**Оглавление**

[Список принятых сокращений 2](#_Toc515499932)

[Введение 3](#_Toc515499933)

[1. Постановка задачи о рюкзаке 5](#_Toc515499934)

[1.1 Описание 5](#_Toc515499935)

[1.2 Варианты постановки задачи о рюкзаке 5](#_Toc515499936)

[1.3 Постановка задачи о 0-1 рюкзаке 6](#_Toc515499937)

[1.4 Постановка задачи о неограниченном рюкзаке 6](#_Toc515499940)

[1.5 Свойства задачи о неограниченном рюкзаке 7](#_Toc515499943)

[2. Области применения задачи о рюкзаке 9](#_Toc515499946)

[2.1 Изучение в математике 9](#_Toc515499947)

[2.2 Изучение в экономике 9](#_Toc515499948)

[2.3 Изучение в криптографии 10](#_Toc515499949)

[2.4 Изучение в информатике 12](#_Toc515499950)

[3. Алгоритмы решения задачи о рюкзаке 13](#_Toc515499951)

[3.1 Точные алгоритмы решения ЗОР 13](#_Toc515499952)

[3.2 Точные алгоритмы решения ЗОНР 15](#_Toc515499955)

[3.3 Генетический алгоритм 20](#_Toc515499959)

[4. Генетический алгоритм 22](#_Toc515499960)

[4.1 Интерпретация задачи о ранце и операторов генетического алгоритма c помощью понятий популяционной генетики 22](#_Toc515499961)

[4.2 Представление генотипа 23](#_Toc515499962)

[4.3 Операторы создания начальной популяции 24](#_Toc515499963)

[4.4 Операторы скрещивания 24](#_Toc515499966)

[4.5 Операторы мутации 25](#_Toc515499970)

[4.6 Оператор обработки ограничений 26](#_Toc515499975)

[4.7 Операторы селекции 30](#_Toc515499977)

[4.8 Оператор восстановления 32](#_Toc515499980)

[5. Классы тестовых задач 33](#_Toc515499981)

[5.1 Задачи без корреляции 33](#_Toc515499982)

[5.2 Задачи с слабой корреляцией 33](#_Toc515499983)

[5.3 Задачи с сильной корреляцией 33](#_Toc515499984)

[5.4 Задачи с подсуммами 34](#_Toc515499985)

[6. Программный модуль 36](#_Toc515499986)

[6.1 Структура программного модуля 36](#_Toc515499987)

[7. Эксперименты 42](#_Toc515499995)

[7.1 Эксперимент по поиску эффективных комбинаций параметров ГА 42](#_Toc515499996)

[7.2 Эксперимент по исследованию производительности и качества алгоритмов 46](#_Toc515499999)

[Заключение 53](#_Toc515500002)

[Литература 55](#_Toc515500003)

[Приложение 56](#_Toc515500004)

# **Список принятых сокращений**

ЗОР – задача о 0-1 рюкзаке

ЗОНР – задача о неограниченном рюкзаке

ДП – динамическое программирование

МВИГ – метод ветвей и границ

ЭДП – алгоритм эффективного динамического программирования

ГА – генетический алгоритм

ПМ – программный модуль

# **Введение**

В качестве *объекта исследования* в данной работе выступает задача о рюкзаке, которая относится к числу широко известных задач комбинаторной оптимизации. Она часто используется на практике для решения таких проблем, как: погрузка груза, бюджетирование капитала, планирование проектов и выбор портфельных инвестиций.

В качестве *предметов исследования* в работе выступают NP-трудные задачи о рюкзаке двух типов: задача о 0-1 ранце (**ЗОР**) и задача о неограниченном ранце (**ЗОНР**).

Рассматривается несколько подходов к решению:

Относительно ЗОР:

1. Табличный и рекуррентный методы динамического программирования
2. Метод ветвей и границ с обходом в глубину/ширину

Относительно ЗОНР:

1. Классический метод динамического программирования
2. Метод эффективного динамического программирования
3. Метод ветвей и границ с обходом в глубину/ширину, использующий верхнюю границу по трем предметам с лучшим соотношением цены и веса ()

Также рассматривается генетический алгоритм с различной комбинацией параметров, применимый к обоим типам задач.

Задача о рюкзаке имеет особенности в виде корреляций между стоимостью и весом, поэтому эксперименты в данной работе проводятся для 4 классов тестовых задач:

* задачи без корреляции,
* задачи с слабой корреляцией,
* задачи с сильной корреляцией,
* задачи с подсуммами.

*Цель исследовательской работы* – подбор лучшего среди исследуемых алгоритма решения для каждого класса рассматриваемых типов задач о ранце.

*Методы исследования* представляют собой следующий набор действий:

* Разработка точных алгоритмов для решения ЗОР.
* Разработка точных алгоритмов для решения ЗОНР, а также исследование специфичных свойств для данной задачи, позволяющих сократить пространство поиска, ускорить нахождение оптимального решения.
* Разработка генетического алгоритма с различными наборами параметров.
* Анализ поведения ГА при различных комбинациях параметров применительно к каждому классу рассматриваемых типов задач для выявления комбинаций, дающих наиболее выгодное решение.
* Анализ производительности и качества описанных алгоритмов, чтобы подобрать к каждому классу рассматриваемых типов задач о рюкзаке наилучший подход к решению.

# **1. Постановка задачи о рюкзаке**

## 1.1 Описание

Задача о ранце (рюкзаке) — оптимизационная задача укладки максимального числа ценных предметов в рюкзак при условии, что общая его емкость ограничена. Она является актуальной и достаточно востребованной с точки зрения приложения в реальной жизни. Задача о загрузке и её модификации возникают в таких областях, как: прикладная математика, экономика, криптография, генетика и логистика (например, для нахождения оптимальной загрузки транспорта или склада).

## 1.2 Варианты постановки задачи о рюкзаке[[1]](#footnote-1)

Существует множество разновидностей задачи о ранце, отличия заключаются в условиях, наложенных на рюкзак, предметы или их выбор.

1. **Рюкзак 0-1** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) 0-1 Knapsack Problem): не более одного экземпляра каждого предмета.
2. **Ограниченный рюкзак** (англ. Bounded Knapsack Problem) – обобщение классической задачи, когда любой предмет может быть взят некоторое количество раз.
3. **Неограниченный рюкзак** (целочисленный рюкзак) ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Unbounded Knapsack Problem (integer knapsack)): произвольное количество экземпляров каждого предмета.
4. **Непрерывный рюкзак** (англ. Continuous Knapsack Problem) - вариант задачи, в котором возможно брать любою дробную часть от предмета, при этом удельная стоимость сохраняется.
5. **Рюкзак с мультивыбором** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Multiple-choice Knapsack Problem): предметы разделены на группы, и из каждой группы требуется выбрать только один предмет.
6. **Мультипликативный рюкзак** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Multiple Knapsack Problem): есть несколько рюкзаков, каждый со своим максимальным весом. Каждый предмет можно положить в любой рюкзак или оставить.
7. **Многомерный рюкзак** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Multy-dimensional knapsack problem): вместо веса дано несколько разных ресурсов (например, вес, объём и время укладки). Каждый предмет тратит заданное количество каждого ресурса. Надо выбрать подмножество предметов так, чтобы общие затраты каждого ресурса не превышали максимума по этому ресурсу, и при этом общая ценность предметов была максимальна.

В данной работе мы будем рассматривать задачи 1 и 3.

## 1.3 Постановка задачи о 0-1 рюкзаке

### 1.3.1 Словесная

Пусть дан набор предметов, каждый из которых имеет два параметра – вес и ценность. Дан рюкзак c заданным значением емкости. Задача заключается в том, чтобы заполнить рюкзак предметами, получить максимальную суммарную ценность, соблюдая при этом весовое ограничение.

### 1.3.2 Математическая

Пусть дано n предметов. Для каждого i-го предмета задан вес *> 0* и ценность  *> 0, i =1,..,n.* Задано ограничение на максимальный вес рюкзака . Каждый может принимать только одно из двух значений: *xi = 1*, если *i-*й предмет попадает в рюкзак, или*,* в противном случае. Требуется выбрать из заданного множества предметов набор с максимальной суммарной ценностью при одновременном соблюдении ограничения на суммарный вес найденного набора .

|  |  |  |
| --- | --- | --- |
|  |  | **(1.3.1)** |

## 1.4 Постановка задачи о неограниченном рюкзаке

### 1.4.1 Словесная

Пусть дан набор предметов, каждый из которых имеет два параметра – вес и ценность. Каждый предмет можно класть неограниченное число раз. Дан рюкзак c заданным значением емкости. Задача заключается в том, чтобы заполнить рюкзак предметами, получить максимальную суммарную ценность, соблюдая при этом весовое ограничение.

### 1.4.2 Математическая

Пусть дано n предметов. Для каждого i-го предмета задан вес *> 0* и ценность  *> 0, i =1,..,n.* Задано ограничение на максимальный вес рюкзака . Каждый может принимать значение в диапазоне. Требуется выбрать из заданного множества предметов набор с максимальной суммарной ценностью при одновременном соблюдении ограничения на суммарный вес найденного набора .

|  |  |  |
| --- | --- | --- |
|  |  | **(1.4.1)** |

## 1.5 Свойства задачи о неограниченном рюкзаке

### 1.5.1 Доминирование

Если i-й предмет менее выгодный и тяжелее чем j-й предмет, то он никогда не будет присутствовать в оптимальном решении, так как всегда лучше заменить любую копию i-ого предмета одной или более копиями j-ого предмета без уменьшения выгодности [8, c. 20; 9, c. 3].

Такое отношение между предметами называется **простым**(**однократным) доминированием**. Его расширением является **множественное доминирование,** при котором i-й предмет множественно доминируем j-ым предметом, если *,* то есть всегда лучше заменить одну копию i-ого предмета копиями j-ого предмета.

Существует **пороговое доминирование**, при котором i-й предмет порогово доминируем набором предметов , если, для  *,* то есть всегда лучше заменить копий i-ого предмета некоторой линейной комбинацией предметов Для случая доминирование называется **коллективным**.



Рисунок 1. Типы доминирования

*Для каждой ЗОНР всегда существует оптимальное решение, которое не содержит любые просто, множественно или коллективно доминируемые типы предметов.*

Это означает, что любые просто, множественно или коллективно доминируемые типы предметов могут быть отброшены без изменения оптимального решения, значительно уменьшая пространство поиска [8, c. 20 – 21].

### 1.5.2 Периодичность

Периодичность – это свойство ЗОНР, которое устанавливает, что емкость, большая чем некоторая емкость будет использовать только лучший предмет. Итак, когда такая емкость достигнута (например, методом динамического программирования), оптимальное решение может быть вычислено по следующей формуле [8, c. 25; 9, c. 8]:

|  |  |  |
| --- | --- | --- |
|  |  | **(1.5.1)** |

# **2. Области применения задачи о рюкзаке**

Задача о ранце имеет применение в различных областях знаний: в математике, в экономике, информатике, криптографии. В вычислительной лингвистике в одной из работ предложена формулировка задачи [автоматического реферирования текстов](https://en.wikipedia.org/wiki/Automatic_summarization) (англ.)[русск.](https://ru.wikipedia.org/w/index.php?title=%D0%90%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D1%80%D0%B5%D1%84%D0%B5%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D1%82%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2&action=edit&redlink=1), вырожденный (более простой) случай которой соответствует постановке задачи о ранце.

## 2.1 Изучение в математике

Одно из первых упоминаний о задаче о ранце можно найти в статье [Джорджа Балларда Мэтьюса](https://en.wikipedia.org/wiki/George_Ballard_Mathews) (англ.)[русск.](https://ru.wikipedia.org/w/index.php?title=%D0%9C%D1%8D%D1%82%D1%8C%D1%8E%D1%81,_%D0%94%D0%B6%D0%BE%D1%80%D0%B4%D0%B6_%D0%91%D0%B0%D0%BB%D0%BB%D0%B0%D1%80%D0%B4&action=edit&redlink=1), датированной 1897 годом. Интенсивное изучение данной проблемы началось после публикации [Д. Данцигом](https://ru.wikipedia.org/wiki/%D0%94%D0%B0%D0%BD%D1%86%D0%B8%D0%B3,_%D0%94%D0%B6%D0%BE%D1%80%D0%B4%D0%B6) в 1957 году книги «[англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Discrete Variable Extremum Problem», особенно в 70-90-е годы 20-го века, как теоретиками, так и практиками. Во многом, данный интерес вызван достаточно простой формулировкой задачи, большим числом её разновидностей и свойств и в то же время сложностью их решения [12].

С практической точки зрения задача о рюкзаке может служить моделью для большого числа промышленных ситуаций:

* Размещение грузов в помещении минимального объёма.
* Раскройка ткани — для заданного куска материала получить максимальное число выкроек определенной формы.
* Расчет оптимальных капиталовложений.

С задачей о ранце сталкивается любой человек, собирающий рюкзак.

## 2.2 Изучение в экономике

Часто модификации задачи о ранце имеют практическое значение в экономике.

Предположим, что у нас есть 1000 миллионов долларов, которые нужно вложить в несколько возможных инвестиций. Каждое из этих вложений имеет различную стоимость и различный ожидаемый доход (выгодность). Мы должны решить, как потратить деньги, чтобы получить максимальную прибыль.

Подобные задачи называются задачами формирования инвестиционного портфеля. У нас есть несколько позиций (инвестиций), которые должны поместиться в портфель фиксированного размера (1000 миллионов долларов). Каждая позиция имеет свою прибыльность (выгодность). Необходимо найти набор позиций, которые помещаются в портфель и дают максимальную прибыль.

При составлении портфеля инвестор должен учитывать следующие факторы: степень риска – доходности, срок вложения, тип ценной бумаги. В зависимости от инвестиционной цели инвестор формирует портфель определенного типа. Сейчас принято выделять следующие типы портфелей.

* **Портфели роста**. Целью такого типа является рост капитала преимущественно не за счет получения дивидендов и процентов, а за счет роста курса ценных бумаг. По-другому такой портфель называют курсовым портфелем. Основные вложения делаются преимущественно в акции. В зависимости от соотношения ожидаемого роста капитала и риска можно выделить среди портфелей роста еще и виды портфелей:
  + **Портфели дохода.** Целью этого типа портфелей является получение дохода за набор дивидендов и процентов. Этот тип обеспечивает заранее спланированный уровень дохода при почти нулевом риске. Объектами инвестирования данного типа портфелей выступают высоконадежные ценные бумаги. По-другому портфель называют дивидендным.

Для портфелей роста свойственно быстрое изменение их структуры в зависимости от изменения курсов, входящих в портфель ценных бумаг. Портфели доходов имеют почти постоянные состав и структуру.

Перечисленные типы и виды представляют спектр возможных портфелей, но на практике инвесторы часто формируют портфели смешанного типа.

Наибольший вклад в создании теории инвестиционного портфеля был внесен американскими учеными Г. Марковицем и У. Шарпом, получившими за достижение в этой области Нобелевскую премию в 1990г.

Существует несколько моделей определения характеристик инвестиционного портфеля:

* Модель Марковица
* Модель Шарпа
* Модель «Квази - Шарп»

## 2.3 Изучение в криптографии

Проблема рюкзака лежит в основе первого алгоритма [асимметричного шифрования](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%81_%D0%BE%D1%82%D0%BA%D1%80%D1%8B%D1%82%D1%8B%D0%BC_%D0%BA%D0%BB%D1%8E%D1%87%D0%BE%D0%BC) (шифрования с открытым ключом). Идея [криптографии](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F) с открытыми ключами была выдвинута [Уитфилдом Диффи](https://ru.wikipedia.org/wiki/%D0%A3%D0%B8%D1%82%D1%84%D0%B8%D0%BB%D0%B4_%D0%94%D0%B8%D1%84%D1%84%D0%B8), [Мартином Хеллманом](https://ru.wikipedia.org/wiki/%D0%A5%D0%B5%D0%BB%D0%BB%D0%BC%D0%B0%D0%BD,_%D0%9C%D0%B0%D1%80%D1%82%D0%B8%D0%BD) и независимо – [Ральфом Мерклом](https://ru.wikipedia.org/w/index.php?title=Ralph_Merkle&action=edit&redlink=1) ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Ralph Merkle). Впервые она была представлена Диффи и Хеллманом на Национальной компьютерной конференции ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) National Computer Conference). Новизна по отношению к [симметричным криптосистемам](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BC%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%87%D0%BD%D1%8B%D0%B5_%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B) заключалась в использовании парных ключей – секретного ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) private key, secret key, SK) и открытого ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) public key, PK), создаваемых пользователем. Из названия понятно, что секретный ключ пользователь должен скрывать, а открытый может быть общедоступным. Открытый ключ нужен для шифрования, а секретный для расшифровки. Часто из секретного ключа получают открытый ключ [12].

**Криптосистема Меркла-Хеллмана** – первый, основанный на задаче о ранце, алгоритм для обобщённого шифрования с [открытым ключом](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%81_%D0%BE%D1%82%D0%BA%D1%80%D1%8B%D1%82%D1%8B%D0%BC_%D0%BA%D0%BB%D1%8E%D1%87%D0%BE%D0%BC). Разработан Ральфом Мерклом и Мартином Хеллманом в 1978 году. Был опубликован одностадийный ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) singly-iterated) и мультистадийный варианты ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) multiply-iterated). Алгоритм мог быть использован только для шифрования, но [Ади Шамир](https://ru.wikipedia.org/wiki/%D0%A8%D0%B0%D0%BC%D0%B8%D1%80,_%D0%90%D0%B4%D0%B8) адаптировал его для использования в [цифровых подписях](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%86%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D1%8F_%D0%BF%D0%BE%D0%B4%D0%BF%D0%B8%D1%81%D1%8C).

В дальнейшем было предложено как множество модификаций криптосистемы Меркла-Хеллмана, так и совершенно новых криптосистем на основе задачи о ранце. Среди них:

1. Рюкзак Грэм-Шамира
2. Рюкзак Гудмана-Макколи
3. Рюкзак Накаше-Штерна
4. Рюкзак Шора-Ривеста

**Шифрование с помощью задачи о рюкзаке**

Сообщение шифруется как решение набора задач о ранце.

Рюкзачным вектором назовём упорядоченный набор из n предметов. Для шифрования [открытого текста](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D1%8B%D0%B9_%D1%82%D0%B5%D0%BA%D1%81%D1%82) в двоичном представлении его разбивают на блоки длины (например, (11100) соответствует 5-ти предметам в рюкзаке). Считается, что единица указывает на наличие предмета в рюкзаке, а ноль на его отсутствие [11].

**Пример** шифра, полученного по данному алгоритму. Пусть задан рюкзачный вектор Α = (3 4 6 7 10 11) с длинной n = 6.

Таблица 1. Пример шифрования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| открытый текст | 1 1 1 1 1 0 | 0 0 1 1 0 0 | 0 0 0 0 0 0 | 0 0 0 0 0 1 |
| вещи в рюкзаке | 3 4 6 7 10 11 | 3 4 6 7 10 11 | 3 4 6 7 10 11 | 3 4 6 7 10 11 |
| шифр | 3 + 4 + 6 + 7 + 10 = 30 | 6 + 7 = 13 | 0 | 11 |

Для заданного Α - все криптосистемы – есть числа, не превышающие 41, суммарный вес всех предметов в рюкзачном векторе. Для каждого исходного текста существует единственный криптотекст. В указанном примере можно получить одинаковый шифр для векторов 100010 и 001100. Ошибка?

Верно, ошибка. Для шифрования необходим сверхвозрастающий рюкзак, то есть элементы упорядоченного рюкзачного вектора должны являться [сверхвозрастающей последовательностью.](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%B5%D1%80%D1%85%D0%B2%D0%BE%D0%B7%D1%80%D0%B0%D1%81%D1%82%D0%B0%D1%8E%D1%89%D0%B0%D1%8F_%D0%BF%D0%BE%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C)

**2.4 Изучение в информатике**

При решении задачи о ранце всегда нужно выбирать между точными алгоритмами, которые не применимы для «больших» рюкзаков, и приближенными, которые работают быстро, но не обеспечивают точного решения задачи. Естественно, создание быстрого и достаточно точного алгоритма представляет большой интерес с точки зрения информатики.

Существует множество различных областей в IT индустрии, в которых можно применить задачу о ранце. Например, для планирования спринтов (заполнения sprint backlog) в Scrum методологии гибкой разработки ПО.

*Sprint backlog* – список всех задач, которые нужно сделать в ближайщий спринт, ограниченный четкими временными рамки (1–4 недели). Данный список заполняется из *Product backlog*, который содержит полный набор задач по проекту.

Заполнение sprint backlog – это задача о ранце, где в роли цены выступает  
приоритет задачи, важность для бизнеса; а в роли веса – требуемое количество очков или времени. Также существует еще один параметр – срочность задачи.   
Решения данной задачи можно интегрировать в одну из систем планирования (Jira, Rally) в качестве предлагаемых наборов по умолчанию, как дополнительную возможность. Это ускорит процесс планирования, автоматизировав часть работы.

# **3. Алгоритмы решения задачи о рюкзаке**

## 3.1 Точные алгоритмы решения ЗОР

### 3.1.1 Метод динамического программирования

В основе метода динамического программирования (ДП) лежит принцип оптимальности Беллмана:” Каково *бы ни было состояние системы перед очередным шагом, надо выбирать управление на этом шаге так, чтобы выигрыш на этом шаге плюс оптимальный выигрыш на всех последующих шагах был оптимальным*”. Проще говоря, оптимум на i-ом шаге находится исходя из найденных ранее на предшествующих шагах. Из этого следует, что для того чтобы найти оптимальное решение на последнем шаге надо сначала найти оптимальное решения для первого, затем для второго и так далее пока не пройдем все шаги до последнего.

|  |  |  |
| --- | --- | --- |
|  | оптимальное решение для текущей задачи: | **(3.1.1)** |
|  | находится по следующей формуле: | **(3.1.2)** |
|  | вычисляется по рекуррентной формуле*:* | **(3.1.3)** |

В работе рассмотрены два подхода к реализации ДП:

* *Табличный* (прямой). По формуле **(3.1.3)** последовательно вычисляются ячейки таблицы .
* *Рекуррентный (обратный).* По формуле **(3.1.3)** рекурсивно вычисляется *.* В обратном методе вычисляются только необходимые ячейки таблицы.

Для хранения таблицы стоимости и запоминания того, брался каждый предмет или нет, требуется порядка памяти, временная сложность равна

**3.1.2 Метод ветвей и границ**

[Метод ветвей и границ](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%B2%D0%B5%D1%82%D0%B2%D0%B5%D0%B9_%D0%B8_%D0%B3%D1%80%D0%B0%D0%BD%D0%B8%D1%86) (МВИГ) является вариацией метода полного перебора с исключением заведомо неоптимальных решений.

Пусть есть оптимальное решение . Попытаемся его улучшить, рассмотрев решение на другой ветви. Если на рассматриваемой в данной момент ветви решение становится хуже (с какого-то шага), чем , то прекращаем его исследование и выбираем другую ветвь дерева.

Применяя метод ветвей и границ, можно значительно сократить количество вариантов для перебора. Однако МВИГ работает эффективно не для всех наборов данных. Существуют примеры, в которых время выполнения будет таким же, как и для простого перебора.

В работе рассмотрен следующий подход к реализации метода ветвей и границ:

1. Создаем узел 1. {(индекс предмета)}. Добавляем узел 1 в контейнер. Тип контейнера зависит от выбранного пользователем обхода: в глубину – стек, в ширину – очередь. Задаем нижнюю оценку .
2. *Если* контейнер пуст*,*  – искомое оптимальное решение. *Иначе*, вынимаем узел из контейнера. Создаем узел :

|  |  |  |
| --- | --- | --- |
|  |  | **(3.1.4)** |

1. Если , не добавляем этот узел в контейнер. Переходим к шагу 6.
2. Если то устанавливаем .
3. Считаем нижнюю оценку для текущего узла. Если нижняя оценка больше, чем, добавляем узел в контейнер.
4. Создаем узел . Узел, который не содержит предмет в решении.

|  |  |  |
| --- | --- | --- |
|  |  | **(3.1.5)** |

Считаем нижнюю оценку для текущего узла. Если нижняя оценка больше, чем, добавляем узел в контейнер. Возвращаемся к шагу 2.

**Примечания:**

В качестве алгоритма нахождения нижней оценки выбрана **жадная нижняя оценка**. Предметы, нерассмотренные в текущем решении, сортируются по невозрастанию удельной стоимости и заполняют рюкзак, если не превышают его объем. Если не все предметы поместились в ранец, то берем последний непоместившийся i-ый предмет и увеличиваем нижнюю оценку по следующей формуле:

|  |  |  |
| --- | --- | --- |
|  |  | **(3.1.6)** |

Перед запуском метода ветвей и границ предметы сортируются по невозрастанию удельной стоимости.

Сложность метода ветвей и границ .

## 3.2 Точные алгоритмы решения ЗОНР

### 3.2.1 Классический метод динамического программирования

Динамическое программирование (ДП) — это метод, использующий для решения больших задач решения меньших задач. Большие задачи, решаемые ДП, имеют перекрывающиеся подзадачи, которые используются для построения их оптимального решения. Решение каждой подзадачи обычно хранится в таблице, поэтому подзадачи никогда не рассчитываются более одного раза. ЗОНР может быть решена путем рассмотрения рюкзаков меньших мощностей в качестве подзадач, сохраняющих максимальную прибыль для каждой емкости. Оптимальное решение для рюкзака емкости можно найти, используя решение для рюкзаков с емкостью, меньшими [8, c. 24 – 25].

Предположим, что рюкзак емкости 11 должен быть заполнен двумя предметами: один из них с весом 5 и прибылью 10, а другой с весом 2 и прибылью 3. Предположим, что решение (наибольшая сумма прибыли) для рюкзаков емкостей до 10 известны. Если мы используем первый элемент, мы по-прежнему будем иметь 6 единиц веса, поэтому, если оптимальное решение для ранца веса 6 равно 10, использование первого элемента приведет к решению с прибылью 10 + 10 = 20. Если мы, однако, используем второй элемент, мы по-прежнему будем иметь 9 единиц веса, если оптимальное решение для рюкзака с емкостью 9 равно 16, то при использовании второго элемента получается решение с прибылью 16 + 3 = 19. Таким образом, оптимальное решение для ранца с весом 11 равно 20. И это значение может быть использовано для решения рюкзаков емкостью более 11.

Способ, проиллюстрированный приведенным выше примером, суммируется в следующей функции:

|  |  |  |
| --- | --- | --- |
|  | – текущая емкость | **(3.2.1)** |

Для вычисления функции **(3.2.1)** используется вектор . Этот вектор содержит наилучшую прибыль для каждой емкости . Классический алгоритм динамического программирования для ЗОНР показан на рисунке 3. Сначала в строке 1 устанавливается в ноль, так как все предметы имеют вес больше нуля. Затем для каждой емкости используется тип элемента, который вписывается в текущий рюкзак и который дает наибольшую прибыль. Обратите внимание, что предыдущие решения меньших задач никогда не пересчитываются, они извлекаются из вектора . После вычисления для каждого оптимальное решение можно получить из .



Рисунок 2. Псевдокод классического метода динамического программирования

Временная сложность данного метода – , строка 6 выполняется раз (количество предметов) для каждой емкости, меньшей . Сложность пространства – , так как прибыль для каждой емкости, меньшей , сохраняется.

### 3.2.2 Метод эффективного динамического программирования

Метод эффективного динамического программирования (ЭДП) состоит из двух основных фаз: **фаза сокращения** и **стандартная фаза** [8, c. 39 – 41].

**Предварительная обработка**

ЭДП не требует предварительной обработки, так как любой тип предмета обрабатывается после того, как будет посчитан каждый объем, меньший чем вес текущего предмета, то есть будет уже посчитан для каждого предмета j с . Преимущество данного подхода в том, что простые, множественные и коллективные доминирования могут быть определены простым тестом, выполняющимся за для каждого типа предмета: если для i-ого предмета выполняется неравенство , то i-ый предмет никогда не будет использоваться в оптимальном решении, поэтому он не попадает в набор недоминируемых предметов.

**Фаза сокращения**

На этом этапе предметы обрабатываются в срезах по t, отсортированные по увеличению веса. Держится список недоминируемых предметов F, упорядоченных по невозрастанию удельной стоимости (соотношению цены и веса). В начале F пусто, . Для каждого элемента вычисляется по формуле **(3.2.1)**.

Если прибыль i-ого предмета больше чем текущая , то i-й предмет попадает в F. В противном случае, он отбрасывается, как порогово доминируемый.

**Определение порогового доминирования**

В конце каждого среза предметов проверяется пороговое доминирование для каждого предмета. Предмет является порогово доминируемым для емкостей, больших чем , если он неэффективный для какого-нибудь ранца с емкостью .

Чтобы эффективно определить это, вектор L сохраняет последнюю посчитанную емкость для каждого i-ого предмета***,*** где i-йбыл наиболее эффективным, используемым в оптимальном решении. Для поддержания данного вектора действительным, F сортируется по невозрастанию удельной стоимости, устанавливается равным , если . Итак, если полностью посчитана, любой предмет, для которого *,* может быть удален.



Рисунок 3. Псевдокод определения порогового доминирования

**Стандартная фаза**

На этом шаге высчитывается в срезах по q емкостям. В конце каждого среза проверяется пороговое доминирование для каждого предмета таким же образом, как описано выше. Если мощность F достигает единицы, алгоритм останавливается и использует формулу **периодичности(1.5.1)** для вычисления оптимального решения. Если этого не происходит, алгоритм останавливается, когда емкость достигает .

**Параметры**

Алгоритм требует два параметра:

* t– размер срезов предметов в фазе сокращения;
* q – размер срезов емкости в стандартной фазе.

В конце каждого среза проверяется пороговое доминирование для каждого предмета из F. Если значения параметров будут слишком малы, проверка будет занимать значительный промежуток времени. Если же они будут слишком велики, предметы, которые требуется удалить, будут занимать много времени в процедуре удаления. Подбор параметров t и q не исследуется в данной работе, но планируется в дальнейшем.

**Сложность**

Количество примитивных операций, выполняющихся в данном алгоритме, зависит от размера рюкзака и от количества предметов. Данный алгоритм можно рассматривать как вычисление таблицы. Хотя в большинстве реальных случаев периодичность достигается до вычисления , худший случай все еще может произойти. Заметим, что поскольку сложность алгоритма ограничена не размером задачи, а числовым значением ввода, его временная сложность называется псевдо-полиномиальной.



Рисунок 4. Псевдокод ЭДП

### 3.2.3 Метод ветвей и границ с использованием верхней границы по трем предметам с лучшим соотношением цены и веса ()

Метод ветвей и границ состоит в перечислении каждой комбинации типов предметов, сохраняя нижнюю и верхнюю границы на оптимальном решении. Нижняя граница вычисляется для каждой текущей ветки набора предметов, например, используя жадный алгоритм соотношения цены и веса. Верхняя граница вычисляется для всей задачи.

Для последней мы использовали алгоритм, называемый , который использует три наиболее эффективных типа предметов [8, c. 22 – 23; 9, c. 3]. Предположим, что *.* Верхняя граница определяется как максимум среди и . Пусть – это емкость, оставшаяся после использования максимального количества предметов типа 1, – это емкость, оставшаяся после использования максимального количества предметов типа 2 в рюкзаке емкостью и – это прибыль при решении, использовавшем максимальное количество предметов типа 1 и оставшуюся емкость с предметами типа 2. Таким образом, получаем две оценки:

|  |  |  |
| --- | --- | --- |
|  |  | **(3.2.2)** |
|  |  | **(3.2.3)** |

Граница – это прибыль, полученная использованием оставшейся емкости с предметом 3-его типа, – это значение удаления некоторых предметов 1-ого типа из решения полученного из и замены их предметами 2-ого типа.

|  |  |  |
| --- | --- | --- |
|  |  | **(3.2.4)** |

**Алгоритм МВИГ с верхней оценкой**:

1. Создаем узел 1. {(индекс предмета)}. Добавляем узел 1 в контейнер. Тип контейнера зависит от выбранного пользователем обхода: в глубину – стек, в ширину – очередь. Задаем нижнюю оценку . Верхнюю оценку высчитываем по формуле 3.2.4.
2. *Если* контейнер пуст*,*  – искомое оптимальное решение. *Иначе*, вынимаем узел из контейнера.

Создаем узлы для , изменяющегося от максильмально допустимого количества предмета типа дo 0:

|  |  |  |
| --- | --- | --- |
|  |  | **(3.2.5)** |

2.1 Создаем узел:

2.2 *Если* , не добавляем этот узел в контейнер. Переходим к шагу 2.6.

*2.3 Если* то устанавливаем .

*2.4 Если* , – искомое оптимальное решение.

2.5 Считаем нижнюю оценку для текущего узла. Если нижняя оценка больше, чем, добавляем узел в контейнер.

2.6 Если , переходим к шагу 2, иначе к шагу 2.1.

Примечания, описанные в МВИГ для ЗОР (пункт 3.1.2) также применяются и для данного алгоритма.

Производительность метода ветвей и границ зависит от класса задач, из-за чего трудно предсказать его поведение. В некоторых случаях, он может деградировать до экспоненциального времени выполнения, однако, МВИГ лучше классического метода динамического программирования для задач о рюкзаке с большими емкостями [8, c. 23].

## 3.3 Приближенные алгоритмы

### 3.3.1 Жадный алгоритм (алгоритм Данцига)

Предметы сортируются по убыванию удельной стоимости. Помещаем в рюкзак то, что может поместиться и имеет самую большую цену, т. е. с максимальным отношением цены к весу.

Для сортировки предметов потребуется Далее организуется проход по всем элементам цикла. Точное решение можно получить не всегда.

Пример для ЗОР: Пусть вместимость рюкзака Предметы уже отсортированы. Применяем к ним жадный алгоритм.

Таблица 3. Применение жадного алгоритма

|  |  |  |  |
| --- | --- | --- | --- |
| i | вес | цена | цена/вес |
| 1 | 20 | 60 | 3 |
| 2 | 30 | 90 | 3 |
| 3 | 50 | 100 | 2 |

Кладём в рюкзак первый, а за ним второй предметы. Третий предмет в рюкзак не влезает. Суммарная ценность поместившегося равна 150. Если бы были взяты второй и третий предметы, то суммарная ценность составила бы 190. Видно, что жадный алгоритм не обеспечивает оптимального решения, поэтому относится к приближенным.

Рассмотрим непрерывную задачу о ранце, условия для нее те же самые, отличие лишь в том, что мы можем взять часть предмета. Пусть у нас есть тот же набор, тогда следуя жадному алгоритму, берем первый и второй предметы, полностью третий предмет не помещается, т. к. места осталось всего на 30кг, но мы можем брать части предметов, тогда возьмем веса третьего предмета, соответственно и его стоимости, таким образом мы нагрузили рюкзак полностью, стоимость груза стала равна 210**.** Для непрерывной задачи о рюкзаке жадный алгоритм будет давать оптимальное решение.

### 3.3.2 Генетический алгоритм

**Генетический алгоритм (ГА)** – это [эвристический алгоритм](https://ru.wikipedia.org/wiki/%D0%AD%D0%B2%D1%80%D0%B8%D1%81%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC) поиска, используемый для решения задач оптимизации и моделирования путём случайного подбора, комбинирования и вариации искомых параметров с использованием механизмов, аналогичных [естественному отбору](https://ru.wikipedia.org/wiki/%D0%95%D1%81%D1%82%D0%B5%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BE%D1%82%D0%B1%D0%BE%D1%80) в природе. Является разновидностью [эволюционных вычислений](https://ru.wikipedia.org/wiki/%D0%AD%D0%B2%D0%BE%D0%BB%D1%8E%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F), с помощью которых решаются оптимизационные задачи с использованием методов естественной эволюции, таких как: [наследование](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_(%D0%B1%D0%B8%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%8F)), [мутации](https://ru.wikipedia.org/wiki/%D0%9C%D1%83%D1%82%D0%B0%D1%86%D0%B8%D1%8F), [отбор](https://ru.wikipedia.org/wiki/%D0%95%D1%81%D1%82%D0%B5%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BE%D1%82%D0%B1%D0%BE%D1%80) и [кроссинговер](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%BE%D1%81%D1%81%D0%B8%D0%BD%D0%B3%D0%BE%D0%B2%D0%B5%D1%80). Отличительной особенностью генетического алгоритма является акцент на использование оператора «скрещивания», который производит операцию рекомбинации решений-кандидатов, роль которой аналогична роли скрещивания в живой природе.

Генетический алгоритм не гарантирует нахождение оптимального решения, однако показывает хорошие результаты за меньшее время по сравнению с другими алгоритмами.

Детальное описание ГА и его параметров представлено в следующей главе.

**4. Генетический алгоритм**

**4.1 Интерпретация задачи о ранце и операторов генетического алгоритма c помощью понятий популяционной генетики**

Наименьшей неделимой единицей биологического вида, подверженной действию факторов эволюции, является особь , (индекс *k* обозначает номер особи, а индекс *t* – некоторый момент времени эволюционного процесса). В качестве аналога особи *,* примем произвольное допустимое решение, которому присвоено имя . Вектор управляющих переменных – это наименьшая неделимая единица, характеризующая в задаче о рюкзаке внутренние параметры на каждом *t-*м шаге поиска оптимального решения, которые изменяют свои значения в процессе максимизации критерия оптимальности *Q*. Символьная модель задачи о ранце может быть представлена в виде множества двоичных кодировок, которые описывают конечное множество допустимых решений , принадлежащих области поиска . Для описания особей введем два типа вариабельных признаков, отражающих качественные и количественные различия между особями по степени их выраженности:

* качественные признаки – признаки, которые позволяют однозначно разделять совокупность особей на четко различимые группы;
* количественные признаки – признаки, проявляющие непрерывную изменчивость, в связи с чем степень их выраженности можно охарактеризовать числом.

Качественные признаки особи , определяются из символьной модели задачи – как кодировка , соответствующая точке с именем и составляющие ее . Приведем интерпретацию этих признаков в терминах хромосомной теории наследственности. В качестве гена – единицы наследственного материала, ответственного за формирование альтернативных признаков особи, примем комбинацию , которая определяет фиксированное значение целочисленного кода управляющей переменной . Каждая особь характеризуется генами, а структуру строки можно интерпретировать хромосомой, содержащей сцепленных между собой генов, которые следуют друг за другом в строго определенной последовательности. Хромосому особи будем обозначать , т.е.

|  |  |  |
| --- | --- | --- |
|  |  | **(4.1.1)** |

Согласно хромосомной теории наследственности передача генетической информации будет осуществляться через хромосомы от «родителей» к «потомкам». Местоположение определенного гена в хромосоме называется локусом, а альтернативные формы одного и того же гена, расположенные в одинаковых локусах хромосомы, называются аллелями (аллелеформами). В задаче поиска i-й локус соответствует i-й позиции в строковой кодировке *s(x)*, а аллели – это аналоги множества значений управляющих переменных.

Задача формализуется таким образом, чтобы её решение могло быть закодировано в виде [вектора](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%BA%D1%82%D0%BE%D1%80_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) («[генотипа](https://ru.wikipedia.org/wiki/%D0%93%D0%B5%D0%BD%D0%BE%D1%82%D0%B8%D0%BF)») генов, где каждый ген может быть [битом](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D1%82), числом или неким другим объектом. В классических реализациях генетического алгоритма (ГА) предполагается, что генотип имеет фиксированную длину. Однако существуют вариации ГА, свободные от этого ограничения.

Некоторым, обычно случайным, образом создаётся множество генотипов начальной популяции. Они оцениваются с использованием «функции приспособленности», в результате чего с каждым генотипом ассоциируется определённое значение («приспособленность»), которое определяет насколько хорошо [фенотип](https://ru.wikipedia.org/wiki/%D0%A4%D0%B5%D0%BD%D0%BE%D1%82%D0%B8%D0%BF), им описываемый, решает поставленную задачу. Полученные генотипы проходят несколько стадий (рисунок 5): отбор (селекция), скрещивание, мутация, создание новой популяции (нового поколения). Алгоритм повторяется заданное число итераций [1, с. 33 – 34].



Рисунок 5. Стадии генетического алгоритма

**4.2 Представление генотипа**

Первый этап в реализации генетического алгоритма для определенной задачи – это разработка схемы представления. Для задачи о 0-1 рюкзаке в качестве гена будем рассматривать бит {0,1}, где 1 – предмет попал в рюкзак, 0 – иначе. Генотип будет являться набором битов.

Чтобы решить задачу о неограниченном рюкзаке, мы должны интерпретировать ее в стандартном 0-1 представлении. В качестве гена будет выступать набор битов размером, подходящим для двоичного представления максимального допустимого количества предметов определенного типа [7, c. 1587]. Размер вычисляется как . Например, для задачи с емкостью рюкзака 16, предмет с весом 2 может уместиться 8 раз. Соответственно, для представления числа 8 – нам нужно 4 бита. Ген для предмета с весом 2 будет состоять из четырех бит. Пример представления генотипа в задаче о неограниченном рюкзаке представлен на рисунке 6.



Рисунок 6. Представление генотипа в ЗОНР

**4.3 Операторы создания начальной популяции**

### 4.3.1 Алгоритм Данцига

Начальная популяция генерируется в зависимости от удельной стоимости. Предметы сортируются по убыванию удельной стоимости и в таком порядке помещаются в рюкзак, при этом каждый из предметов кладется с вероятностью.

Что бы получить несколько особей, введен элемент случайности – предметы с вероятностью 50% могут попасть в рюкзак.

### 4.3.2 Случайный алгоритм

Особи генерируются случайным образом с учётом функции приспособленности.

## 4.4 Операторы скрещивания

Механизм размножения с помощью операторов кроссовера прост, он содержит следующие простейшие операции:

* генерация случайных чисел;
* копирование строк фиксированной длины , являющихся «родительскими» кодировками;
* обмен кусками «родительских» кодировок, разрываемых в одной и той же точке хромосомы.

Применение этих операторов приводит к тому, что кодировки-потомки будут содержать новые сочетания аллелей генов, принадлежащих кодировкам родителей, т.е. происходит только перераспределение существующих аллелей родителей по хромосомам потомков, [1, с. 73 – 75].

### 4.4.1 Одноточечный кроссовер

Случайным образом выбирается одна точка разрыва из интервала  с вероятностью *.* Точка разрыва – участок между соседними битами в строке. Обе родительские структуры разрываются на два сегмента по этой точке. Затем, соответствующие сегменты различных родителей склеиваются и получаются два генотипа потомков.

### 4.4.2 Двухточечный кроссовер

«Родительские» кодировки разрываются в двух точках и случайным образом, выбранных с равной вероятностью без возвращения из интервала Затем, соответствующие сегменты различных родителей склеиваются, и получаются два генотипа потомков.

### 4.4.3 Однородный кроссовер

Оператор кроссовера, в котором единственный потомок воспроизводится таким образом, что аллель каждого гена копируется в хромосому потомка либо из хромосомы одного родителя, либо из хромосомы другого родителя с вероятностью .

## 4.5 Операторы мутации

### 4.5.1 Точечная мутация

Кодировка мутанта получается путем однократной перестановки аллелей в случайно выбранном локусе и соседнем с ним.

### 4.5.2 Инверсия

Кодировка разрывается в двух точках и *,* случайно выбранных с равной вероятностью из интервала . Значения аллелей во всех генах куска изменяются на обратные («1» на «0» или «0» на «1»). Кодировка мутанта образуется путем соединения вновь двух старых кусков и нового кусках.

### 4.5.3 Сальтация

В кодировке случайным образом выбираются локусов, в каждой паре локусов переставляются значений аллелей.

### 4.5.4 Транслокация

В кодировке выделяется не один кусок , а сразу несколько непересекающихся между собой участков «родительской» хромосомы. Значения аллелей в мутанте внутри этих участков инвертируются [1, с. 77 – 78].

## 4.6 Оператор обработки ограничений

### 4.6.1 Метод штрафных функций

Идея использования штрафных функций – значительное уменьшение приспособленности особей, не являющихся кодировками допустимых решений. Штрафные функции можно разделить на постоянные и адаптивные. *Постоянная* функция штрафа на все недопустимые особи накладывает одинаковый штраф, *адаптивная* накладывает штраф тем больше, чем «дальше» представляемое решение от допустимой области, например, квадратичная функция штрафа [1, c. 27 – 28].

*В программе использована адаптивная функция штрафа:*

|  |  |  |
| --- | --- | --- |
|  |  | **(4.6.1)** |
|  | *Функция приспособленности:* | **(4.6.2)** |

После применения штрафа к функции приспособленности, некоторые особи, генотип которых является кодировкой недопустимого решения, могут иметь отрицательную или нулевую приспособленность. Существуют два варианта работы с недопустимыми особями:

* Первый – *удаление особей* с приспособленностью меньшей или равной нулю из популяции, но это значительно сокращает численность и генетическое разнообразие популяции.
* Второй вариант *– масштабирование приспособленностей*. В этом случае все особи сохраняются в популяции, но недопустимые особи имеют очень низкую, по сравнению с остальными особями, приспособленность. Одним из наиболее простых и часто используемых методов является *линейное динамическое масштабирование*.

В реализации генетического алгоритма был использован второй вариант, *метод линейного динамического* масштабирования во избежание сокращения численности и генетического разнообразия [1, c. 65 – 71]:

|  |  |  |
| --- | --- | --- |
|  | ,  где ( для задачи максимизации);  – коэффициент масштабирования | **(4.6.3)** |

Коэффициенты и выбираются в каждом *t-*м поколении по информации о значениях функции приспособленности для обрабатываемых кодировок, исходя из следующих двух условий.

* Среднее значение масштабированной функции приспособленности для множества

|  |  |  |
| --- | --- | --- |
|  |  | **(4.6.4)** |

должно равняться среднему значению необработанной функции приспособленности для множества

|  |  |  |
| --- | --- | --- |
|  |  | **(4.6.5)** |

Учитывая **(4.6.1)** получим:

|  |  |  |
| --- | --- | --- |
|  |  | **(4.6.6)** |

Выполнение этого равенства гарантирует, что всякая кодировка , имеющая значение функции приспособленности совпадающее со значением средней приспособленности по популяции, при использовании схемы пропорциональной селекции репродуцирует в среднем одну ожидаемую копию в популяцию .

* Наибольшее значение масштабируемой функции приспособленности должно находиться в следующем отношении со средним значением необработанной функции приспособленности:

|  |  |  |
| --- | --- | --- |
|  |  | **(4.6.7)** |

где – ожидаемое число копий для особи, которая имеет максимальное значение необработанной функции приспособленности в . Выполнение равенства гарантирует, что «наилучшая» кодировка множества репродуцирует при использовании схемы пропорциональной селекции в среднем ожидаемых копий в популяцию .

Таким образом, решая уравнения **(4.6.5) – (4.6.7)** получаем, что для масштабируемой функции приспособленности **(4.6.3)** в каждом t-м поколении коэффициенты и должны вычисляться согласно следующим выражениям:

|  |  |  |
| --- | --- | --- |
|  |  | **(4.6.8)** |

Здесь – значение функции приспособленности «наилучшей» особи в .



Рисунок 7. Зависимость значений масштабируемой функции приспособленности от для

и

Одно из требований, предъявляемых к необработанной функции приспособленности *m*, состоит в том, что она должна принимать неотрицательные значения для всех . Это требование должно сохраняться и для масштабируемой функции приспособленности:

|  |  |  |
| --- | --- | --- |
|  |  | **(4.6.9)** |

Однако для некоторых репродукционных множеств масштабируемая функция приспособленности **(4.6.3)** с коэффициентами и из выражений **(4.6.8)** может принимать отрицательные значения. Такая ситуация возникает обычно, когда большинство

кодировок репродукционного множества являются наиболее приспособленными, и их значения необработанной функции приспособленности отличаются друг от друга очень мало, в то время как ряд кодировок этого же множества имеет значения необработанной функции приспособленности значительно ниже среднего значения приспособленности *.*



Рисунок 8. Зависимость значений масштабируемой функции приспособленности от для и

Для решения проблемы исключения отрицательных значений масштабируемой функции приспособленности **(4.6.3)** потребуем, чтобы она удовлетворяла следующим двум условиям:

1. – значение масштабированной функции приспособленности «наихудшей» кодировки. С учетом этих условий находим, что коэффициенты и , обеспечивающие неотрицательные значения масштабируемой функции приспособленности, должны вычисляться согласно следующим выражениям:

|  |  |  |
| --- | --- | --- |
|  |  | **(4.6.10)** |

*Зависимость значений масштабированной функции приспособленности от значений необработанной функции приспособленности приведена на данном рисунке.*

**

Рисунок 9. Зависимость значений масштабируемой функции приспособленности от для и

Учитывая выражение **(4.6.3)**, условие **(4.6.9)** для «наихудшей» кодировки из можно записать в следующем виде:

|  |  |  |
| --- | --- | --- |
|  |  | **(4.6.11)** |

Тогда, подставляя в неравенство **(4.6.11)** значения , из выражений **(4.6.8)**, получаем неравенство:

|  |  |  |
| --- | --- | --- |
|  |  | **(4.6.12)** |

Таким образом, при линейном динамическом масштабировании **(4.6.3)**, если неравенство **(4.6.12)** для заданного значения выполняется, то коэффициенты и вычисляются с помощью выражений **(4.6.8)**, в противном случае коэффициенты и вычисляются с помощью выражений **(4.6.10)**. В нашем случае, коэффициенты вычисляются по формуле **(4.6.10).**

## 4.7 Операторы селекции

### 4.7.1 Линейно-ранговая схема селекции

Схема линейной ранговой селекции основана на том, что в ней вместо значений функции приспособленности используются ранги :

|  |  |  |
| --- | --- | --- |
|  |  | **(4.7.1)** |

– ранг *i-*й кодировки, когда все особи репродукционного множества , отсортированы в порядке монотонного неубывания значений функции приспособленности.

– нижняя граница ожидаемого числа копий для «наихудшей» особи с наименьшим значением функции приспособленности, имеющей ранг ;

– верхняя граница ожидаемого числа копий для «наилучшей» кодировки с наибольшим значением функции приспособленности, имеющей ранг [1, c. 55 – 58].

Для того чтобы определить численные значения граничных оценок ожидаемого числа копийи, требуется, чтобы значения,определяемые формулой **(4.7.1)** удовлетворяли следующим условиям:

* значениядолжны монотонно увеличиватьсяпо отношению к возрастанию значений функции приспособленности *;*
* общая сумма ожидаемого числа копий решений, репродуцируемых в популяцию , должна равняться численности n этой популяции.

выбирается из интервала1,2случайным образом или детерминировано, авычисляется по формуле: .

На втором этапе селекции был использован алгоритм вероятностного отбора –*остаточный стохастический выбор с возвращением:*

* Пусть целая часть числа, а – дробная часть числа,являются действительными числами, определенными на первом этапе оператора селекции.

*Целочисленная фаза.*

1. Детерминировано в популяцию репродуцируется копий для каждого решения из репродукционного множества .
2. Определяется число недостающих копий ,необходимых для заполнения популяции полностью копиями:

|  |  |  |
| --- | --- | --- |
|  |  | **(4.7.2)** |

1. Если*,*то алгоритм завершает свою работу.

*Дробная фаза.*

1. Определяется
2. Осуществляется отображение всех решений из в смежные отрезки на оси действительных чисел так, чтобы каждый сегмент i-го решения равнялся дробной частиего ожидаемого числа копий .
3. Случайным образом с равной вероятностью генерируется число.
4. В качестве копии выбирается то решение, в отрезок которого попало число .
5. Процедура стохастического выбора с возвращением повторяется с шага 6 до тех пор, пока не будет получено копий.

### 4.7.2турнир

Схема -турнирной селекции организована следующим образом:

Случайно, с равной вероятностью из выбирается группа из особей. Среди выбранных в группу особей определяется «наилучшая» особь, копия которой репродуцируется в популяцию следующего поколения. Группа особей возвращается в . Процедура выбора копии «наилучших» особей повторяется требуемое число раз. Число решений называется размером турнира.

Наиболее простым является бинарный турнир, проводимый между двумя решениями *2*

## 4.8 Оператор восстановления

Попадаются задачи, с которыми метод штрафов не справляется - *незначительно* уменьшает функцию приспособленности у недопустимых кодировок, в результате чего алгоритм может сойтись к такой кодировке. Например, случай, когда у особи, не являющейся кодировкой допустимого решения, вес на единицу превосходит ограничение, а функция приспособленности близка к максимальной.

Во избежание такой ситуации, после селекции применяется *оператор восстановления* [7, c. 1588]. Оператор восстановления берет за основу удельную стоимость , увеличивает или уменьшает значение гена для i-го предмета на основе . Чем больше удельная стоимость, тем выше шанс, что ген будет иметь максимальное допустимое значение для i-го предмета. Оператор восстановления состоит из двух фаз:

* В *первой фазе* анализируется каждая переменная(ген) в порядке неубывания и уменьшается, пока решение не становится допустимым.
* Во *второй фазе* анализируется каждая переменная в порядке невозрастания и увеличивается, пока решение может быть допустимым.

Цель первой фазы – получить допустимое решение из недопустимого, в то время как вторая фаза направлена на улучшение пригодности допустимого решения.

Оператор восстановления гарантированно всегда создает допустимое решение, независимо от исходного решения.

# **5. Классы тестовых задач**

Задача о рюкзаке имеет особенности в виде корреляций между стоимостью и весом, поэтому исследования проводятся для 4 классов тестовых задач:

## 5.1 Задачи без корреляции

В этом случае не существует корреляции между стоимостью и весом элемента. Такие примеры иллюстрируют ситуации, когда вес и стоимость независимы. Некоррелированные экземпляры, как правило, легко решить, так как существует большая разница между весами, из-за чего легко заполнить ранец [4, c. 20; 10, c. 31].

случайно выбираются из диапазона

Рисунок 10. Задачи без корреляции

## 5.2 Задачи с слабой корреляцией

Стоимость коррелирует с весом. Как правило, стоимость отличается от веса на пару процентов. Такие случаи – одни из наиболее реалистичных в управлении (возврат инвестиций, как правило, пропорционален вложенной сумме в некоторых небольших вариациях) [4, с. 20 – 21, 10, c. 31]

****

Рисунок 11. Задачи с корреляцией

## 5.3 Задачи с сильной корреляцией

Подобные ЗОР трудно решить по двум причинам:

* Все аллели вокруг элемента разрыва имеют сходные веса, а это значит, что трудно объединить их таким образом, чтобы заполнить ранец.
* Существует относительно большая потеря при удалении мелких взвешенных элементов, что значит, что мы вообще не можем удалить любые мелкие предметы, чтобы освободить место для большого предмета.



Рисунок 12. Задачи с сильной корреляцией

Такие задачи используются в качестве меры возможности алгоритма для решения сложных проблем. Сортировка по невозрастанию относительной стоимости

соответствует упорядочению согласно неубывания весов.

Так для *0-1* рюкзака относительно легко ввести дополнительное ограничение:

|  |  |  |
| --- | --- | --- |
|  | , *b –* индекс элемента разрыва | **(5.3.1)** |

Ограничение говорит, что ни одно решение не будет содержать более предметов, и эти предметы будут самыми лёгкими. Любое решение, содержащее более тяжелые предметы, будет содержать меньшее количество предметов. Ограничения с противоположным неравенством могут быть наложены в том случае, если имеет место быть отрицательная стоимость.

На основании этого наблюдения Martello, Toth разработали эффективный алгоритм для решения сильно коррелированных задач. Но очевидно, что если хотя бы несколько предметов не последуют структуре сильно коррелированной задачи, то ограничение **(5.3.1)** будет слишком слабым, чтобы иметь какой-нибудь эффект [4, с. 21 – 22, 10, c. 31].

## 5.4 Задачи с подсуммами

Они отражают ситуацию, когда прибыль каждого элемента является линейной функцией веса. Задачи с подсуммами нетрудно решаемы.

  
Рисунок 13. Задачи с подсуммами

Емкость рюкзака выбирается случайным образом из [10000, 100000], удовлетворяя условию, что [4, с. 22, 10, c. 31].

# **6. Программный модуль**

В рамках данной исследовательской работы был реализован программный модуль (ПМ) «Knapsack-Problems», предоставляющий:

* алгоритмы для решения задач о 0-1 рюкзаке и задач о неограниченном рюкзаке,
* проведение экспериментов, описанных в 7 главе с генерацией отчетов по ним.

## 6.1 Структура программного модуля

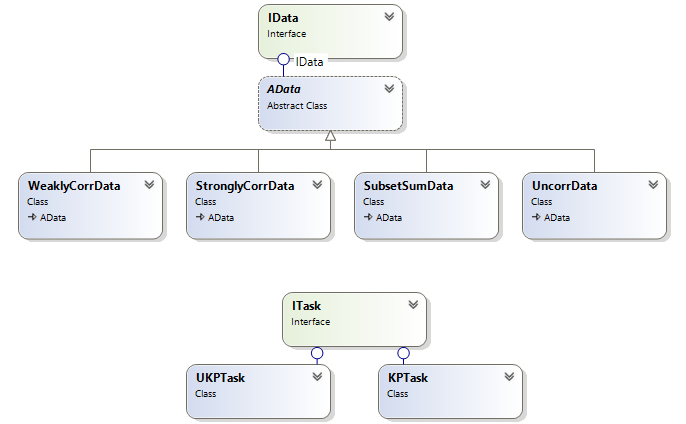
Основными элементами, обеспечивающими функционирование ПО, являются:

1. KnapsackProblem.dll;
2. GenetiAlgorithm.dll;
3. ExactAlgorithms.dll;
4. ExcelReport.dll;
5. CLI.exe.

Также модуль имеет две компоненты тестирования: UnitTests.dll, CorrectnessTests.dll.

### 6.1.1 Библиотека «KnapsackProblem.dll»

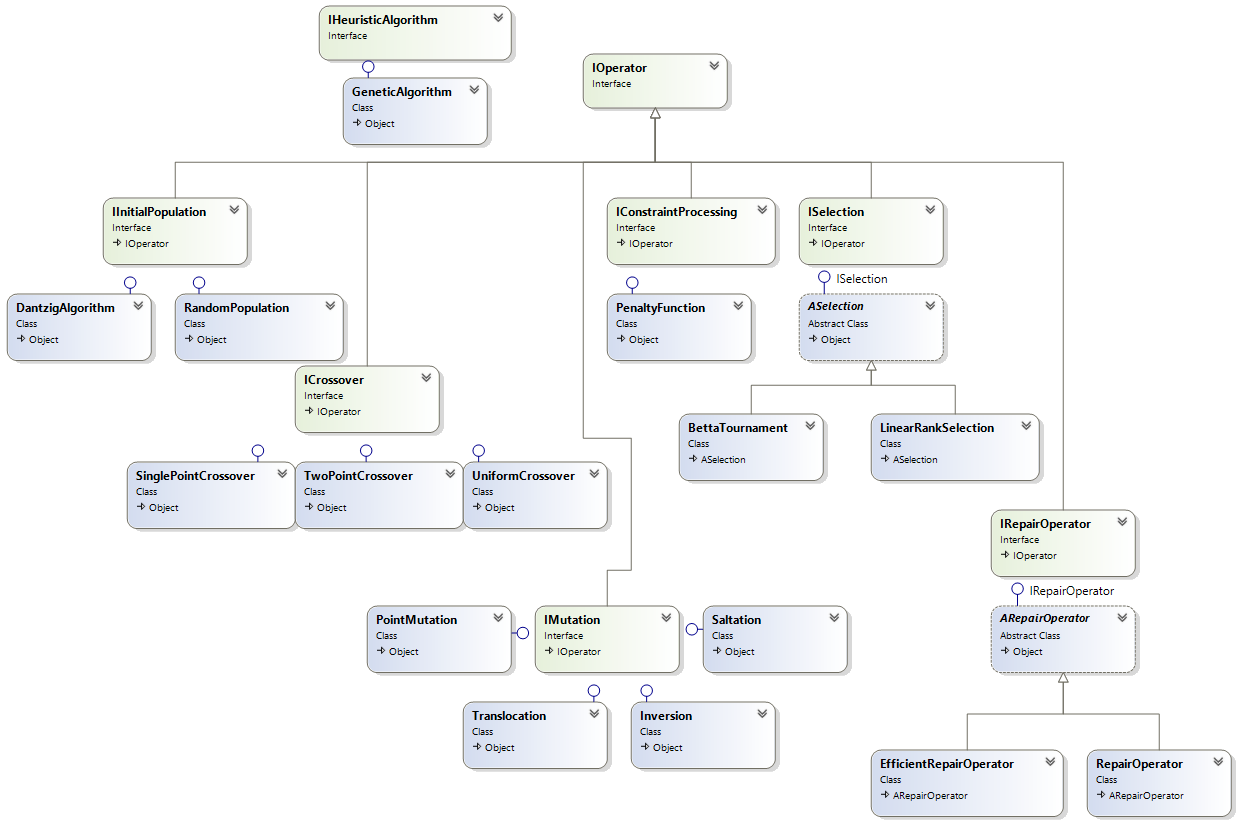
Компонента «KnapsackProblem» предназначена для хранения данных задач ЗОР и ЗОНР в определенном формате для возможности работы различных алгоритмов с уникальными типами. Она представляет собой два интерфейса: IData – хранение «первичных» данных (данных, считанных непосредственно из входной информации), ITask – тип задачи о рюкзаке, в функциональные обязанности которого входит обработка первичных данных для дальнейшего использования алгоритмами.

Рисунок 14. Диаграмма классов библиотеки «KnapsackProblem.dll»

### 6.1.2 Библиотека «GenetiAlgorithm.dll»

Компонента «GenetiAlgorithm» предназначена для решения генетическим алгоритмом задач о 0-1 рюкзаке и задач о неограниченном рюкзаке в формате, представленном в «KnapsackProblem» подсистеме. Основу компоненты представляет реализация GeneticAlgorithm, которая основана на шаблоне «стратегия» и принимает шесть параметров-операторов, представленных следующими интерфейсами:

* IInitialPopulation
* ICrossover
* IMutation
* IConstraintProcessing
* ISelection
* IRepairOperator

Рисунок 15. Диаграмма классов библиотеки «GeneticAlgorithm.dll»

Входными параметрами компоненты являются:

* Данные задачи
* Количество особей
* Количество итераций (поколений)
* Значение для -турнира (необязательный параметр)

### 6.1.3 Библиотека «ExactAlgorithms.dll»

Компонента «ExactAlgorithms» предназначена для решения точными алгоритмами задач о 0-1 рюкзаке и задач о неограниченном рюкзаке в формате, представленном в «KnapsackProblem» подсистеме. Основу компоненты представляют реализации интерфейса точных алгоритмов IExactAlgorithm: DynamicProgramming, BranchAndBound. Алгоритмы основаны на шаблоне «стратегия» для взаимозаменяемости требуемых параметров. DynamicProgramming класс принимает взаимозаменяемый параметр IDPApproach – подход к реализации динамического программирования. BranchAndBound принимает три взаимозаменяемых параметра: IBound – верхняя/нижняя граница для каждой ветви, ITotalBound – верхняя/нижняя общая граница, IFS – обход графа задачи.

Рисунок 16. Диаграмма классов библиотеки «ExactAlgorithms.dll»

Входными параметрами компоненты являются:

* Данные задачи
* Специфичные параметры для алгоритмов

### 6.1.4 Библиотека «ExcelReport.dll»

Компонента «ExcelReport» предназначена для проведения экспериментов, описанных в 7-ой главе и генерации отчетов по ним в формате Excel. Основу компоненты составляют два класса (типа отчетов):

* CombinationsCompareReport – отчет по сравнению различных комбинаций параметров ГА
* MeasurePerformanceReport – отчет по измерению производительности и качества алгоритмов решения ЗОР и ЗОНР. Данный класс использует шаблон «абстрактная фабрика» для создания семейства параметров, используемых в генетическом алгоритме, специфичных для каждого класса тестовых задач, полученных из результатов экспериментов.

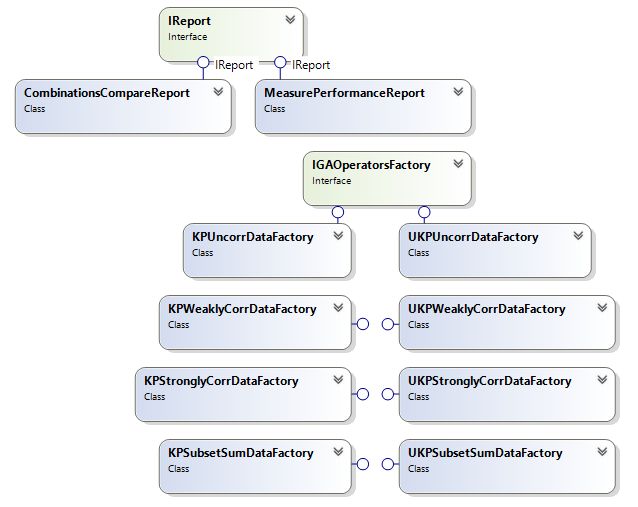


Рисунок 17. Диаграмма классов библиотеки «ExcelReport.dll»

Входными параметрами компоненты являются:

* Тип задачи
* Размер задачи
* Количество экземпляров для каждого класса тестовых задач
* Количество запусков алгоритма(ов) для каждого экземпляра
* Данные, требуемые для генетического алгоритма

### 6.1.5 Приложение «CLI.exe»

Консольное приложение «CLI.exe» запускает эксперимент и генерирует отчет по нему.

Входными параметрами являются:

* Тип эксперимента
* Тип задачи
* Размер задачи
* Количество экземпляров для каждого класса тестовых задач
* Количество запусков алгоритма(ов) для каждого экземпляра
* Данные, требуемые для генетического алгоритма

Все параметры имеют значения по умолчанию.



Рисунок 18. Пример работы с приложением «CLI.exe»

### 6.1.6 Библиотека «UnitTests.dll»

Компонента «UnitTests» предназначена для модульного тестирования – проверки на корректность составляющих исходного кода. Для повышения качества разработанного программного обеспечения основные компоненты покрыты юнит-тестами.



Рисунок 19. Модульные тесты программного обеспечения «Knapsack-Problems»

### 6.1.7 Библиотека «CorrectnessTests.dll»

Компонента «CorrectnessTests» предназначена для валидации алгоритмов из компонент «GeneticAlgorithm», «ExactAlgorithms». Тестовая база состоит из 10 экземпляров тестов для задачи о 0-1 рюкзаке и 28 экземпляров тестов для задачи о неограниченном рюкзаке.

|  |  |
| --- | --- |
|  |  |

Рисунки 20-21. Тесты на корректность программного обеспечения «Knapsack-Problems»

# **7. Эксперименты**

В данной исследовательской работе были поставлены следующие эксперименты:

**Э1**. Поиск «эффективных» комбинаций параметров ГА, дающих наиболее выгодное решение для каждого типа задачи о рюкзаке.

**Э2**. Исследование производительности и качества рассмотренных точных и приближенных алгоритмов, оценка ошибки приближения.

## 7.1 Эксперимент по поиску эффективных комбинаций параметров ГА

### 7.1.1 Описание

Для подбора параметров ГА, влияющих на наиболее эффективный поиск оптимального решения был проведен сравнительный анализ.

Для проведения исследования задаются начальные данные:

* Количество особей.
* Количество итераций (поколений).
* Так как перебираются все возможные комбинации операторов генетического алгоритма, учитывается задание значения для -турнира (по умолчанию = 2).
* Количество экспериментов для конкретной задачи из класса тестовых задач. Один эксперимент – запуск программы с различными комбинациями операторов генетического алгоритма (всего 48 комбинаций = 48 запусков ГА для одного эксперимента).
* Количество запускаемых задач для каждого класса.

Все отчеты представляют собой Excel таблицу.

Для каждой задачи создается отдельная книга, название которой состоит из типа класса тестовых задач и индекса самой задачи, например, “uncorreled\_1.xlsx”, в ней:

* Для каждого эксперимента создается отдельный лист, в котором столбцы – комбинации операторов, а в строках выводится следующая информация:
  + Значение функции приспособленности для каждого поколения.
  + Максимальная функция приспособленности среди всех поколений.
  + Количество итераций до получения к лучшей особи.
* Последний лист – общий результат экспериментов по данной задаче. В нем столбцы – комбинации операторов, а строки содержат следующую информацию:
  + Максимальное значение функций приспособленности среди всех экспериментов.
  + Вероятность получения лучшей особи среди всех экспериментов.
  + Среднее значение количества итераций до получения лучшей особи среди всех экспериментов.

По всем суммарным отчётам одного класса тестовых задач делается общий отчёт в отдельной книге “(type)\_summary.xlsx” (type – тип класса тестовых задач), в котором в строках выводятся комбинации операторов ГА, а в столбцах следующие характеристики:

* Минимальное отклонение значения функции приспособленности в процентном соотношении от оптимума (для нахождения оптимума использован метод динамического программирования).
* Средняя вероятность получения лучшей особи в процентах.
* Среднее количество итераций среди всех задач в процентом соотношении от наибольшего среднего количества итераций до получения лучшей особи (средняя скорость до наилучшего решения).

По данным из общего отчета по конкретному классу тестовых задач, отбираются пять эффективных комбинаций по минимальному отклонению среди минимальных, по максимальной средней вероятности получения лучшей особи и по минимальному среднему количеству итераций. Среди всех пятерок выбирается тройка эффективных комбинаций по максимальному количеству классов, содержащих комбинацию.

*Эффективные (лучшие)* комбинации – комбинации операторов, влияющих на наиболее эффективный поиск решения.

**Для одного запуска генетического алгоритма взяты следующие параметры:**

* 40 особей
* 30 поколений
* = 14 (для бета-турнирной селекции)

**Для каждого типа задачи о рюкзаке проведено по 600 экспериментов (28800 запусков ГА), из них:**

* 30 экспериментов на 1 задачу
* По 5 задач на конкретный класс тестовых задач
* Рассмотрено 4 класса тестовых задач

Размерность задач составляет 15 предметов.

### 7.1.2 Результаты

Ниже представлены таблицы с комбинациями параметров ГА, дающими «лучшие» решения. В таблицах отсутствует минимальное среднее отклонение, так как для всех представленных комбинаций оно равно нулю.

Результаты для **Э1** задачи **ЗОР** предоставлены в таблицах 2 – 5 со следующими значениями строк:

1 – *набор параметров*

2 – *вероятность нахождения глобального оптимума %*

3 – *среднее количество итераций % (скорость нахождения глобального оптимума)*

Таблица 2. Задачи без корреляции

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Алгоритм Данцига Одноточечный  кроссовер Точечная мутация Линейная ранговая схема | Алгоритм Данцига Одноточечный  кроссовер Инверсия Линейная ранговая схема | Алгоритм Данцига Одноточечный  кроссовер Транслокация Линейная ранговая схема | Алгоритм Данцига Одноточечный  кроссовер Сальтация Линейная ранговая схема | Алгоритм Данцига Двухточечный  кроссовер Точечная мутация Линейная ранговая схема |
| 100 | 100 | 100 | 100 | 100 |
| 3,33 | 3,33 | 3,33 | 3,33 | 3,33 |

Таблица 3. Задачи с слабой корреляцией

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Алгоритм Данцига Одноточечный  кроссовер Точечная мутация Линейная ранговая схема | Алгоритм Данцига  Однородный  кроссовер Точечная мутация Линейная ранговая схема | Случайный алгоритм Одноточечный  кроссовер Точечная мутация Линейная ранговая схема | Алгоритм Данцига Двухточечный  кроссовер Точечная мутация Линейная ранговая схема | Случайный алгоритм Двухточечный  кроссовер Точечная мутация Линейная ранговая схема |
| 97,334 | 96 | 94,67 | 93,334 | 93,334 |
| 13,864 | 14,468 | 15,932 | 16,154 | 17,87 |

Таблица 4. Задачи с сильной корреляцией

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Алгоритм Данцига Однородный кроссовер  Инверсия  Бета-Турнир | Случайный алгоритм Одноточечный кроссовер  Сальтация Линейная ранговая схема | Алгоритм Данцига  Одноточечный кроссовер Сальтация Линейная ранговая схема | Алгоритм Данцига Однородный кроссовер Транслокация Линейная ранговая схема | Случайный алгоритм  Одноточечный кроссовер  Инверсия  Бета-Турнир |
| 29,998 | 28,67 | 28 | 28 | 26,67 |
| 10,82 | 5,022 | 6 | 9,27 | 9,064 |

Таблица 5. Задачи с подсуммами

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Случайный алгоритм Двухточечный кроссовер Инверсия Линейная ранговая схема | Алгоритм Данцига Однородный кроссовер Инверсия Линейная ранговая схема | Случайный алгоритм Однородный кроссовер  Инверсия Линейная ранговая схема | Алгоритм Данцига Двухточечный кроссовер Инверсия Линейная ранговая схема | Алгоритм Данцига Одноточечный кроссовер Инверсия Линейная ранговая схема |
| 84,668 | 81,332 | 79,334 | 79,334 | 74,67 |
| 15,554 | 15,732 | 13,512 | 14,776 | 17,534 |

Таблица 6. По всем классам задач

|  |  |  |  |
| --- | --- | --- | --- |
| Набор параметров | Алгоритм Данцига  Одноточечный кроссовер  Точечная мутация  Линейная ранговая схема | Алгоритм Данцига  Одноточечный кроссовер  Сальтация  Линейная ранговая схема | Алгоритм Данцига  Одноточечный кроссовер  Инверсия  Линейная ранговая схема |
| Количество классов, содержащих комбинацию | 2 | 2 | 2 |

Результаты для **Э1** задачи **ЗОНР** предоставлены в таблицах 7 – 10 со следующими значениями строк:

1 – *набор параметров*

2 – *вероятность нахождения глобального оптимума %*

3 – *среднее количество итераций % (скорость нахождения глобального оптимума)*

Таблица 7. Задачи без корреляции

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Алгоритм Данцига Однородный кроссовер Сальтация Линейная ранговая схема | Алгоритм Данцига Одноточечный  кроссовер Транслокация Бета-Турнир | Алгоритм Данцига  Двухточечный кроссовер Транслокация Бета-Турнир | Алгоритм Данцига  Однородный кроссовер Транслокация Бета-Турнир | Алгоритм Данцига  Однородный кроссовер Инверсия Линейная ранговая схема |
| 100 | 100 | 100 | 100 | 100 |
| 3,486 | 3,53 | 3,576 | 3,62 | 3,622 |

Таблица 8. Задачи с слабой корреляцией

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Алгоритм Данцига Однородный кроссовер Инверсия Линейная ранговая схема | Алгоритм Данцига Однородный кроссовер Точечная мутация Линейная ранговая схема | Алгоритм Данцига Двухточечный кроссовер  Сальтация Линейная ранговая схема | Алгоритм Данцига Двухточечный кроссовер  Точечная мутация Линейная ранговая схема | Алгоритм Данцига Одноточечный кроссовер Сальтация Линейная ранговая схема |
| 88 | 82,668 | 80,668 | 78,67 | 78,67 |
| 17,246 | 9,708 | 7,222 | 9,044 | 6,178 |

Таблица 9. Задачи с сильной корреляцией

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Алгоритм Данцига Однородный кроссовер Сальтация Линейная ранговая схема | Алгоритм Данцига Одноточечный кроссовер  Сальтация Линейная ранговая схема | Алгоритм Данцига Одноточечный кроссовер  Точечная мутация Линейная ранговая схема | Алгоритм Данцига Двухточечный кроссовер  Точечная мутация Линейная ранговая схема | Алгоритм Данцига Однородный кроссовер  Точечная мутация Линейная ранговая схема |
| 94 | 93,334 | 92,67 | 89,334 | 88 |
| 9,822 | 13,758 | 12,668 | 10,756 | 9,022 |

Таблица 10. Задачи с подсуммами

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Случайный алгоритм Однородный кроссовер Точечная мутация Линейная ранговая схема | Случайный алгоритм Двухточечный кроссовер Точечная мутация Линейная ранговая схема | Случайный алгоритм Одноточечный кроссовер Точечная мутация Линейная ранговая схема | Алгоритм Данцига Одноточечный кроссовер Точечная мутация  Линейная ранговая схема | Алгоритм Данцига Двухточечный кроссовер Точечная мутация  Линейная ранговая схема |
| 83,334 | 79,998 | 76,668 | 72,668 | 72,67 |
| 24,022 | 23,954 | 23,264 | 25,976 | 23,98 |

Таблица 11. По всем классам задач

|  |  |  |  |
| --- | --- | --- | --- |
| Набор параметров | Алгоритм Данцига  Двухточечный кроссовер  Точечная мутация  Линейная ранговая схема | Алгоритм Данцига  Однородный кроссовер  Сальтация  Линейная ранговая схема | Алгоритм Данцига  Однородный кроссовер  Инверсия  Линейная ранговая схема |
| Количество классов, содержащих комбинацию | 3 | 2 | 2 |

В 90% от числа рассмотренных оптимальных комбинаций для задачи о 0-1 рюкзаке и в 85% для задачи о неограниченном рюкзаке в качестве оператора селекции выступает линейная ранговая схема. Это подтверждает теорию о том, что данная схема предотвращает преждевременную сходимость и приводит к наиболее эффективному решению.

Лидирующую позицию среди операторов начальной популяции занимает алгоритм Данцига: 70% от числа оптимальных комбинаций для ЗОР и 85% – для ЗОНР.

Среди операторов кроссовера и мутации определились следующие лидеры:

* Относительно задачи о 0-1 рюкзаке, чаще остальных операторов кроссовера встречается одноточечный кроссовер – 50% от числа оптимальных комбинаций; чаще остальных операторов мутации – инверсия в 40% комбинаций.
* Относительно задачи о неограниченном рюкзаке, чаще остальных операторов кроссовера встречается однородный кроссовер – 40% от числа оптимальных комбинаций; чаще остальных операторов мутации – точечная мутация в 50% комбинаций.

Относительно классов тестовых задач для ЗОР, для задач с сильной корреляцией вероятность нахождения глобального оптимума не превышает 29,9%, что подтверждает теорию о труднорешаемости таких задач, напротив, для задач без корреляции, с слабой корреляцией и с подсуммами вероятность нахождения глобального оптимума превышает 72%, что говорит о менее трудном нахождении их решения.

Что касается ЗОНР, генетический алгоритм показывает хорошие результаты – для всех классов задач вероятность нахождения глобального оптимума превышает 72%.

## 7.2 Эксперимент по исследованию производительности и качества алгоритмов

### 7.2.1 Описание

Произведен анализ производительности (скорости выполнения) и качества среди реализованных алгоритмов, чтобы подобрать к каждому классу тестовых задач наилучший подход к решению.

Рассматриваемые алгоритмы для решения задачи о 0-1 рюкзаке:

1. Табличный метод динамического программирования
2. Рекуррентный метод динамического программирования
3. Метод ветвей и границ, обход в глубину
4. Метод ветвей и границ, обход в ширину
5. Генетический алгоритм

Рассматриваемые алгоритмы для решения задачи о неограниченном рюкзаке:

1. Классический метод динамического программирования
2. Метод эффективного динамического программирования
3. Метод ветвей и границ с использованием верхней границы по трем предметам с лучшим соотношением цены и веса (U3), обход в глубину
4. Метод ветвей и границ с использованием верхней границы по трем предметам с лучшим соотношением цены и веса (U3), обход в ширину
5. Генетический алгоритм

Входные данные:

1. Размер экземпляров задач
2. Количество запусков алгоритма для каждого экземпляра для нахождения медианы/среднего значения
3. Количество экземпляров на каждый класс тестовых задач
4. Требуемые входные параметры для ГА

В качестве операторов ГА для каждого класса тестовых задач выбраны эффективные комбинации, полученные из результатов эксперимента Э1.

Для каждого класса тестовых задач создается отчет в виде Excel таблицы, столбцы которой содержат названия исследуемых алгоритмов, а в строках для каждого решенного экземпляра задачи выводится следующая информация:

* Медиана времени выполнения алгоритма по заданному количеству запусков
* Среднее отклонение от оптимума % по заданному количеству запусков

Среди всех алгоритмов для каждого экземпляра класса находятся:

* Минимальная медиана времени выполнения
* Минимальное среднее отклонение от оптимума %

Алгоритм с данными показателем считается лучшим для текущего экземпляра класса тестовых задач.

Среди всех лучших алгоритмов для текущего экземпляра задачи выбираются лучшие алгоритмы для конкретного класса задач по максимальному количеству минимальных медиан времени выполнения и максимальному количеству минимальных средний отклонений.

В дополнение, для ГА выводятся следующие характеристики:

* Информация о количестве задач в %, в котором алгоритм дает приближенное решение
* Информация о максимальном и среднем отклонениях решения ГА от глобального оптимума по средним отклонениям каждого экземпляра задачи (в %)

Для задачи о неограниченном рюкзаке дополнительно выводятся следующие характеристики:

* Количество задач в %, для решения которых применилось свойство периодичности
* Количество задач в %, решения которых достигли верхней оценки

**Для одного запуска генетического алгоритма взяты следующие параметры:**

* 40 особей
* 30 поколений
* = 14 (для бета-турнирной селекции)

**Для каждого типа задачи о рюкзаке проведены эксперименты со следующими параметрами:**

* Размеры задачи: 50, 500 предметов
* По 30 запусков каждого алгоритма для решения одного экземпляра задачи
* По 30 экземпляров на конкретный класс тестовых задач
* Рассмотрено 4 класса тестовых задач

### 7.2.2 Результаты

Ниже представлены таблицы с алгоритмами, эффективными по скорости выполнения среди рассмотренных для конкретного класса задач. Эффективность алгоритма по скорости выполнения подразумевает количество задач, при котором была достигнута лучшая производительность среди прочих.

Результаты для **Э2** задачи **ЗОР** предоставлены в таблицах 12 – 15.

Результаты для задач из 50 предметов:

Таблица 12. Эффективные алгоритмы для каждого класса задач

|  |  |  |
| --- | --- | --- |
| Класс задач | Алгоритм решения | Количество задач, при котором алгоритм эффективнее других по скорости выполнения, % |
| Задачи без корреляции | Метод ветвей и границ, обход в глубину | 93,33 |
| Метод ветвей и границ, обход в ширину | 16,67 |
| Задачи с слабой корреляцией | Метод ветвей и границ, обход в ширину | 100 |
| Метод ветвей и границ, обход в глубину | 10 |
| Задачи с сильной корреляцией | Табличный метод динамического программирования | 100 |
| Задачи с подсуммами | Метод ветвей и границ, обход в глубину | 100 |

Таблица 13. Характеристики ГА для каждого класса задач

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Задачи без корреляции | Задачи с слабой корреляций | Задачи с сильной корреляцией | Задачи с подсуммами |
| Количество задач, в котором ГА дает приближенное решение, % | 40 | 53,33 | 100 | 83,33 |
| Макс. отклонение от глобального оптимума, % | 0,4 | 0,22 | 0,3 | 0,00072 |
| Среднее отклонение от глобального оптимума, % | 0,05 | 0,04 | 0,21 | 0,00018 |

Результаты для задач из 500 предметов:

Таблица 14. Эффективные алгоритмы для каждого класса задач

|  |  |  |
| --- | --- | --- |
| Класс задач | Алгоритм решения | Количество задач, при котором алгоритм эффективнее других по скорости выполнения, % |
| Задачи без корреляции | Табличный метод динамического программирования | 90 |
| Метод ветвей и границ, обход в глубину | 10 |
| Задачи с слабой корреляцией | Табличный метод динамического программирования | 100 |
| Задачи с сильной корреляцией | Табличный метод динамического программирования | 100 |
| Задачи с подсуммами | Табличный метод динамического программирования | 100 |

Таблица 15. Характеристики ГА для каждого класса задач

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Задачи без корреляции | Задачи с слабой корреляций | Задачи с сильной корреляцией | Задачи с подсуммами |
| Количество задач, в котором ГА дает приближенное решение, % | 100 | 96,67 | 100 | 0 |
| Макс. отклонение от глобального оптимума, % | 0,7 | 0,03 | 1,58 | 0 |
| Среднее отклонение от глобального оптимума, % | 0,39 | 0,01 | 1,27 | 0 |

Результаты для **Э2** задачи **ЗОНР** предоставлены в таблицах 16 – 21.

Результаты для задач из 50 предметов:

Таблица 16. Эффективные алгоритмы для каждого класса задач

|  |  |  |
| --- | --- | --- |
| Класс задач | Алгоритм решения | Количество задач, при котором алгоритм эффективнее других по скорости выполнения, % |
| Задачи без корреляции | Метод эффективного динамического программирования | 86,67 |
| Метод ветвей и границ с верхней границей , обход в глубину | 80 |
| Метод ветвей и границ с верхней границей , обход в ширину | 80 |
| Задачи с слабой корреляцией | Метод ветвей и границ с верхней границей , обход в ширину | 80 |
| Метод ветвей и границ с верхней границей , обход в глубину | 66,67 |
| Метод эффективного динамического программирования | 43,33 |
| Задачи с сильной корреляцией | Метод ветвей и границ с верхней границей , обход в ширину | 96,67 |
| Метод ветвей и границ с верхней границей , обход в глубину | 86,67 |
| Метод эффективного динамического программирования | 30 |
| Задачи с подсуммами | Метод ветвей и границ с верхней границей , обход в ширину | 80 |
| Метод ветвей и границ с верхней границей , обход в глубину | 23,33 |
| Классический метод динамического программирования | 10 |

Таблица 17. Характеристики ГА для каждого класса задач

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Задачи без корреляции | Задачи с слабой корреляций | Задачи с сильной корреляцией | Задачи с подсуммами |
| Количество задач, в котором ГА дает приближенное решение, % | 0 | 3,33 | 16,67 | 36,67 |
| Макс. отклонение от глобального оптимума, % | 0 | 0,0039 | 0,29 | 0,0031 |
| Среднее отклонение от глобального оптимума, % | 0 | 0,00013 | 0,01 | 0,00045 |

Таблица 18. Характеристики решения ЗОНР для каждого класса задач

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Задачи без корреляции | Задачи с слабой корреляций | Задачи с сильной корреляцией | Задачи с подсуммами |
| Количество задач, для решения которых применилось свойство периодичности, % | 86,67 | 43,33 | 30 | 0 |
| Количество задач, решения которых достигли верхней оценки , % | 0 | 0 | 0 | 36,67 |

Результаты для задач из 500 предметов:

Таблица 19. Эффективные алгоритмы для каждого класса задач

|  |  |  |
| --- | --- | --- |
| Класс задач | Алгоритм решения | Количество задач, при котором алгоритм эффективнее других по скорости выполнения, % |
| Задачи без корреляции | Метод эффективного динамического программирования | 66,7 |
| Метод ветвей и границ с верхней границей , обход в глубину | 63,33 |
| Метод ветвей и границ с верхней границей , обход в ширину | 60 |
| Задачи с слабой корреляцией | Метод ветвей и границ с верхней границей , обход в ширину | 80 |
| Метод ветвей и границ с верхней границей , обход в глубину | 70 |
| Метод эффективного динамического программирования | 33,33 |
| Задачи с сильной корреляцией | Метод ветвей и границ с верхней границей , обход в ширину | 63,33 |
| Метод ветвей и границ с верхней границей , обход в глубину | 63,33 |
| Метод эффективного динамического программирования | 63,33 |
| Задачи с подсуммами | Классический метод динамического программирования | 73,33 |
| Метод ветвей и границ с верхней границей , обход в ширину | 26,67 |
| Метод ветвей и границ с верхней границей , обход в глубину | 6,67 |

Таблица 20. Характеристики ГА для каждого класса задач

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Задачи без корреляции | Задачи с слабой корреляций | Задачи с сильной корреляцией | Задачи с подсуммами |
| Количество задач, в котором ГА дает приближенное решение, % | 16,7 | 30 | 16,7 | 0 |
| Макс. отклонение от глобального оптимума, % | 0,063 | 0,203 | 0,026 | 0 |
| Среднее отклонение от глобального оптимума, % | 0,0035 | 0,011 | 0,0014 | 0 |

Таблица 21. Характеристики решения ЗОНР для каждого класса задач

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Задачи без корреляции | Задачи с слабой корреляций | Задачи с сильной корреляцией | Задачи с подсуммами |
| Количество задач, для решения которых применилось свойство периодичности, % | 50 | 30 | 40 | 0 |
| Количество задач, решения которых достигли верхней оценки , % | 0 | 0 | 0 | 23,33 |

Из полученных результатов для задачи о 0-1 рюкзаке размером в 50 предметов видно, что для класса задач без корреляции и класса задач с подсуммами метод ветвей и границ с обходом в глубину показывает лучший результат по скорости решения (минимальное время выполнения в миллисекундах) среди прочих рассмотренных алгоритмов. Для задач с слабой корреляцией лидирующую позицию занимает метод ветвей и границ с обходом в ширину, для задач с сильной корреляцией – табличный метод динамического программирования.

Для умеренных ЗОР размером в 500 предметов для всех классов тестовых задач лучшим по скорости решения среди прочих рассмотренных алгоритмов является табличный метод динамического программирования.

Из результатов для задачи о неограниченном рюкзаке видно, что для класса задач без корреляции свойство периодичности достигается в 86,67% от числа решенных экземпляров задач размером в 50 предметов и в 50% от числа решенных экземпляров задач из 500 предметов, вследствие чего алгоритм эффективного динамического программирования показывает лучший результат по времени выполнения. Для классов задач с слабой корреляцией и сильной корреляцией показатель достижения свойства периодичности снижается, поэтому «в лидеры вырываются» методы ветвей и границ с обходом в ширину и в глубину.

Для задач с подсуммами достигается общая верхняя граница в 36,67% от числа решенных экземпляров задач из 50 предметов, что позволяет методу ветвей и границ удерживать преимущество по сравнению с остальными алгоритмами. Для умеренных задач с подсуммами размером в 500 предметов достижение общей верхней границы составляет 23,33% от числа решенных экземпляров задач, а самым быстрым алгоритмом по скорости решения является классический метод динамического программирования.

Стоит учитывать, что при использовании МВИГ в больших задачах можно столкнуться с проблемой недостатка памяти, заложенной под хранение узлов, в результате чего алгоритм закончит работу с ошибкой о нехватке памяти. Особенно часто такая ситуация возникает в методе ветвей и границ с обходом в ширину. Для сильно коррелированных задач и задач с подсуммами о 0-1 рюкзаке в 500 предметов МВИГ с обходом в ширину ни разу не выполнился успешно.

Относительно ГА, для ЗОР глобальный оптимум был достигнут не более, чем в 60% решенных экземпляров задач для каждого класса тестовых задач, исключая задачи с подсуммами в 500 предметов; напротив, для ЗОНР генетический алгоритм нашел точное решение не менее, чем в 63% от числа решенных экземпляров для каждого класса тестовых задач.

# **Заключение**

В рамках данной работы:

1. Изучен теоретический материал о задаче о 0-1 рюкзаке, о задаче о неограниченном рюкзаке и её свойствах. Произведен разбор статей с предлагаемыми методами решения рассмотренных типов задачи о ранце. Теоритические выкладки представлены в главах 1–5.
2. Разработан программный модуль «Knapsack-Problems», предоставляющий:
   * алгоритмы для решения ЗОР и ЗОНР,
   * проведение экспериментов, описанных в 7 главе с генерацией отчетов по ним.

ПМ содержит реализации следующих алгоритмов:

1. Для решения задачи о 0-1 рюкзаке:
   * Табличный метод динамического программирования
   * Рекуррентный метод динамического программирования
   * Метод ветвей и границ, обход в глубину
   * Метод ветвей и границ, обход в ширину
2. Для решения задачи о неограниченном рюкзаке:

* Классический метод динамического программирования
* Метод эффективного динамического программирования
* Метод ветвей и границ с использованием верхней границы по трем предметам с лучшим соотношением цены и веса (), обход в глубину
* Метод ветвей и границ с использованием верхней границы по трем предметам с лучшим соотношением цены и веса (), обход в ширину

1. Генетический алгоритм c различными наборами параметров

Программный модуль покрыт юнит-тестами для поддержания качества предлагаемых решений, тестами на корректность для валидации реализованных алгоритмов.

Полное описание ПМ представлено в главе 6.

1. Проведены следующие эксперименты:
   * Поиск «эффективных» комбинаций параметров ГА, дающих наиболее выгодное решение для каждого типа задачи о рюкзаке.

Из полученных результатов исследования, относительно рассматриваемого типа задачи о рюкзаке, для каждого класса тестовых задач выбрано по 5 эффективных комбинаций по минимальному отклонению от глобального оптимума, максимальной вероятности нахождения глобального оптимума и лучшей скорости нахождения глобального оптимума. Для всех классов тестовых задач выбраны три эффективные комбинации по максимальному количеству классов, содержащих комбинацию.

* + Исследование производительности и качества рассмотренных точных и приближенных алгоритмов, оценка ошибки приближения.

Из полученных результатов выбираются лучшие алгоритмы для конкретного класса задач (относительно рассматриваемого типа задач) по максимальному количеству минимальных медиан времени выполнения и максимальному количеству минимальных средний отклонений от точного решения.

В дополнение, для ГА выводятся следующие характеристики:

* Информация о количестве задач в %, в которых алгоритм дает приближенное решение
* Информация о максимальном и среднем отклонениях решения ГА от глобального оптимума по средним отклонениям каждого экземпляра задачи (в %)

Для задачи о неограниченном рюкзаке дополнительно выводятся следующие характеристики:

* Количество задач в %, для решения которых применилось свойство периодичности
* Количество задач в %, решения которых достигли верхней оценки

Описание экспериментов и полученных результатов представлено в главе 7.

В планах на будущее:

* + Подбор параметров t и q, влияющий на скорость решения, для алгоритма эффективного динамического программирования
  + Исследование возможности гибридизации или иной модернизации рассмотренных алгоритмов для улучшения скорости/качества решения

# **Литература**

1. Батищев, «Применение генетических алгоритмов к решению задач дискретной оптимизации» / Д.И. Батищев, Е. А. Неймарк, Н. В. Старостин. – Н.Н : ННГУ им. Лобачевского, 2007.
2. Пападимитриу, «Комбинаторная оптимизация. Алгоритмы и сложность» / Х. Пападимитриу, К. Стайглиц. – М.: Мир, 1985.
3. Гери, «Вычислительные машины и трудноҏешаемые задачи» / М. Гери, Д. Джонсон. – М.: Мир, 1982.
4. David Pisinger, Algorithms for Knapsack Problems, Ph.D. thesis, February 1995, Dept, of Computer Science, University of Copenhagen, Universitetsparken 1, DK-2100 Copenhagen, Denmark
5. И. Сигал, «Введение в прикладное дискретное программирование: модели и вычислительные алгоритмы» / И. Сигал, А. Иванова – 2 изд. – М.: Физматлит, 2007 – 304 с.
6. [A.N.M. Bazlur Rashid](http://www.amazon.com/s/ref=dp_byline_sr_book_1?ie=UTF8&text=A.N.M.+Bazlur+Rashid&search-alias=books&field-author=A.N.M.+Bazlur+Rashid&sort=relevancerank), The 0-1 Knapsack Problem: A solution by Genetic Algorithm Paperback – October 1, 2010
7. KEN-LI LI, GUANG-MING DAI, QING-HUA LI, A Genetic Algorithm for the Unbounded Knapsack Problem, November 2003, Computer School, Huazhong University of Science and Technology, Wuhan, 430074, China
8. LEONARDO FERNANDO DOS SANTOS MOURA, An Efficient Dynamic Programming Algorithm For The Unbounded Knapsack Problem, Porto Alege, December 14th, 2012
9. Vincent Poirriez, Nicola Yanev, Rumen Andonov. A Hybrid Algorithm for the Unbounded Knapsack Problem. Discrete Optimization, Elsevier, 2009, 6, pp.110-124.
10. Swarna Chitra Lyer, A Complementary Heuristic for the Unbounded Knapsack Problem, Department of Computer and Mathematical Sciences Victoria University of Technology December, 199
11. Официальный сайт «Википедия», раздел «Задача о ранце» – Режим доступа: <https://ru.wikipedia.org/wiki/Задача_о_ранце>
12. Официальный сайт «Словари и энциклопедии на Академике», раздел «Задача о ранце» – Режим доступа: https://dic.academic.ru/dic.nsf/ruwiki/107129

**Приложение**

Программный продукт имеет открытый исходный код, доступный по ссылке на GitHub репозиторий - <https://github.com/ElizJogar/Knapsack-Problems.git>

Данные проведенных экспериментов находятся в репозитории по упомянутой ссылке, в папке “results”:

* Результаты эксперимента по поиску эффективных комбинаций параметров ГА – “results/combinations\_compare\_reports/”
* Результаты эксперимента по исследованию производительности и качества алгоритмов – “results/measure\_performance\_reports/”

1. dic.academic.ru/dic.nsf/ruwiki/107129 – словари и энциклопедии на Академике [↑](#footnote-ref-1)