 3, 1,
MaxWeight: 10
1 330 330
```

Рис. 20. Вывод программы при введенных ограничениях на вектор цен и весов

Как видим, сумма всех цен равна 100 – введенное ограничение на сумму компонент вектора цен ($2 + 10 + 33 + 55 = 100$), и сумма всех весов равна 10 - ограничение на сумму компонент вектора весов ($3 + 3 + 1 + 3 = 10$).

6.5. Исследование

Ранее, без введенного ограничения, не получилось добиться теоретических нижних оценок для функции $\alpha_{2,n}$ из статьи А.Ю. Чиркова, В.Н. Шевченко [1]. Запустим программу с введенным ограничением, и исследуем полученный результат. Из статьи должны получиться следующие значения:

$$\frac{2^4 - 2^2}{2^4 - 1} = 0.8 \leq \alpha_{2,4} \leq \frac{2^3 - 2}{2^3 - 1} = 0.857143,$$

$$\frac{2^5 - 2^3}{2^5 - 1} = 0.774194 \leq \alpha_{2,5} \leq \frac{2^3 - 2}{2^3 - 1} = 0.857143,$$

$$\frac{2^6 - 2^4}{2^6 - 1} = 0.761905 \leq \alpha_{2,6} \leq \frac{2^3 - 2}{2^3 - 1} = 0.857143$$

$$\frac{2^7 - 2^5}{2^7 - 1} = 0.755906 \leq \alpha_{2,7} \leq \frac{2^3 - 2}{2^3 - 1} = 0.857143,$$

$$\frac{2^8 - 2^6}{2^8 - 1} = 0.752941 \leq \alpha_{2,8} \leq \frac{2^3 - 2}{2^3 - 1} = 0.857143,$$

$$\frac{2^8 - 2^7}{2^8 - 1} = 0.751468 \leq \alpha_{2,9} \leq \frac{2^3 - 2}{2^3 - 1} = 0.857143,$$

$$\frac{2^8 - 2^8}{2^8 - 1} = 0.750733 \leq \alpha_{2,10} \leq \frac{2^3 - 2}{2^3 - 1} = 0.857143.$$

6.5.1. Исследование $\alpha_{2,4}$ ($n = 4$, и $k = 2$)

Пусть задано $n = 4$, и $k = 2$ ($n - k = 2$ - количество нулевых компонент в векторе решений $x = (x_1, x_2, \dots, x_4)^T$ для приближенной задачи). Пусть компоненты вектора цен и компоненты вектора веса будем искать на следующем отрезке: $prices[i], weights[i] \in [1; 2001]$. Возьмем такой отрезок поиска состояния системы для каждой итерации метода имитации отжига, т.к. в исследовании этой же задачи без введенного ограничения на сумму компонент вектора цен и весов [п. 5.3.1.] мы получили наименьшее значение минимизируемой функции $\alpha_2 = \frac{\lambda_2(a,b,c)}{\lambda(a,b,c)}$ именно на этом отрезке (в [п. 5.3.1.] рассматривался также отрезок для вектора цен и весов $[1; 101]$, но методом отжига удалось уменьшить минимизируемое значение на отрезке $[1; 2001]$). Запустим программу, получим следующий результат(рис. 21):


```

Prices: 350, 1124, 2256, 6270,
Weights: 40, 94, 145, 292,
Appr prices : 6270, 2256,
Appr weights: 292, 145,
MaxWeight: 571
0.8526 8526 10000

Prices: 16, 381, 759, 1685,
Weights: 38, 610, 1098, 1850,
Appr prices : 381, 1685,
Appr weights: 610, 1850,
MaxWeight: 3596
0.861316 2447 2841

```

Рис. 21. Результаты при $n = 4, k = 2$ при $1 \leq prices[i], weights[i] \leq 2001$ с введенным ограничением

С введенным ограничением на сумму компонент вектора цен и весов отношение приближенного решения к точному $\alpha_2 = \frac{\lambda_2(a,b,c)}{\lambda(a,b,c)}$ удалось минимизировать до 0.8526 (где $\lambda_2(a, b, c)$ - приближенное решение задачи о рюкзаке, $\lambda(a, b, c)$ – функция точного решения для поставленной целочисленной задачи о рюкзаке). Полученное значение 0.8526 удовлетворяет интервальной оценке для гарантированной точности k -оптимального решения $\alpha_{2,4}$ из статьи [1]: $(0.8 \leq 0.8526 \leq 0.857143)$.

Ранее, без введенного ограничения, результат минимизации отношения приближенного решения к точному был равен: 0.86602. Таким образом, введя ограничение на сумму цен и весов, удалось минимизировать точность k -оптимального значения по параметрам $a, c \in N_+^4, b \in N_+$ и попасть в интервал теоретических оценок для гарантированной точности k -оптимального значения. Но добиться нижней оценки, равной 0.8 не удалось. Полученное число 0.8526 ближе к верхней оценке $(0.8526 - 0.8 = 0.0526, 0.857143 - 0.8526 = 0.004543)$.

Рассмотрим, при каком наборе параметров целочисленной задачи о рюкзаке получилось минимизированное значение точности k -оптимального решения, равное 0.8526 (таблица 19):

Вектор цен (Prices)	(350, 1124, 2256, 6270)
Вектор весов (Weights)	(40, 94, 145, 292)
Максимальная вместимость рюкзака	571
Вектор цен для приближенного решения из набора k предметов (Appr prices)	(6270, 2256)

Вектор весов для приближенного решения из набора k предметов (Appr weights)	(292, 145)
Точное решение задачи	10000
Приближенное решение	8526
Отношение приближенного решения к точному	0.8526

Таблица 19. Оптимальные параметры задачи поиска $\alpha_{2,4}$

Максимальная вместимость рюкзака $\sum weight[i] = 40 + 94 + 145 + 292 = 571$.

Точное решение задачи – оно достигается, когда каждый предмет взят по одному разу ($350 \cdot 1 + 1124 \cdot 1 + 2256 \cdot 1 + 6270 \cdot 1 = 10000$)

Приближенное решение – оно достигается при следующем наборе: каждый предмет из двух взят по одному разу ($6270 \cdot 1 + 2256 \cdot 1 = 8526$). При этом их вес в сумме: $292 \cdot 1 + 145 \cdot 1 = 437 < 571$, но и добавить из набора двух предметов больше ничего нельзя, т.к. $437 + 145 = 582 > 571$.

Получили, что отношение приближенного решения к точному: $\frac{8526}{10000} = 0.8526$.

6.5.2. Исследование $\alpha_{2,5}$ ($n = 5$, и $k = 2$)

Пусть задано $n = 5$ и $k = 2$, тогда $n - k = 3$ количество нулевых компонент в векторе решений $x = (x_1, x_2, \dots, x_5)^T$ для приближенной задачи Пусть компоненты вектора цен и компоненты вектора веса будем искать в отрезке $[1; 2001]$. Запустим программу, получим следующие значения (рис. 22):

```
Prices: 1, 9, 140, 255, 595,
Weights: 2, 21, 157, 215, 396,
Appr prices : 595, 140,
Appr weights: 396, 157,
MaxWeight: 791
0.874126 875 1001

Prices: 53, 199, 819, 2803, 6126,
Weights: 59, 51, 110, 271, 509,
Appr prices : 819, 6126,
Appr weights: 110, 509,
MaxWeight: 1000
0.889751 9402 10567

Prices: 7, 331, 908, 2588, 6166,
Weights: 8, 278, 472, 999, 1860,
Appr prices : 6166, 908,
Appr weights: 1860, 472,
MaxWeight: 3617
0.889 8890 10000

Prices: 183, 372, 449, 2757, 6239,
Weights: 165, 373, 244, 915, 1733,
Appr prices : 2757, 449,
Appr weights: 915, 244,
MaxWeight: 3430
0.886493 9169 10343
```

Рис. 22. Результаты при $n = 5$, $k = 2$ при $1 \leq prices[i], weights[i] \leq 2001$ с введенным ограничением

Запишем в таблицу полученные минимизированные значения точности k -оптимального решения $\alpha_2 = \frac{\lambda_2(a,b,c)}{\lambda(a,b,c)}$ по параметрам $a, c \in Z_+^5, b \in N$ при $k = 2$ и $n = 5$ на отрезке $[1; 2001]$ в порядке возрастания (таблица 20):

0.874126
0.886493
0.889
0.889751

Таблица 20. Полученные значение $\alpha_{2,5}$ с введенными ограничениями

С введенным ограничением точность k -оптимального решения удалось минимизировать до 0.874126. Полученное число 0.874126 не удовлетворяет теоретической интервальной оценке для значений $\alpha_{2,5}$ ($0.874126 > 0.857143$, $0.874126 - 0.857143 = 0.016983$).

Ранее, без введенного ограничения, результат минимизации отношения приближенного решения к точному был равен: 0.89473. Таким образом, введя ограничение на сумму цен, удалось минимизировать отношение $\alpha_2 = \frac{\lambda_2(a,b,c)}{\lambda(a,b,c)}$, чем при запуске метода без ограничения [п. 5.3.2.], но попасть в интервал теоретических оценок не удалось.

Разберем, при каком наборе параметров получилось значение $\frac{\lambda_2(a,b,c)}{\lambda(a,b,c)} = 0.874126$

(таблица 21):

Вектор цен (Prices)	(1, 9, 140, 255, 595)
Вектор весов (Weights)	(2, 21, 157, 215, 396)
Максимальная вместимость рюкзака	791
Вектор цен для приближенного решения из набора k предметов (Appr prices)	(595, 140)
Вектор весов для приближенного решения из набора k предметов (Appr weights)	(396, 157)
Точное решение задачи	1001
Приближенное решение	875
Отношение приближенного решения к точному	0.874126

Таблица 21. теоретические оценки для $\alpha_{2,5}$

Максимальная вместимость рюкзака $\sum weight[i] = 2 + 21 + 157 + 215 + 396 = 791$.

Точное решение задачи – оно достигается, когда предмет с ценой 1 взят одиннадцать раз, и предметы с ценой 140, 255, 595 взяты по одному разу ($11 \cdot 1 + 140 \cdot 1 + 255 \cdot 1 + 595 \cdot 1 = 1001$).

Приближенное решение – оно достигается при следующем наборе: предмет ценой 595 взят один раз и предмет ценой 140 взят два раза ($595 \cdot 1 + 140 \cdot 2 = 875$). При этом их вес в сумме: $396 \cdot 1 + 157 \cdot 2 = 710 < 791$, но и добавить из набора двух предметов больше ничего нельзя, т.к. $710 + 140 = 850 > 791$.

Получили, что отношение приближенного решения к точному: $\frac{875}{1001} = 0.874126$.

И при этом ограничение на сумму цен выполняется: $1000 (1 + 9 + 140 + 225 + 595 = 1000)$.

6.5.3. Итог исследования:

Чтобы полученные результаты минимизации точности k-оптимального решения с введенным ограничением на сумму компонент вектора цен и весов (вектора с и а) сравнить с результатами в п.[5.3], когда ограничений введено не было – запишем таблицу с вычисленными в программе минимальными значениями функции $\alpha_k = \frac{\lambda_k(a,b,c)}{\lambda(a,b,c)}$ среди нескольких запусков алгоритма. Для сравнения результатов рассмотрим для задачи по поиску $\alpha_{k,n}$, где $k = 2$ и $n = \overline{4, 10}$ (таблица 22):

	$\alpha_{2,4}$	$\alpha_{2,5}$	$\alpha_{2,6}$	$\alpha_{2,7}$	$\alpha_{2,8}$	$\alpha_{2,9}$	$\alpha_{2,10}$
$\min_{a,c \in Z_+^n, b \in N} \left \frac{\lambda_{n-1}(a,b,c)}{\lambda(a,b,c)} \right $	0.86602	0.89473.	0.9	0.909091	0.916667	0.925926	0.941176
$\min_{a,c \in Z_+^n, b \in N} \left \frac{\lambda_{n-1}(a,b,c)}{\lambda(a,b,c)} \right $ с ограничением	0.8526	0.874126	-	-	-	-	-
$\alpha_{2,n}$ нижняя оценка	0.8	0.774194	0.761905	0.755906	0.752941	0.751468	0.750733
$\alpha_{2,n}$ верхняя оценка	0.857143						

Таблица 22. минимальные вычисленные значения $\alpha_{n,k}$

Методом отжига с введенным ограничением на сумму компонент цен в случае $n = 4$, $k = 2$ получилось минимизировать отношение приближенного решения к точному до интервальной оценки гарантированной точности k-оптимального решения из статьи [1]. В случае больших, начиная с $n = 5$ (при $k = 2$), получили меньшее значение минимизируемой по параметрам $a, c \in Z_+^n, b \in N$ функции $\alpha_k = \frac{\lambda_k(a,b,c)}{\lambda(a,b,c)}$, чем при запуске программы без ограничения, но попасть в интервальную оценку для гарантированной точности k-оптимального решения $\alpha_{k,n}$ из статьи [1] не удалось.

Глава 7. Использование предыдущих результатов для отыскания минимума $\alpha_{2,n}$ ($n > 4$)

Для поставленной задачи по поиску точной нижней грани k -оптимального решения целочисленной задачи о ранце $\alpha_{2,n}$ при $n = 4$ ранее получилось минимизировать отношение приближенного решения к точному $\frac{\lambda_k(a,b,c)}{\lambda(a,b,c)}$ до значения из отрезка $0.8 \leq \alpha_{2,4} = 0.8526 \leq 0.857143$, введя ограничения на сумму цен и весов. Но минимальные вычисленные программой значения $\alpha_{2,n}$ для случая, когда $n > 4$, не принадлежат соответствующему отрезку из статьи [1]. Чем больше n , тем большая разница между полученным минимизированным значением $\frac{\lambda_k(a,b,c)}{\lambda(a,b,c)}$ при найденных методом имитации отжига $a, c \in Z_+^n, b \in N_+$ и между верхней оценкой для гарантированной точности k -оптимального решения $\alpha_{2,n}$. Т.к. работа алгоритма метода имитации отжига зависит от начальных условий и параметров метода запуска программы - попробуем использовать полученные результаты из исследования $\alpha_{2,4}$ для минимизации $\frac{\lambda_k(a,b,c)}{\lambda(a,b,c)}$ в задаче с $\alpha_{2,5}$. Если получится минимизировать $\frac{\lambda_k(a,b,c)}{\lambda(a,b,c)}$ до необходимых значений таким образом – вычислим $\alpha_{2,6}$, используя полученные результаты, для вычисления $\alpha_{2,7}$ используем результаты для $\alpha_{2,6}$ и т.д.

7.1. Идея вычисления $\alpha_{2,5}$ на основе полученных результатов $\alpha_{2,4}$

Напомним, что в задаче по поиску $\alpha_{2,4}$ значение точности k -оптимального решения, 0.8526 получилось при следующем наборе параметров:

- вектор цен: $c = (350, 1124, 2256, 6270)$,
- вектор весов: $a = (40, 94, 145, 292)$,
- вектор цен для приближенного решения из набора k предметов: $\text{Appr_prices} = (6270, 2256)$
- вектор весов для приближенного решения из набора k предметов: $\text{Appr_weights} = (292, 145)$
- Ограничение на сумму цен: 10000 ($\sum_{i=1}^5 c_i = 350 + 1124 + 2256 + 6270 = 10000$)

Максимальная вместимость рюкзака $\sum a_i = 40 + 94 + 145 + 292 = 571$.

Приближенное решение – оно достигается при следующем наборе: каждый предмет из двух взят по одному разу ($\lambda_k = 6270 \cdot 1 + 2256 \cdot 1 = 8526$).

Точность k-оптимального решения: $\alpha_2 = \frac{\lambda_2(a,b,c)}{\lambda(a,b,c)} = \frac{8526}{10000} = 0.8526$.

Тогда для $\alpha_{2,5}$ в качестве первых четырех компонент для вектора цен (с) и вектора весов (а) возьмем те же, что и в случае $\alpha_{2,4}$. Пятую компоненту вектора цен и весов сгенерируем случайным образом. Далее подсчитаем b , как сумму всех компонент вектора весов: $b = \sum_{i=1}^5 a_i = B$ ($B \in N$ – раннее введение ограничение на сумму компонент вектора весов [глава 6]). И ограничение на сумму цен, как и в случае $\alpha_{2,4}$ выберем равным $A = 10000$. Идея такого метода заключается в том, что решение близкой задачи поиска точной нижней грани k-оптимального решения $\alpha_{2,5}$ находится около решения задачи $\alpha_{2,4}$ - таким образом, используя метод отжига, попробуем отыскать решение задачи поиска $\alpha_{2,5}$ вблизи полученного решения для задачи минимизации $\alpha_{2,4}$.

7.2. Исследование $\alpha_{2,5}$ с использованием результатов $\alpha_{2,4}$

Запустим программу при параметрах, описанных выше [п. 7.1]. (т.е. параметры, при которых было получен $\alpha_{2,4}$ в [п. 6.5.1.]). Получим следующий результат (рис. 23).

```
Prices: 346, 452, 828, 1522, 1827,
Weights: 335, 105, 191, 1107, 342,
Appr prices : 1827, 452,
Appr weights: 342, 105,
MaxWeight: 655
0.878983 2731 3107
```

Рис. 23. $\alpha_{2,5}$ с использованием результатов $\alpha_{2,4}$

Таким образом, используя решение задачи по поиску точной нижней грани точности k-оптимального решения $\alpha_{2,4}$, не удалось получить меньшее значение, чем найденное ранее, в задаче по поиску минимума функции $\alpha_2 = \frac{\lambda_2(a,b,c)}{\lambda(a,b,c)}$ в случае $\alpha_{2,5}$. Наименьшее достигнутое значение $\frac{\lambda_2(a,b,c)}{\lambda(a,b,c)}$ для $\alpha_{2,5}$ получилось при введение ограничения на сумму компонент вектора цен и весов [п. 6.5.2.], и равняется 0.874126 (из статьи [1] $\alpha_{2,5}$ лежит в пределах от 0.774194 и до 0.857143).

Взяв в качестве первых четырех компонент для вектора цен (prices) и вектора весов (weights) те же, что и в $\alpha_{2,4}$ и добавив к ним пятую случайно-сгенерированную, в большинстве случаев на первой итерации значение $\frac{\lambda_2(a,b,c)}{\lambda(a,b,c)} = 1$. Так как сумма первых четырех компонент уже равна 10000 (потому что эти компоненты вектора цен и весов из задачи по поиску $\alpha_{2,4}$ с введенным ограничением на сумму компонент вектора цен и весов), и ограничение на сумму компонент цены в этой задаче так же равно 10000, то функция `change_prices_to_limit` (описанная в [п.6.3.]) изменит значение некоторых компонент вектора цен так, чтобы их сумма была равна введенному ограничению A. Начальные значения вектора цен были: prices = (350, 1124, 2256, 6270). Но после применения функции `change_prices_to_limit` и `change_weight_to_limit` значения векторов изменились на следующие: prices (350, 1124, 2256, 6270, 70) (рис. 24):

```
{ size=5 }
  [capacity]: 5
  [allocator]: allocator
  [0]: 350
  [1]: 1124
  [2]: 2256
  [3]: 6270
  [4]: 70

prices
{ size=5 }
  [capacity]: 6
  [allocator]: allocator
  [0]: 70
  [1]: 1048
  [2]: 1141
  [3]: 2256
  [4]: 5485
  [Базовое представление]: {_MyPair=allocator }
tmp_inf_ratio
1.00000000
price_approx
15720
price_dynamic_programming
15720
```

Рис. 24. $\alpha_{2,5}$ на первой итерации при использовании результатов $\alpha_{2,4}$

При этом добавленная пятая компонента вектора цен не изменилась в функции `change_prices_to_limit`, сумма компонент вектора цен теперь равняется 10000 (введенному ограничению A), и отношение приближенного решения к точному равняется 1, т.к. решение приближенной задачи и точное решение равняется 15720 (каждый предмет на первой итерации в рюкзак для точной задачи кладется не один раз, т.к. иначе бы $\lambda(a,b,c) = 10000 = A$).

7.3. Исследование $\alpha_{2,5}$ с использованием наилучшего решения той же задачи

При использовании результатов задачи по поиску $\alpha_{2,4}$ для задачи по поиску гарантированной точности k-оптимального решения $\alpha_{2,5}$ [п. 7.2.] из-за введенного ограничения на сумму компонент цен менялось так же и начальное состояние системы (вектор цен) при запуске первой итерации метода имитации отжига. Т.к. в задаче по поиску $\alpha_{2,4}$ в итоге получили вектор цен и весов, состоящий из 4 компонент, то в задаче с $\alpha_{2,5}$, добавив к вектору цен еще одну компоненту, вектор цен изменился так, чтобы сумма всех пяти компонент prices равнялась введенному ограничению А. Т.е. решать задачу по поиску $\alpha_{2,5}$ с использованием результатов $\alpha_{2,4}$ не получилось, т.к. задачи хоть и близки, но имеют разную размерность.

Ранее, введя ограничения на сумму цен и весов, получилось минимизировать $\alpha_{2,5}$ до меньшего значения, чем в [п.5.3.2.] когда ограничений не было. Но все же минимальные вычисленные программой значения $\alpha_2 = \frac{\lambda_2(a,b,c)}{\lambda(a,b,c)}$ для этой же задачи больше, чем верхняя оценка для $\alpha_{2,5}$ из статьи [1] (т.е. минимизировать выбранным методом отжига с введенным ограничением на сумму компонент вектора цен и весов в данном случае не удалось). При увеличении n так же и увеличивается разность между верхней оценкой из теории для точной нижней грани k-оптимального значения задачи $\alpha_{2,5}$ и между минимизированным программой отношением приближенного решения задачи к точному $\alpha_2 = \frac{\lambda_2(a,b,c)}{\lambda(a,b,c)}$.

Тогда для минимизации функции $\alpha_2 = \frac{\lambda_2(a,b,c)}{\lambda(a,b,c)}$ для задачи по поиску $\alpha_{2,5}$ попробуем использовать параметры уже найденного решения для этой же задачи, т.е. будем использовать результаты исследования в [п. 6.5.2] - “наилучший” результат для задачи $\alpha_{2,5}$. Наименьшее значение найденного отношения $\alpha_2 = \frac{\lambda_2(a,b,c)}{\lambda(a,b,c)}$ в этой задаче равняется 0.874126 – т.е. теперь минимизация этой функции будет начинаться с этого же значения (начальные параметры системы, вектора цены и веса, те же, что и в случае, когда достигается это значения), и далее методом отжига компоненты вектора цен и весов будут изменяться для поиска наименьшего значения, чем уже найденное начальное 0.874126. В качестве параметров задачи возьмем следующие значения [см. п.6.5.2.]:

- вектор цен $c = (1, 9, 140, 255, 595)$,
- вектор весов $a = (2, 21, 157, 215, 396)$,
- вместимость рюкзака $b = 791 = 2 + 21 + 157 + 215 + 396$
- ограничение на сумму цен: $A = 1000 (1 + 9 + 140 + 225 + 595 = 1000)$

Запустим программу при таких параметрах (рис. 25):

```
Prices: 1, 9, 140, 255, 595,
Weights: 2, 21, 157, 215, 396,
Appr prices : 140, 595,
Appr weights: 157, 396,
MaxWeight: 791
0.874126 875 1001

Prices: 1, 9, 140, 255, 595,
Weights: 2, 21, 157, 215, 396,
Appr prices : 140, 595,
Appr weights: 157, 396,
MaxWeight: 791
0.874126 875 1001
```

Рис. 25. $\alpha_{2,5}$ с использованием лучшего уже найденного решения

Таким образом, используя параметры решения задачи по поиску $\alpha_{2,5}$ – не удалось получить меньшее значение отношения приближенного решения к точному, чем найденное ранее. В начале оптимизации получаемое решение уже “хорошее” и не сильно отличающееся от действительного, то метод не может позволить перейти при начальной высокой температуре в состояние хуже. Вероятность перехода к худшему решению при таком решении $\alpha_{2,5}$ уже маленькая, в следствие чего метод только перебирает значения параметров весов и цен. И только в случае, когда решение получится меньше, чем 0.874126 – удастся минимизировать отношение приближенного решения к точному $\alpha_2 = \frac{\lambda_2(a,b,c)}{\lambda(a,b,c)}$.

7.4. Исследование $\alpha_{2,5}$ с использованием наилучшего решения той же задачи с ограничением на сумму компонент цен, равным 1000000

Пусть теперь изменим ограничение на сумму компонент вектора цен на $A = 1000000$, но минимизировать $\alpha_2 = \frac{\lambda_2(a,b,c)}{\lambda(a,b,c)}$ так же будем, начиная с “наилучшего” достигнутого результата [п.6.5.2.]. При этом компоненты цен и весов будем генерировать так же случайным образом в отрезке от 0 до 1000000. Тогда качестве параметров задачи возьмем следующие значения (как и полученные в решении задачи по поиску $\alpha_{2,5} = 0.874126$), за исключением ограничения на сумму компонент цен:

- вектор цен $c = (1, 9, 140, 255, 595)$,
- вектор весов $a = (2, 21, 157, 215, 396)$,
- вместимость рюкзака $b = 791 = 2 + 21 + 157 + 215 + 396$
- ограничение на сумму цен: $A = 10^6$

Но т.к. сумма компонент цен равна 1000 ($1 + 9 + 140 + 225 + 595 = 1000$), то до запуска метода отжига цены изменятся функцией `change_prices_to_limit` соответственно, чтобы их сумма равнялась 10^6 . Поэтому, в отличие от пункта 9.3., на первой итерации значение $\alpha_2 = \frac{\lambda_2(a,b,c)}{\lambda(a,b,c)}$ уже не будет равняться 0.874126 (т.е. наименьшему полученному значению для данной задачи). В худшем случае $\alpha_{2,5} = 1$ на первой итерации. Запустим программу при таких параметрах (рис. 26):

```
Prices: 75205, 148890, 234141, 261169, 280595,  
Weights: 130, 227, 492, 445, 414,  
Appr prices : 234141, 75205,  
Appr weights: 492, 130,  
MaxWeight: 791  
0.894074 451230 504690
```

Рис. 26. $\alpha_{2,5}$ с использованием лучшего уже найденного решения и ограничением, равным 1000000

Таким образом, не удалось отыскать такой набор вектора цен и весов (prices и weights) на отрезке от 0 до 10^6 , чтобы $\alpha_2 = \frac{\lambda_2(a,b,c)}{\lambda(a,b,c)}$ было меньше, чем найденное в предыдущих случаях (или найденное минимальное в [п. 7.3.]). Так как размерность задачи и область поиска большие – решить поставленную задачу не получилось.

7.5. Исследование $\alpha_{2,5}$ с использованием наилучшего решения той же задачи с ограничением на сумму компонент цен, но без ограничения на вектор весов.

В главе 6 [см. 6.1-6.3.] было введено ограничение на сумму компонент вектора цен и весов, и для весов считалось, что введенное ограничение $B: \sum_{i=1}^n a_i x_i = B = b$ – т.е. совпадает с вместимостью рюкзака и методом отжига вместимость рюкзака не перебирается при минимизации функции точности k -оптимального решения $\alpha_2 = \frac{\lambda_2(a,b,c)}{\lambda(a,b,c)}$ по параметрам $a, c \in Z_+^n, b \in N$. Тогда состояние системы на каждой итерации метода отжига – были компоненты вектора цен и весов. Теперь, пусть $b \neq \sum_{i=1}^n a_i x_i$ и решаем поставленную задачу без ограничения на сумму компонент вектора весов. Начальные параметры задачи будем использовать, полученные в ходе запуска программной реализации с “лучшим” результатом для задачи по поиску $\alpha_{2,5}$ [см. 6.5.2.]. В качестве параметров задачи возьмем следующие:

- вектор цен $c = (1, 9, 140, 255, 595)$,
- вектор весов $a = (2, 21, 157, 215, 396)$,
- ограничение на сумму цен: $A = 1000$ ($1 + 9 + 140 + 225 + 595 = 1000$)

Запустим программу при таких параметрах (рис. 27):

```
Prices: 91, 103, 178, 272, 356,
Weights: 150, 367, 219, 409, 418,
Appr prices : 356, 91,
Appr weights: 418, 150,
MaxWeight: 791
0.8608 538 625
```

Рис. 27. $\alpha_{2,5}$ с использованием “лучшего” решения, $A = 1000$, без ограничения B

Таким образом, используя параметры решения задачи по поиску $\alpha_{2,5}$, удалось получить меньшее значение для минимизируемой функции $\alpha_2 = \frac{\lambda_2(a,b,c)}{\lambda(a,b,c)}$ по параметрам $a, c \in N_+^5, b \in N_+$, чем найденное ранее (наименьшее значение для точности k -оптимального решения получилось при введении ограничения на сумму компонент вектора цен, равное 0.874126). Но все же $\alpha_2 = \frac{\lambda_2(a,b,c)}{\lambda(a,b,c)} = 0.8608$ не принадлежит интервальной оценки для гарантированной точности k -оптимального решения из теории (статья [1]):

$$\frac{2^5 - 2^3}{2^5 - 1} = 0.774194 \leq \alpha_{2,5} \leq \frac{2^3 - 2}{2^3 - 1} = 0.857143.$$

7.6. Итог исследования $\alpha_{2,5}$ с использованием наилучшего решения той же задачи с ограничением на сумму компонент цен

Наилучшего результата по минимизации точности k -оптимального решения для целочисленной задачи о рюкзаке $\alpha_2 = \frac{\lambda_2(a,b,c)}{\lambda(a,b,c)}$ по поиску точной нижней грани $\alpha_{2,5}$ удалось добиться в п. [7.6], когда для поиска решения начальные параметры задачи равнялись итоговому состоянию системы для этой же задачи после применения метода имитации отжига и искали решение без обречения на сумму компонент вектора весов. Удалось минимизировать $\alpha_2 = \frac{\lambda_2(a,b,c)}{\lambda(a,b,c)}$ по параметрам $a, c \in Z_+^n, b \in N$ до значения 0.8602, но все же это значение не принадлежит интервальной оценки для гарантированной точности k -оптимального решения $\alpha_{2,5} \cdot \frac{2^5 - 2^3}{2^5 - 1} = 0.774194 \leq \alpha_{2,5} \leq \frac{2^3 - 2}{2^3 - 1} = 0.857143$. Далее, как и было описано, можно было бы используя результаты задачи по поиску $\alpha_{2,5}$ искать решение для задачи $\alpha_{2,6}$ и т.д., но попробуем минимизировать $\alpha_2 = \frac{\lambda_2(a,b,c)}{\lambda(a,b,c)}$, понизив размерность задачи (см. [Глава 8]).

Глава 8. Решение задачи с понижением размерности

8.1. Описание идеи

Так как не получилось добиться интервальной оценки из теории (статьи [1]) для $\alpha_{2,n}$ при $n \geq 5$ - понизим размерность задачи.

Напомним задачу: $\begin{cases} \max_x cx \\ ax \leq b \end{cases}$

$c = (c_1, c_2, \dots, c_n)^T$ – вектор цен предметов

$a = (a_1, a_2, \dots, a_n)^T$ – вектор весов предметов

$x = (x_1, x_2, \dots, x_n)^T$ – вектор, где x_i количество предметов i -ого типа.

$\alpha_k = \frac{\lambda_k(a,b,c)}{\lambda(a,b,c)}$ – точность k -оптимального значения задачи.

$\alpha_{k,n} = \inf_{a,b,c} \left\{ \left| \frac{\lambda_k(a,b,c)}{\lambda(a,b,c)} \right| : a, c \in Z_+^n, b \in N_+ \right\}$ – гарантированная точность k -оптимального

значения задачи ($\lambda(a,b,c)$ – оптимальное значение задачи, $\lambda_k(a,b,c)$ – k -оптимальное значение задачи).

Необходимо, используя различные методы для нахождения минимума функции,

найти: $\min_{a,b,c} \left| \frac{\lambda_k(a,b,c)}{\lambda(a,b,c)} \right|$, если $a, c \in Z_+^n, b \in N$. Для нахождения точного решения используется

метод динамического программирования, а для нахождения значения $\alpha_{k,n} = \inf_{a,b,c} \left| \frac{\lambda_k(a,b,c)}{\lambda(a,b,c)} \right|$

используется метод имитации отжига (simulated annealing).

Ранее программная реализация метод имитации отжига была следующей: на каждой итерации метод менял случайным образом на случайную величину компоненты вектора c (цен) и вектора a (весов) – и на новом наборе вычислялось значение $\alpha_k = \frac{\lambda_k(a,b,c)}{\lambda(a,b,c)}$. Если это значение получалось меньше, чем найденное ранее, то такие найденные компоненты вектора вектора c (цен), вектора a (весов) и само значение $\frac{\lambda_k(a,b,c)}{\lambda(a,b,c)}$ принимались в качестве ответа.

Теперь же запустим метод отжига для поиска весов – т.е. будем в процессе метода изменять только вектора a (весов). Тогда на каждой итерации если получили меньшее значение $\frac{\lambda_k(a,b,c)}{\lambda(a,b,c)}$, то сгенерированный вначале итерации случайным образом вектор c (весов) и само значение

$\alpha_k = \frac{\lambda_k(a,b,c)}{\lambda(a,b,c)}$ принимаем в качестве ответа. Далее снова изменяем случайным образом вектор a (весов) и сравниваем $\alpha_k = \frac{\lambda_k(a,b,c)}{\lambda(a,b,c)}$. После окончания метода отжига – имеем на выходе какое-то значение $\alpha_k = \frac{\lambda_k(a,b,c)}{\lambda(a,b,c)}$, которое было минимальное найденное в ходе запуска метода, и соответствующий ему вектор a (весов). Запустим теперь второй раз метод отжига, но уже изменяя вектор c (цен). По окончании работы двух методов отжига на выходе будем иметь найденное значение $\alpha_k = \frac{\lambda_k(a,b,c)}{\lambda(a,b,c)}$, и соответствующие ему вектора c (цен) и a (весов). Ограничение на сумму компонент вектора цен в реализации метода остается.

8.2. Описание программной реализации

Программная реализация для работы с поставленной задачи была продолжена на языке C++ в среде Microsoft Visual Studio 2019.

К классу `Backpack` добавился 1 метод `Method_anneal_2`. Метод принимает в качестве параметров `float _MaxWeight` (вместимость рюкзака), как и ранее написанный `Method_anneal` (см. п. [3.3.]). На выходе – `float` значение: `inf_ratio` – минимальное найденное значение $\frac{\lambda_k(a,b,c)}{\lambda(a,b,c)}$. Реализация метода отжига `anneal` представлена в [Приложение А (метод Отжига)]

8.3. Описание вывода программы

Во время работы программы теперь каждые 10000 итераций выводится промежуточный результат (рис. 28):

1. номер итерации,
2. достигнутая температура,
3. достигнутое минимизируемое значение $\frac{\lambda_k(a,b,c)}{\lambda(a,b,c)}$.

```

C:\Users\rozan\Desktop\Курсовая\Программа\Project3\64\Debug\Min_func.exe
2)
Count_iter: 10000
Temperat: 0.001/100
Inf: 1

Count_iter: 20000
Temperat: 0.0005/100
Inf: 0.927445

Count_iter: 30000
Temperat: 0.000333333/100
Inf: 0.927445

Count_iter: 40000
Temperat: 0.00025/100
Inf: 0.927445

Count_iter: 50000
Temperat: 0.0002/100
Inf: 0.927445

Count_iter: 60000
Temperat: 0.000166667/100
Inf: 0.927445

```

Рис. 28. Вывод промежуточных результатов

Далее, по окончании первого метода отжига (когда изменяли только вектора весов weights) программа выведет следующие данные (рис. 29):

1. найденный вектор цен;
2. вектор весов;
3. вектор цен при аппроксимации (т.е. когда две компоненты);
4. вектор весов при аппроксимации (т.е. когда две компоненты);
5. вместимость рюкзака
6. значения $\frac{\lambda_k(a,b,c)}{\lambda(a,b,c)}$, $\lambda_k(a,b,c)$ и $\lambda(a,b,c)$ на этом наборе.

```

C:\Users\rozan\Desktop\Курсовая\Программа\Project3\64\Debug\Min_func.exe
Count_iter: 960000
Temperat: 1.04167e-05/100
Inf: 0.875

Count_iter: 970000
Temperat: 1.03093e-05/100
Inf: 0.868056

Count_iter: 980000
Temperat: 1.02041e-05/100
Inf: 0.868056

Count_iter: 990000
Temperat: 1.0101e-05/100
Inf: 0.868056

Count_iter: 1000000
Temperat: 1e-05/100
Inf: 0.868056

Count_iter: 1000001
Temperat: 9.99999e-06/100
Inf: 0.868056

Prices: 1, 9, 140, 255, 595,
Weights: 3, 51, 263, 449, 790,
Appr prices : 140, 595,
Appr weights: 263, 790,
MaxWeight: 1556
0.868056 875 1000

```

Рис. 29. Вывод промежуточных результатов

Далее запускается второй метод отжига для поиска вектора цен (prices). Вывод программы аналогичен, как и описанный ранее. По окончании работы программы будут выведены те же данные, что и после работы метода отжига на поиске весов, но уже и с найденным оптимальным набором цен (т.е. найденный вектор цен, найденный вектор весов, вектор цен при аппроксимации, вектор весов при аппроксимации, вместимость рюкзака \max_weight , $\frac{\lambda_k(a,b,c)}{\lambda(a,b,c)}$, $\lambda_k(a,b,c)$ и $\lambda(a,b,c)$ на этом наборе

8.4. Исследование $\alpha_{2,5}$

Ранее не получилось добиться теоретических интервальных оценок для гарантированной точности k-оптимального решения (при $n \geq 5$) из статьи А.Ю. Чиркова, В.Н. Шевченко [1]. Напомним, что из статьи $\alpha_{2,5}$ лежит в следующем интервальном отрезке:

$$\frac{2^5 - 2^3}{2^5 - 1} = 0.774194 \leq \alpha_{2,5} \leq \frac{2^3 - 2}{2^3 - 1} = 0.857143.$$

Запустим программу с введенным ограничением на сумму компонент цен, и измененным поиском минимального значения отношения приближенного решения к точному $\frac{\lambda_2(a,b,c)}{\lambda(a,b,c)}$ при $k = 2$ (т.е. поочередно запускаются два метода отжига: для поиска весов, и далее для поиска цен, в результате чего получаем минимальное найденное отношение приближенного решения к точному, и соответствующие ему вектора цен и весов).

Для минимизации $\alpha_2 = \frac{\lambda_2(a,b,c)}{\lambda(a,b,c)}$ в задаче по поиску $\alpha_{2,5}$ используем параметры уже найденного ранее решения из п. [6.5.2.], где функцию точности k-оптимального решения удалось минимизировать до значения, равного 0.874126. Т.е. теперь минимизация $\alpha_2 = \frac{\lambda_2(a,b,c)}{\lambda(a,b,c)}$ будет начинаться с этого же значения (начальные вектора цен и веса выберем соответствующие этому значению α_2), и далее методом отжига компоненты вектора цен и весов будут изменяться для поиска наименьшего значения, чем уже найденное. В качестве параметров задачи возьмем следующие значения [п. 6.5.2.]:

- вектор цен Prices = (1, 9, 140, 255, 595),
- вектор весов Weights = (2, 21, 157, 215, 396),
- вместимость рюкзака $MaxWeight = 791 = 2 + 21 + 157 + 215 + 396$
- Ограничение на сумму цен: $1000 (1 + 9 + 140 + 225 + 595 = 1000)$

Ограничения на сумму компонент вектора цен и весов введены: $A = 1000$, $B = \text{MaxWeight} = \sum_{i=1}^n a_i x_i$. Исследуем полученный результат. Запустим программу при таких параметрах (рис. 30):

```
Prices: 1, 9, 140, 255, 595,
Weights: 24, 29, 214, 373, 674,
Appr prices : 140, 595,
Appr weights: 214, 674,
MaxWeight: 1314
0.875 875 1000
```

Рис. 30. Полученные программой значения $\alpha_{2,5}$

Получили $\alpha_2 = \frac{\lambda_2(a,b,c)}{\lambda(a,b,c)} = 0.875 > 0.874126$. Но в решение каждая компонента вектора цен (и веса соответственно) входит один раз. Имея ограничение на сумму компонент вектора цен 1000, получили решение поставленной задачи о рюкзаке $\lambda(a,b,c) = 1000$. Каждый предмет взят в рюкзак один раз, и вместимость рюкзака так же равна сумме всех компонент вектор весов ($1314 = 24 + 29 + 214 + 373 + 674$). Из теории следует, что если такой набор достигается, то у такой задачи можно найти еще меньшее значение, чем на этом же наборе. Запустим программу еще раз (рис. 31):

```
C:\Users\rozan\Desktop\Кыпован\Программа\Project3\64\Debug\Min_func.exe
Count_iter: 960000
Temperat: 1.04167e-05/100
Inf: 0.875

Count_iter: 970000
Temperat: 1.03093e-05/100
Inf: 0.868056

Count_iter: 980000
Temperat: 1.02041e-05/100
Inf: 0.868056

Count_iter: 990000
Temperat: 1.0101e-05/100
Inf: 0.868056

Count_iter: 1000000
Temperat: 1e-05/100
Inf: 0.868056

Count_iter: 1000001
Temperat: 9.99999e-06/100
Inf: 0.868056

Prices: 1, 9, 140, 255, 595,
Weights: 3, 51, 263, 449, 790,
Appr prices : 140, 595,
Appr weights: 263, 790,
MaxWeight: 1556
0.868056 875 1000
```

Рис. 31. Полученные программой значения $\alpha_{2,5}$

По завершении метода отжига получили значение $\alpha_{2,5} = 0.868056$ – т.е. начиная изменять параметры вектора цен и весов, на которых достигалось значение $\frac{\lambda_2(a,b,c)}{\lambda(a,b,c)} = 0.874126$,

перешли к набору $\frac{\lambda_2(a,b,c)}{\lambda(a,b,c)} = 0.875$, на котором решение задачи о рюкзаке - это набор из единиц, и в итоге, увеличив количество итераций, получили меньшее значение искомого отношения. Уменьшить решение получилось, т.к. $\lambda(a,b,c) = 1008$ – т.е. точное решение задачи о рюкзаке стало больше, чем в случае, когда в качестве отношения получили 0.875, но при этом решение приближенного решения $\lambda_2(a,b,c)$ осталось равным 875. Проанализируем полученное решение (таблица 23):

Вектор цен (Prices)	(1, 9, 140, 255, 595),
Вектор весов (Weights)	(3, 51, 263, 449, 790)
Максимальная вместимость рюкзака	1556
Вектор цен для приближенного решения из набора k предметов (Appr prices)	(140, 595)
Вектор весов для приближенного решения из набора k предметов (Appr weights)	(263, 790)
Точное решение задачи	1008
Приближенное решение	875
Отношение приближенного решения к точному:	0.86056

Таблица 23. Результаты полученного решения для $\alpha_{2,3}$

Максимальная вместимость рюкзака $\max_weight = \sum weight[i] = 3 + 51 + 263 + 449 + 790 = 1556$

Точное решение задачи - оно достигается при следующем наборе предметов: предмет ценой 1 взят 18 раз, и предметы ценой 140, 255, 595 взяты по одному разу ($1 \cdot 18 + 1 \cdot 140 + 1 \cdot 255 + 1 \cdot 595 = 1008$)

Приближенное решение $\lambda_2(a,b,c) = 875$ – оно достигается при следующем наборе предметов: предмет ценной 140 взят два раза и предмет ценой 595 взят один раз ($140 \cdot 2 + 595 \cdot 1 = 875$)

И тогда получаем, что отношение приближенного решения к точному:

$$\frac{\lambda_2(a,b,c)}{\lambda(a,b,c)} = \frac{875}{1008} = 0.868056.$$

Таким образом, полученное наименьшее значение из ранее найденных $\frac{\lambda_2(a,b,c)}{\lambda(a,b,c)} = 0.868056 > 0.857143$ (т.е. больше, чем теоретическая верхняя оценка из статьи на 0.003417).

Полученное минимизированное значение точности k -оптимального решения α_2 , равное 0.868056, близко к значению из [п. 7.5.], когда введено было только ограничение вектора цен, и поиск минимума функции α_2 был по параметрам $a, c \in Z_+^n, b \in N$ (в п. 7.5. удалось минимизировать до значения 0.8605). Но все же не удалось минимизировать о теоретической интервальной оценки для гарантированной точности k -оптимального решения.

Заключение

В данной работе был использован метод имитации отжига для поиска минимума неявно заданной функции. Этот алгоритм является одним из многих эвристических алгоритмов. Если необходимо решить задачу из класса NP-полных, тогда именно эвристические алгоритмы предоставляют наиболее приемлемое решение из всех доступных вариантов, но, возможно, с некоторой неточностью.

Метод имитации отжига был использован для минимизации неявно заданной функции α_k вида: $\frac{\lambda_k(a,b,c)}{\lambda(a,b,c)}$, где $\lambda(a,b,c)$ – оптимальное значение задачи, $\lambda_k(a,b,c)$ – k-оптимальное значение задачи. Т.е. решалась задача вида: $\min_{a,b,c} \left| \frac{\lambda_k(a,b,c)}{\lambda(a,b,c)} \right|$ $a, c \in Z_+^n, b \in N$.

В ходе работы были получены следующие основные результаты:

1. Были изучены эмпирические алгоритмы, включая метод имитации отжига.
2. Была выполнена программная реализация решения задачи о целочисленном рюкзаке: поиск точного решения и приближенного. Так же в ходе этой же программной реализации был реализован метод имитации отжига с различными модификациями.
3. Проведен ряд экспериментов для сравнения модификаций метода отжига для поиска минимума отношения приближенного решения к точному для задачи о целочисленном рюкзаке.
4. Выполнен анализ полученных результатов

По результатам проведённых исследований модификаций метода отжига было выявлено, что ограничение на сумму компонент вектора цен и весов помогает найти $\min_{a,b,c} \alpha_{2,4}$. Данную задачу удалось решить и минимизировать функцию $\alpha_{2,4}$ до теоретического интервала из статьи А.Ю. Чиркова, В.Н. Шевченко [1]. В ходе запуска программы получили $\frac{\lambda_k(a,b,c)}{\lambda(a,b,c)} = 0.8526$, когда в статье доказано, что $\alpha_{2,4} \in [0.8; 0.857143]$. Так же с введенным ограничением удалось и уменьшить значение $\alpha_{2,5}$ относительно ранее найденного. Но полученное минимизируемое значение получилось больше, чем верхняя теоретическая оценка.

Далее, понизив размерность задачи разделением метода отжига на два этапа, удалось еще уменьшить значение искомого минимума отношения $\frac{\lambda_k(a,b,c)}{\lambda(a,b,c)}$. Найденное значение близко к верхней теоретической оценке, равной 0.857143. При понижении размерности, удалось

получить $\frac{\lambda_k(a,b,c)}{\lambda(a,b,c)} = 0.86056$, что на 0.003417 больше, чем верхняя оценка из теории. Более того, с понижением размерности удалось на задаче о рюкзаке найти решение, состоящее из набора единиц. Итого: на наборе задачи $\alpha_{2,n}$ были получены следующие результаты в сравнении с теоретическими оценками из статьи А.Ю. Чиркова, В.Н. Шевченко [1] (Таблица 24):

$\alpha_{2,n}$	$\alpha_{2,4}$	$\alpha_{2,5}$	$\alpha_{2,6}$	$\alpha_{2,7}$	$\alpha_{2,8}$	$\alpha_{2,9}$	$\alpha_{2,10}$
нижняя оценка $\alpha_{2,n}$	0.8	0.774194	0.761905	0.755906	0.752941	0.751468	0.750733
в программной реализации (без модификации): $\min_{a,c \in Z_+^n, b \in N} \left \frac{\lambda_2(a, b, c)}{\lambda(a, b, c)} \right $	0.86602	0.89473.	0.9	0.909091	0.916667	0.925926	0.941176
в программной реализации с использованием предыдущих результатов: $\min_{a,c \in Z_+^n, b \in N} \left \frac{\lambda_2(a, b, c)}{\lambda(a, b, c)} \right $	0.8526	0.8608	-	-	-	-	-
В программной реализации с модиф. методом отжига: $\min_{a,c \in Z_+^n, b \in N} \left \frac{\lambda_2(a, b, c)}{\lambda(a, b, c)} \right $	-	0.868056	-	-	-	-	-
верхняя оценка $\alpha_{2,n}$	0.857143						

Таблица 24. результаты полученного решения для $\alpha_{2,n}$

Данный подход, выбранный для минимизации значения неявно заданной функции, оказался не слишком успешным. В процессе применения выбранного подхода

не удалось достигнуть желаемого результата. Возможно, в дальнейшем при изучении данной темы следовало бы пересмотреть выбранный подход и найти более эффективный способ снижения размерности поставленной задачи за счет других модификации метода отжига, приводящих к упрощению расчетов для данной задачи. Методом имитаций отжига не удалось минимизировать отношение приближенного решения к точному, именно, из-за роста размерности задачи при увеличении числа n . В поставленных задачах поиска минимума функции $\alpha_{k,n}$ с $n > 5$ размерность задачи сильно увеличивается и не удастся минимизировать ее значение до нужных теоретических пределов. Однако в задачах по поиску минимума функции $\alpha_{2,4}$ и $\alpha_{2,5}$ все же удалось добиться необходимых ее минимальных значений, удовлетворяющим теоретическим интервальным оценка из статьи А.Ю. Чиркова, В.Н. Шевченко [1].

Список литературы

1. Статья Чиркова А.Ю., Шевченко В.Н. “О приближении оптимального решения целочисленной задачи о ранце оптимальными решениями целочисленной задачи о ранце с ограничением на мощность”.
2. Лопатин А.С. “Метод отжига”,
3. Носкова Е.Э. “Метод имитации отжига для задачи о расписании с параллельными процессорами”,
4. Алексеев В.Б., Носов В.А. “NP-полные задачи и их полиномиальные варианты”,
5. Громкович Ю., Мельников Б.Ф. “Алгоритмизация труднорешаемых задач. Часть I. простые примеры и простые эвристики”,
6. Косовская Т.М. “О классах сложности алгоритмов”,
7. Землянский А.В., Штыков Г.А. “Программная реализация метода имитации отжига и анализ его эффективности”,
8. Босов А. А., Горбова А. В., Халипова Н. В. “Обоснование эвристического алгоритма в задаче о ранце”,
9. Глушань В. М. “Метод имитации отжига”
10. Пантелеев А.В., Дмитраков И.Ф. “Сравнительный анализ эффективности метода имитации отжига для поиска глобального экстремума функций многих переменных”

Приложение А (метод Отжига)

Реализация метода Отжига:

```
float Method_anneal(float _MaxWeight)
{
    srand(time(NULL));

    int z = 0;
    float sum_of_elems = 0;
    float inf_ratio = 1;
    float tmp_inf_ratio = 1;
    int tmp_MaxWeight = _MaxWeight;
    int price_approx = 0;
    int price_dynamic_programming = 0;

    size_t i = 0;
    size_t j = 0;
    size_t k = 0;

    std::vector<int> tmp_prices;
    std::vector<int> tmp_weights;
    size_t tmp_elements = 0;

    tmp_prices = prices;
    tmp_weights = weights;
    tmp_elements = count_elements;

    std::vector<int> res_prices;
    std::vector<int> res_weights;
    double res_price_dynamic_programming = 0;
    double res_price_approx = 0;
    int res_max_weight = 0;

    std::vector<float> temperature;
    temperature.push_back(10000000);
    float temperature_min = 10;

    while (temperature[i] > temperature_min)
```

```

{
    BackpackItem B(tmp_prices, tmp_weights, tmp_elements);

    int tmp_max_weight = 0;
    for (int i = 0; i < tmp_elements; i++)
    {
        tmp_max_weight = tmp_max_weight + B.weights[i];
        tmp_MaxWeight = tmp_max_weight;
    }

    price_approx = B.Approximate_kZero(tmp_max_weight);
    price_dynamic_programming=B.Method_dynamic_programming(tmp_max_weight);
}

```

```

if(price_dynamic_programming != 0)
{
    tmp_inf_ratio = double(price_approx) /
    double (price_dynamic_programming);
}

if (tmp_inf_ratio <= inf_ratio)
{
    inf_ratio = tmp_inf_ratio;
    res_price_approx = price_approx;
    res_price_dynamic_programming = price_dynamic_programming;

    res_prices = tmp_prices;
    res_weights = tmp_weights;

    res_max_weight = tmp_MaxWeight;
}

for (int k = 0; k < count_elements; k++)
{
if (rand() % 2)
{
    tmp_prices[k] = (tmp_prices[k] + pow(-1, rand() % 2) * (rand() %
    10 + 1));
}

if (tmp_prices[k] <= 0)
{
    tmp_prices[k] = tmp_prices[k] * (-1) + 1;
}

    tmp_prices[k] = tmp_prices[k] % 200 + 1;
}

for (int k = 0; k < count_elements; k++)
{
if (rand() % 2)
{

```

```

        tmp_weights[k] = (tmp_weights[k] + pow(-1, rand() % 2) * (rand()
        % 5 + 1));
    }

    if (tmp_weights[k] <= 0)
    {
        tmp_weights[k] = tmp_weights[k] * (-1) + 1;
    }

    tmp_weights[k] = tmp_weights[k] % 200 + 1;
}

change_prices_to_spec_value_less_1(tmp_prices, tmp_weights);
temperature.push_back(temperature[0] * 0.1 / (i + 1));
i++;
}
}

```

Приложение В (ограничения на сумму компонент вектора цен и веса)

Ограничение на сумму компонент вектора цен:

```
int change_prices_to_limit(std::vector<int>& prices)
{
    int tmp_sum_prices = 0;
    for (int i = 0; i < prices.size(); i++)
    {
        tmp_sum_prices += prices[i];
    }

    int tmp_changed_value_price = 0;
    while (tmp_sum_prices != limitation_sum_prices)
    {
        int tmp_rand_index = rand() % prices.size();
        tmp_changed_value_price = rand() % limitation_sum_prices;

        if (tmp_sum_prices < limitation_sum_prices)
        {
            prices[tmp_rand_index] += tmp_changed_value_price;
            tmp_sum_prices += tmp_changed_value_price;
        }
        if (tmp_sum_prices > limitation_sum_prices)
        {
            if (prices[tmp_rand_index] > tmp_changed_value_price)
            {
                prices[tmp_rand_index] -= tmp_changed_value_price;
                tmp_sum_prices -= tmp_changed_value_price;
            }
        }
    }

    return tmp_sum_prices;
}
```

Ограничение на сумму компонент вектора весов:

```
int change_weights_to_limit(std::vector<int>& weights)
{
    int tmp_sum_weights = 0;
    for (int i = 0; i < weights.size(); i++)
    {
        tmp_sum_weights += weights[i];
    }

    int tmp_changed_value_weight = 0;
    while (tmp_sum_weights != limitation_sum_weights)
    {
        int tmp_rand_index = rand() % weights.size();
        tmp_changed_value_weight = rand() % limitation_sum_weights;

        if (tmp_sum_weights < limitation_sum_weights)
        {
            weights[tmp_rand_index] += tmp_changed_value_weight;
            tmp_sum_weights += tmp_changed_value_weight;
        }
        if (tmp_sum_weights > limitation_sum_weights)
        {
            if (weights[tmp_rand_index] > tmp_changed_value_weight)
            {
                weights[tmp_rand_index] -= tmp_changed_value_weight;
                tmp_sum_weights -= tmp_changed_value_weight;
            }
        }
    }

    return tmp_sum_weights;
}
```

Приложение С (метод отжига с пониженной размерностью)

```
float Method_anneal_2(float _MaxWeight)
{
    int count_iter = 0;
    int z = 0;
    float sum_of_elems = 0;
    float inf_ratio = 1;
    float tmp_inf_ratio = 1;
    int tmp_MaxWeight = _MaxWeight;
    int price_approx = 0;
    int price_dynamic_programming = 0;

    size_t i = 0;
    size_t j = 0;
    size_t k = 0;

    std::vector<int> tmp_prices;
    std::vector<int> tmp_weights;
    size_t tmp_elements = 0;

    tmp_prices = prices;
    tmp_weights = weights;
    tmp_elements = count_elements;

    std::vector<int> res_prices;
    res_prices = prices;
    std::vector<int> res_weights;
    double res_dynamic_programming = 0;
    double res_approx = 0;
    int res_max_weight = 0;
    std::vector<int> res_appr_prices;
    std::vector<int> res_appr_weights;
    std::vector<int> tmp_appr_prices;
    std::vector<int> tmp_appr_weights;

    //179900000, 1000
    // 100, 0.00001
```

```

std::vector<double> temperature1;
temperature1.push_back(100);
float temperature_min = 0.00001;

/*Запуск метода отжига при изменении ТОЛЬКО веса*/
while (temperature1[i] > temperature_min)
{
    BackpackItem B(tmp_prices, tmp_weights, tmp_elements);

    /*Вычисление MaxWeight для данной задачи*/
    int tmp_max_weight = 0;
    for (int i = 0; i < tmp_elements; i++)
    {
        tmp_max_weight = tmp_max_weight + B.weights[i];
        tmp_MaxWeight = tmp_max_weight;
    }

    /*Запуск приближенного решения и точного*/
    price_approx = B.Approximate_to_k_elements(tmp_max_weight, tmp_appr_prices,
    tmp_appr_weights);
    price_dynamic_programming = B.Method_dynamic_programming(tmp_max_weight);
    /*Вычисление отношения прибиленного решения к точному*/
    if (price_dynamic_programming != 0)
    {
        //полученное отношение
        tmp_inf_ratio = double(price_approx) / double(price_dynamic_programming);
    }

    if (tmp_inf_ratio <= inf_ratio)
    {
        inf_ratio = tmp_inf_ratio;

        res_approx = price_approx;
        res_dynamic_programming = price_dynamic_programming;

        res_prices = tmp_prices;
        res_weights = tmp_weights;
    }
}

```



```

    res_appr_prices = tmp_appr_prices;
    res_appr_weights = tmp_appr_weights;

    res_max_weight = tmp_MaxWeight;
}
else
{
    float probability = exp(-(tmp_inf_ratio - inf_ratio) / temperature1[i]);
    float gen_value = float(rand() % 10000) / 10000;
    if (gen_value < probability)
    {
        inf_ratio = tmp_inf_ratio;

        res_approx = price_approx;
        res_dynamic_programming = price_dynamic_programming;

        res_prices = tmp_prices;
        res_weights = tmp_weights;

        res_appr_prices = tmp_appr_prices;
        res_appr_weights = tmp_appr_weights;

        res_max_weight = tmp_MaxWeight;
    }
    else
    {
        //go next
    }
}

/*Изменение состояния (BECA)*/
for (int k = 0; k < count_elements; k++)
{
    if (rand() % 2)
    {
        tmp_weights[k] = (tmp_weights[k] + pow(-1, rand() % 2) * (rand() %
        500 + 1));
    }

    if (tmp_weights[k] <= 0)

```

```

        {
            tmp_weights[k] = tmp_weights[k] * (-1) + 1;
        }

        tmp_weights[k] = tmp_weights[k] % 500 + 1;
    }

    change_weights_to_limit(tmp_weights);

    temperature1.push_back(temperature1[0] * 0.1 / (i + 1));
    i++;
    count_iter++;

    if (count_iter % 5000 == 0)
    {
        std::cout << "Count_iter: " << count_iter << std::endl << "Temperat: " <<
            temperature1[i] << "/" << temperature1[0] << std::endl << "Inf: " <<
            inf_ratio << std::endl << "-----" << std::endl;
    }
}

sort_price_with_weight(res_prices, res_weights);

std::cout << "Prices: ";
for (int i = 0; i < count_elements; i++)
{
    std::cout << res_prices[i] << ", ";
}
std::cout << std::endl << "Weights: ";
for (int i = 0; i < count_elements; i++)
{
    std::cout << res_weights[i] << ", ";
}
std::cout << std::endl << "Appr prices : ";
for (int i = 0; i < res_appr_prices.size(); i++)
{
    std::cout << res_appr_prices[i] << ", ";
}
std::cout << std::endl << "Appr weights: ";

```

```

for (int i = 0; i < res_appr_prices.size(); i++)
{
    std::cout << res_appr_weights[i] << ", ";
}

std::cout << std::endl << "MaxWeight: " << res_max_weight << std::endl <<
inf_ratio << " " << res_approx << " " << res_dynamic_programming << std::endl <<
std::endl;

i = 0;
std::vector<double> temperature_2;
temperature_2.push_back(100);
while (temperature_2[i] > temperature_min)
{
    BackpackItem B(tmp_prices, tmp_weights, tmp_elements);

    price_approx = B.Approximate_to_k_elements(res_max_weight, tmp_appr_prices,
    tmp_appr_weights);
    price_dynamic_programming = B.Method_dynamic_programming(res_max_weight);
    if (price_dynamic_programming != 0)
    {
        tmp_inf_ratio = double(price_approx) / double(price_dynamic_programming);
    }

    if (tmp_inf_ratio <= inf_ratio)
    {
        inf_ratio = tmp_inf_ratio;

        res_approx = price_approx;
        res_dynamic_programming = price_dynamic_programming;

        res_prices = tmp_prices;
        res_weights = tmp_weights;

        res_appr_prices = tmp_appr_prices;
        res_appr_weights = tmp_appr_weights;

        res_max_weight = tmp_MaxWeight;
    }
    else
    {

```

```

float probability = exp(-(tmp_inf_ratio - inf_ratio) / temperature1[i]);
float gen_value = float(rand() % 10000) / 10000;
if (gen_value < probability)
{
    inf_ratio = tmp_inf_ratio;

    res_approx = price_approx;
    res_dynamic_programming = price_dynamic_programming;

    res_prices = tmp_prices;
    res_weights = tmp_weights;

    res_appr_prices = tmp_appr_prices;
    res_appr_weights = tmp_appr_weights;

    res_max_weight = tmp_MaxWeight;
}
else
{
    //go next
}
}

```

/*Изменение состояния (ЦЕНЫ)*/

```

for (int k = 0; k < count_elements; k++)
{
    if (rand() % 2)
    {
        tmp_prices[k] = (tmp_prices[k] + pow(-1, rand() % 2) * (rand() %
        500 + 1));
    }

    if (tmp_prices[k] <= 0)
    {
        tmp_prices[k] = tmp_prices[k] * (-1) + 1;
    }

    tmp_prices[k] = tmp_prices[k] % 500 + 1;
}

```

```

    }

    change_prices_to_limit(tmp_prices);

    temperature_2.push_back(temperature_2[0] * 0.1 / (i + 1));
    i++;
    count_iter++;

    if (count_iter % 5000 == 0)
    {
        std::cout << "Count_iter: " << count_iter << std::endl << "Temperat: " <<
        temperature_2[i] << "/" << temperature_2[0] << std::endl << "Inf: " <<
        inf_ratio << std::endl << "-----" << std::endl;
    }
}

sort_price_with_weight(res_prices, res_weights);

std::cout << "Prices: ";
for (int i = 0; i < count_elements; i++)
{
    std::cout << res_prices[i] << ", ";
}
std::cout << std::endl << "Weights: ";
for (int i = 0; i < count_elements; i++)
{
    std::cout << res_weights[i] << ", ";
}
std::cout << std::endl << "Appr prices : ";
for (int i = 0; i < res_appr_prices.size(); i++)
{
    std::cout << res_appr_prices[i] << ", ";
}
std::cout << std::endl << "Appr weights: ";
for (int i = 0; i < res_appr_prices.size(); i++)
{
    std::cout << res_appr_weights[i] << ", ";
}
std::cout << std::endl << "MaxWeight: " << res_max_weight << std::endl <<
inf_ratio << " " << res_approx << " " << res_dynamic_programming;

```

```
    return inf_ratio;  
}
```