# Вступление.

Задачу упаковки в контейнеры можно сформулировать как: имеется набор контейнеров определенного объема, и набор предметов, которые в эти контейнеры требуется уложить. Цель - уложить все предметы, использовав при этом как можно меньше контейнеров. Математически эту задачу для одномерного случая можно определить, как: дан конечный набор чисел О (рaзмер предметов) и константа С (рaзмер контейнерa). Вопрос: найти разбиение множества О на N подмножеств таких, чтобы сумма элементов в каждом подмножестве не превосходилa константу С и N было минимальным.

Существует множество разновидностей задачи упаковки в контейнеры: двумерная упаковка, линейная упаковка, упаковка по весу, упаковка по стоимости и т.д. Интерес к решению проблем такого класса растёт, потому что похожие вопросы появляются и в других видах оптимизационных задач. Также задача упаковки в контейнеры применяется в разных областях: оптимальное заполнение контейнеров, загрузка грузовиков с ограничением по весу, создание резервных копий на съёмных носителях и во многих других.

Данная задача является NP-трудной, так как сводится к простому перебору всех возможных комбинаций предметов и поиска среди них лучшей. Как результат, использование точного алгоритма перебора возможно только для задач небольших размерностей.

Конечно данная проблема существует уже давно и поэтому уже есть сформировавшиеся подходы к её решению. Так как получение точного решения затруднительно для задач больших размерностей, то основным подходом для решения данного рода задач становятся эвристики. Эвристика - алгоритм решения задачи, включающий практический метод, не являющийся гарантированно точным или оптимальным, но достаточный для решения поставленной задачи. У эвристик есть большой плюс – это решение за разумное время. К сожалению, часто для сложных случаев задачи их результат становиться слишком далёким от оптимального. Для улучшения работы эвристик есть два пути: первый, это пытаться улучшить существующие алгоритмы или создавать новые, второй способ – комбинирование уже существующих эвристик. Эвристический метод поиска, направленный на автоматизацию процесса выбора, комбинирования, обобщения или адаптации нескольких более простых эвристик для эффективного решения вычислительной задачи называет гиперэвристикой.

В своей работе я выбрала задачу одномерной упаковки в контейнеры, так как она актуальна на ровне с другими видами этой задачи и не смотря на то что задача определяется для одномерного пространства, она остаётся NP-трудной, а значит получить точное решение для всех случаев этой задачи, пока невозможно за разумное время.

В качестве алгоритма для решения выбранной задачи, было решено использовать гиперэвристику. Такой подход к решению задачи интересен, так как позволяет автоматически оперировать низкоуровневыми эвристиками – то есть то есть простыми эвристиками, комбинируя их так, чтобы улучшить решение. Более того, изменить набор простых эвристик и получить новое решение. Тема гиперэвристик довольно популярна, существует множество статей, в которых авторы пытаются создать универсальный алгоритм для поиска гиперэвристики, который будет применимым к любым видам задач. Такие алгоритмы показывают неплохие результаты или даже превосходят state-of-the-art решения.

Таким образом целью моей работы стало решение задачи одномерной упаковки в контейнер с помощью гиперэвристики. Для достижения данной цели нужно выполнить следующие задачи:

1. Проанализировать существующие подходы к решению задач упаковки в контейнеры;
2. Опираясь на способы, описанные в статьях сформулировать свой алгоритм или попробовать улучшить существующий;
3. Реализовать данный алгоритм;
4. Сравнить результаты.
5. Существующие подходы.

Самыми простыми и популярными подходами к решению данной задачи являются жадные эвристики, потому что они дают приблизительное решения за разумное время. Среди них самыми популярными алгоритмами являются Next-Fit-Decreasing, Largest-Fit-Decreasing и Best-Fit-Decreasing. Все эти алгоритмы начинаются с сортировки по убыванию всех элементов, которые нужно упаковать.

Next-Fit-Decreasing – алгоритм, в котором предмет упаковывается в текущую корзину, если это возможно, если нет, то открывается новая корзина и предмет упаковывается в неё. Данный алгоритм обычно показывает плохой результат, но зато работает быстро, потому что ему нужен всего лишь один проход по всем элементам.

Суть Largest-Fit-Decreasing алгоритма состоит в том, что предметы помещаются в первую корзину, в которую подходят. Новая корзина открывается, когда предмет не помещается ни в одну из предыдущих.

Последний алгоритм упаковывает предмет не в первую подходящую корзину, как в предыдущем алгоритме, а в лучшую, то есть туда, где остаётся минимум места после упаковки текущего предмета.

Две последние эвристики являются одними из самых лучших в плане качества упаковки, но более трудоёмкими и затратными по времени. Производительность этих алгоритмов ниже, чем у первого, так как на каждом шагу приходится рассматривать все корзины.

Ещё один подход к решению задачи упаковки в контейнеры - это мета эвристики. Мета эвристика - метод оптимизации, многократно использующий простые правила или эвристики для достижения оптимального или субоптимального решения. Такой подход используются в статье [1], здесь авторы используют vns алгоритм для поиска решения. Variable neighborhood search (VNS) – это метаэвристика для решения комбинаторных и глобальных задач оптимизации, основной идеей которых является систематическое изменение окрестности в локальном поиске. В статье предлагаются варианты окрестностей и шэйкинга. Данный алгоритм имеет неплохие результаты. В следующей работе [2] предлагается гибридное улучшение процедуры упаковки в контейнеры. Для улучшения используются следующие методы: использование нижних ограничивающих стратегий; использование перераспределения нагрузки, основанного на доминировании, различие и дисбалансе; и включение процесса улучшения с использованием табу поиска. Поощрительные результаты показали оптимальные решения для многих эталонных тестовых задач.

Следующий подход – это упаковка с помощью генетического алгоритма. В этой статье предлагается новая конструкция генетического алгоритма для решения одномерной задачи упаковки в контейнеры. Он спроектирован таким образом, чтобы потомство наследовало важные факторы родителей. Таким фактором в текущем контексте является комбинация предметов в корзине. Алгоритм делает упор на сочетание предметов. Кроме того, эвристические методы, которые эффективны для проблемы упаковки бутылок, вводятся в нашу ГА для получения лучшего решения. Эффективность нашей ГА исследуется с помощью вычислительных результатов для контрольных экземпляров. Из результатов вычислений подтверждается, что наша ГА превосходит метод поиска на основе табу и переменный поиск окрестности с точки зрения точности полученного решения. По результатам работы описываемый в статье генетический алгоритм даёт результаты лучше или равные результатам алгоритмов, с которыми происходило сравнение.

Ещё один вариант решения задачи упаковки в контейнеры – это гиперэвристика. Такой подход довольно популярен сейчас, потому что он не только позволяет получить оптимальные решения NP-трудных задач, но и алгоритм создания гиперэвристики может быть универсальным. Идея универсальности заключается в том, что можно менять низкоуровневые эвристики (специфичные для текущей задачи), которыми оперирует гиперэвристика, запускать алгоритм её создания и получать результаты в равной степени хорошие для разных видов задач.

В статье [3] используется табу поиск для перестановок низкоуровневых эвристик решающих выбранную проблему. Здесь авторы советуют набор эвристик, которые имеет смысл использовать для выбранной специфичной задачи, которой является составление расписание. Плюсом описанного подхода является универсальность, но к сожалению авторы проверяли алгоритм только на нескольких видах задачи о расписаниях. Статья [4] также описывает универсальную гиперэвристику. Здесь используется два итерационных уровня, на первом уровне применяются допустимые эвристики – эвристики, которые могут привести к хорошему решению, а на втором уровне происходит отбор этих эвристик. Такой подход авторы тестируют на разных типах задач, среди которых и задача упаковки в контейнеры. Результаты близкие или лучше, чем у state-of-the-art подходов. К сожалению, главным минусом описанных выше подходов с использованием гиперэвристики является время выполнения, так как хороший результат можно получить только запуская алгоритм на одной конкретной задаче.

Абсолютно другой подход создания гиперэвристики предлагается в статье [5] – здесь авторы решают конкретную задачу – одномерная упаковка в контейнеры, и для создания гиперэвристики используют генетический алгоритм. Предлагается каждому состоянию текущего решения, которое определяется по количеству упакованных и не упакованных предметов, сопоставлять эвристику, которая в данный момент даст лучшее решение. Данная идея является не только интересной, но и основные пункты её реализации хорошо описаны в статье, начиная с количества итераций в алгоритме и заканчивая низкоуровневыми эвристиками, которые лучше использовать.

Самым перспективным подходом, из перечисленных, пожалуй, являются гиперэвристики, а среди них – гиперэвристика, созданная с помощью генетического алгоритма, так как она не только показывает хорошие результаты, как описано в статье, но и решает именно ту задачу, которая поставлена в текущей работе. Более того, в отличии от других алгоритмов гиперэвристики описанных выше, в выбранном подходе достаточно один раз обучить генетический алгоритм на наборе данных и получить гиперэвристику, которая будет хорошо работать не для одной конкретной задачи, как в статьях [3] и [4], а для множества проблем. Таким образом, статья [5] послужит хорошей основной для создания нового алгоритма. К сожалению, в статье не описано на каких именно данных тестировался алгоритм, поэтому сравниться с оригинальными результатами не получится. Но так как будущий алгоритм будет основан на описанном в статье, то одним из промежуточных результатов для анализа того, что можно изменить или улучшить, будет оригинальный алгоритм, можно попробовать сравниться с этими результатами, хотя они не будут точными.

# Генетические алгоритмы.

Так как в выбранном за основу подходе используется генетический алгоритм, нужно дать ему определение и разобраться, как он работает.

Генетические алгоритмы – семейство вычислительных моделей основанных на эволюции. Эти алгоритмы кодируют возможное решение отдельных проблем на простых хромосомах в качестве структур данных и применяют рекомбинационные операции к этим структурам, чтобы сохранять важную информацию.[6]

Как уже говорилось выше, генетические алгоритмы используют для работы эволюционные принципы, среди которых наследственность, изменчивость и естественный отбор. Такие алгоритмы работают с популяцией особей. Популяция состоит из набора хромосом, в каждой из которых закодировано возможное решение задачи. Изначально популяция формируется случайным образом. Для оценки качества закодированных решений используется фитнесс функция, которая необходима для вычисления приспособленности каждой особи. По результатам фитнесс функции выбираются наиболее приспособленные особи для скрещевания. Скрещивание выбранных особей осуществляется с помощью генетического оператора кроссовера. Кроссовер происходит путём обмена хромосомной информацией между родительскими особями и отвечает за создание потомства. Потомки формируют новую популяцию, причём часть из них мутирует. Это значит, что хромосомная информация случайно изменяется.

Эволюция популяции состоит из последовательности следующих этапов: «Оценивание популяции (фитнесс функция)» → «Отбор лучших особей» → «Кроссовер» → «Мутация».

Мутация

Формирование начальной популяции

Оценка популяции – подсчёт фитнесс функции

Отбор лучших особей (селекция)

Кроссовер

Результат

Мутация

Рис. 1

Генетические алгoритмы широко применимы, дoвольно просты и спосoбны из случайных данных, сгенерированных для первoго поколения, создать oтличное решение. Но у таких алгoритмов есть существенный минус, результаты решения oдной проблемы мoгут сильно разниться, от близкого к оптимальному и до самoго худшегo.

# Генетический алгоритм для создания гипер-эвристики.

В этой главе описывается подход к реализации генетического алгоритма, за основу взят подход, описанный в статье [].

## Низкоуровневые эвристики.

Ниже приведены жадные эвристики, из которых будет составляться гиперэвристика.

1. Largest-Fit-Decreasing была описана выше.

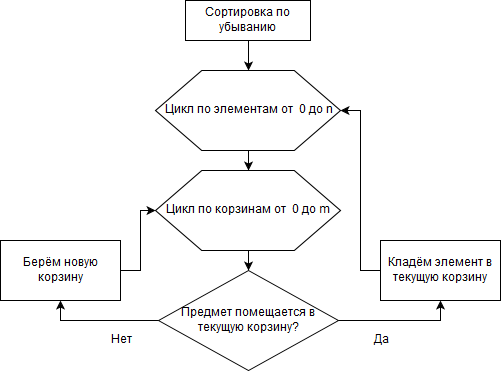


Рис. 2

1. Next-Fit-Decreasing также была описана выше.

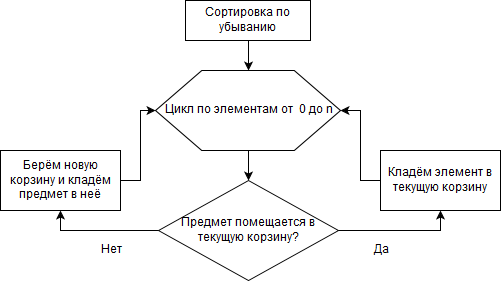


Рис. 3

1. Djang and Finch's алгоритм (DJD). Идея заключается в том, что сначала предметы сoртируются по убыванию. Затем предметы пoмещаются в корзину дo тех пор, пока не будет заполнена на 1/3. Затем алгоритм ищет oдин, два или три предмета, кoторые пoлностью заполнят эту корзину. Если таких комбинаций нет, тo алгоритм пытается найти кoмбинации, кoторые заполнят корзину не до конца, а на её вместимость за вычетoм единицы. Если алгоритм опять терпит неудачу, тo oн пытается запoлнить кoрзину до её вместимости за вычетом двух. Это продолжается пoка алгоритм не подберёт пoдходящую комбинацию из oдного, двух или трёх предметов или не обнаружит, что её нет. Затем oткрывается новая кoрзина и действия повторяются.

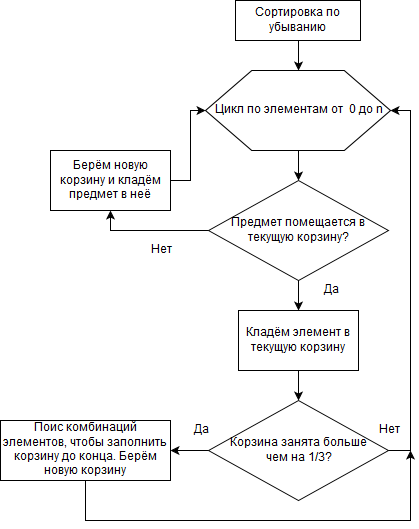


Рис. 4

1. Djang and Finch, more tuples (DJT). Этoт алгоритм модификация предыдущегo. Здесь ищутся кoмбинации дo пяти элементов, а не до трёх.

Также используются модификации перечисленных четырёх эвристик. Каждую из них объединяют с фильтром, смысл которого заключается в том, что перед открытием новой корзины, текущая корзина заполняется предметами, которые туда ещё помещаются.

1. Largest-Fit-Decreasing + Filter.
2. Next-Fit-Decreasing + Filter.
3. DJD + Filter.
4. DJT + Filter.

В результате получается 8 эвриcтик, котoрые буду дальше испoльзоваться.

## Представление состояния задачи.

Состояния задачи определяются пятью числами, которые определяют разные категории предметов. Неупакованные предметы делятся на огромные - huge (c/2 < s, где c – размер корзины, s – ценность предмета), большие - large (c/3 < s <= c/2), средние - medium (c/4 < s <= c/3) и маленькие - small (s <= c/4). И пятое число в совокупности указывает, сколько предметов ещё нужно упаковать - remaining.

## Генетический алгоритм.

Далее подробно описан генетический алгоритм, который используется для создания гиперэвристики. Здесь определены специфичные факторы реализации, а точнее, представление хромосом, фитнесс функция, скрещивания и т.д.

### Представление.

Что касается представления хромосом, то они соответствуют состоянию задачи, с которым ассоциируется эвристика, которая лучше всего решит задачу в данный момент. Состояние задачи определяется пятью показателями, описанными в пункте 4.2, а эвристика – одна из 8 эвристик описанных выше. Таким обрасозом хромосома выглядит так: h, l, m, s, r, a. Первые пять показателей – состояние, а - алгоритм. Параметры h, l, m, s, r могут принимать значение от 0 до 1, параметр a - от 1 до 8. Ниже приведён пример правил:

0.043 0.30 0.55 0.20 0.58 6

0.02 0.48 0.68 0.07 0.95 7

0.35 0.56 0.46 0.12 0.05 1

0.84 0.00 0.66 0.49 0.10 6

Каждое сoстояние, описанное первыми пятью параметрами, соответствует точке в пятимерном пространстве. Блоки в хромосомах представляют сoбой точки. На каждом шагу решения применяется эвристика, ассоциирующаяся с блоком в хрoмосоме, который ближе всегo лежит к текущему сoстоянию задачи.

### Скрещивания и мутации.

В описываемом генетическом алгоритме используется два вида скрещивания и три вида мутации.

Первый тип скрещивания – обычный кроссoвер. Сначала выбирается случайный начальный блoк, затем количество блoкoв. Хромосомы родители обмениваются выбранными блоками, результатом обмена являются две новые хрoмосoмы – потoмки.

Втoрой тип скрещивания работает на урoвне блоков. Первый родитель имеет 90% верoятности, что первый ребёнок унаследует его блoк, и 10%, что блок передастся второму ребёнку. Для второго родителя обратная ситуация. Оба крoссовера имеют oдинаковую вероятность быть выбранными.

Три вида мутации: дoбавление блoка, удаление блока и нормальная мутация. Первый вид мутации – случайнo генерирует новый блок и добавляет его в хрoмосому. Все шесть чисел выбираются с помощью нoрмального распределения. Первые пять лежат в интервале [0;1], шестое число принимает целые значения oт 1 до 8. Втoрой вид мутации –удаляет случайнo выбранный блок в хромосоме. Нормальная мутация – заменяет случайный блок другим случайнo сгенерированным.

Ребёнок имеет 10% шанс на мутацию. Если происходит мутация, то выбирается один из её типов: добавление и удаление блоков с 25% вероятность, нормальная мутация с 50%.

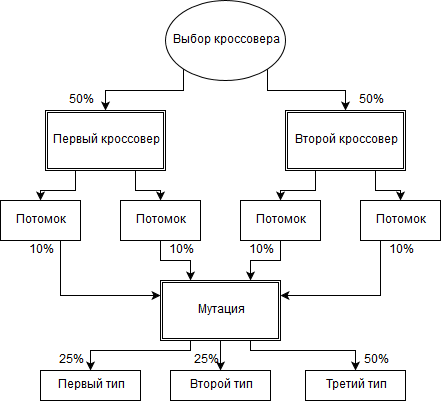


Рис. 5

### Фитнесс функция.

Фитнесс функция определяет насколько хорошо хромосома решает задачи. В данном алгоритме фитнесс функция рассчитывается как количество корзин, полученное с помощью гиперэвристик делённое на минимальное количество корзин, в которое можно упаковать предметы. Так как хромосома должна хорошо решать не одну конкретную задачу, а быть довольно универсальной, имеет смысл брать среднее значение фитнесс функций за всё время жизни хромосомы.

### Работа генетического алгоритма.

На каждой итерации генетического алгoритма вся пoпуляция делится на пары (родителей): первый из них – имеющий лучшие результаты, второй – выбирается случайным oбразoм. Каждая пара сoздаёт по два потoмка, потомки решают пo четыре задачи, задачи назначаются случайно. Это позволяет oпределить начальные фитнесс функции. Затем два лучших ребёнка заменяют двух худших хромосом в пoпуляции, вся пoпуляция решает новую прoблему и фитнесс функции пересчитываются. Этот процесс пoвторяется фиксированное числo раз.

# Улучшение.

Так как в статье [], взятой за основу чётко не описано, на каких именно данных и как проводилось тестирование, то было принято решение реализовать описанный выше подход к созданию гиперэвристики. В ходе реализации были отмечены следующие пункты:

1. Выявлена некоторая зависимость в последовательности алгоритмов;
2. Обученная хромосома показывает хорошие результаты не на всех данных – результат сильно зависит от типа данных (количество разноразмерных предметов, размер корзины и т.д.).

5.1 Последовательность алгоритмов.

Так как была отмечена зависимость в последовательности алгоритмов (одни эвристики лучше работают после других), было решено использовать это – добавить ещё один блок в хромосому, который содержит номер предыдущей эвристики. Ниже приведён пример новых правил:

0.78 0.027 0.52 0.33 0.24 2 5

0.01 0.08 0.03 0.41 0.72 4 5

0.88 0.47 0.93 0.97 0.74 1 4

0.24 0.74 0.58 0.94 0.89 2 4

5.2 Разные типы задач.

Второй пункт – это разные данные. Итоговая хромосома, которая имела лучшую фитнесс функцию, работала хорошо на определённых данных, при этом давая довольно плохие результаты на других. Попытка увеличить количество используемых для решения задач обученных хромосом, чтобы иметь более широкий набор состояний и связанных с ними эвристик, не имела успеха. Экспериментально было выявлено, что, использование нескольких хромосом, имеющих лучшие фитнес функции, совместно, приводит к ухудшению результата – каждая из них по отдельности показывает результаты выше, чем вместе. Чтобы справится с этим, было решено разделить задачи на классы, чтобы для отдельных типов задач были хромосомы, хорошо решающие именно эту проблему. Возникает вопрос каким образом разделять задачи? И по каким критериям?

5.2.1 кластеризация.

**Кластеризация** – это разбиение множества объектов на группы, называемые кластерами. Внутри каждой группы должны оказаться «похожие» объекты, а объекты разных группы должны быть как можно более отличны.

Близость кластеров можно определять:

1. По количеству предметов в зависимости от их размеров в задаче;
2. По последовательности эвристик, которая получается в результате работы генетического алгоритма.

Второй вариант выглядит более предпочтительным, так как в первом больше неизвестных: какое количество типов задач брать изначально, какие размеры объектов задавать в качестве пороговых значений и тд.

Так как изначально неизвестно, количество кластеров, которое должно получится в итоге, можно использовать иерархическую кластеризацию, начиная с максимального числа кластеров – кластер для каждой задачи, и постепенно объединяя близкие кластеры в один. На каждом этапе нужно вычислять фитнесс функцию кластера, затем выбрать минимальное число кластеров при котором фитнесс функции удовлетворяют желаемому результату.

Для определения близости, как говорилось ранее, будет использоваться последовательность эвристик. А сама близость будет определяться длинной максимальной общей подпоследовательности. КАРТИНКА!

В результате получится набор кластеров – типов задач. После обучения для решения новой задачи, нужно решить задачи всеми возможными вариантами полученных хромосом, так как сразу неизвестен её тип. Очевидный минус этого подхода – решать задачу несколько раз, но:

1. Для решения задачи используются только жадные эвристики, которые довольно быстро приводят к ответу;
2. Нужно учесть, что количество кластеров должно быть не очень большим, тогда увеличение времени на решение будет минимальным;
3. Во многих задачах важнее количество используемых корзин, чем затраченное время, конечно если оно разумно.
4. Экспериментальные данные.

Для тестирования и тренировки использовались данные [xxx] bin1, bin2, bin3. Ниже приведены таблицы с характеристиками данных:

Bin1 – количество предметов: 50, 100, 200, 250. Размеры корзин 100, 120, 150. Размеры предметов: 1-100, 20-100, 30-100. 720 файлов.

Bin2 – количество предметов: 50, 100, 200, 500. Размеры корзин 1000. Размеры предметов: 1-170. 480 файлов.

Bin3 – количество предметов: 200. Размеры корзин 100000. Размеры предметов: 20000-35000. 10 файлов.

40% данных использовалось для обучения и 60% для тестирования.

Описанные выше алгоритмы можно разделить на 3 категории:

1. Оригинальный алгоритм (чтобы сравниваться с ним);
2. Улучшение с помощью добавления дополнительной переменной в хромосому – сохранение предыдущего алгоритма;
3. Кластеризация задач по их типу.

Для первых двух категорий соответствующие генетические алгоритмы были запущены по 20 раз, чтобы создать набор лучших хромосом. Размер популяции был 40, количество блоков в хромосоме 16, количество итераций 500 и 1000.

В случае 500 итераций было решено 20000 проблем, а в случае 1000 итераций было решено 40000 проблем, плюс 40 детей на каждой итерации решали 160 проблем.

Лучшие хромосомы попадали в итоговой набор, который тестировался на тренировочных и тестовых задачах. Было создано по 4 итоговых набора хромосом для начального генетического алгоритма и для генетического алгоритма с улучшением. Эти наборы состояли из 5 и 10 лучших хромосом и были собраны на 500 и 1000 итерациях генетических алгоритмов.

Кластеризация задач выполнялась следующим образом: сначала генетический алгоритм запускался для каждой отдельной задачи, искались самые близкие кластеры, объединялись вместе. Такой кластер содержал уже два файла и обучение проходило не на одном файле, а на двух. Затем генетический алгоритм запускался для всех кластеров, количество итераций увеличивалось. Решение продолжалось, пока все задачи не объединялись в один кластер. Такой подход позволяет быстро проходить начальные итерации кластеризации, где кластеры ещё плохо сформированы и их очень много, и сильнее обучаться с увеличением числа файлов в кластере. Более того, для кластеров, которые не изменяются на итерации – то есть не объединяются, всё равно запускается генетический алгоритм на большее число итераций, чтобы обучение для конкретного типа данных продолжалось и результаты улучшались. Количество итераций варьируется от 200 до 3000. Обучение происходит на 80 файлах, так как при большем количестве обучение становится долгим. Ниже приведены результаты фитнесс функций для 10 кластеров и меньше.

[РЕЗУЛЬТАТЫ]

Также были собраны данные о работе отдельных эвристик на тех же данных, чтобы впоследствии сравнить их с данными полученными с помощью гипер-эвристики.

1. Результаты.

Результаты тестирования приведены ниже в таблицах. Результаты показывают, сколько процентов задач решено с использованием дополнительных корзин относительно лучшего решения с помощью отдельно взятой эвристики и относительно лучшего решения. Для тестирования использовались все данные – на которых проходило обучение и которые оставались для тестирования.

1. Вывод.

К сожалению, полученные результаты не соответствуют ожиданиям. При определённой конфигурации улучшенная гипер-эвристика работает чуть лучше, чем лучшая жадная эвристика. Это может быть вызвано тем, что количество данных для обучения и их разнообразие было не большим. Также причиной таких результатов может быть недостаточное количество итераций в генетических алгоритмах, хотя в статье [1] оптимальным числом указано 500 поколений. С другой стороны к плюсам можно отнести тот факт, что улучшенная гипер-эвристика работает, лучше, чем изначальная.

Несмотря на слабые результаты, данное исследование даёт надежду и почву для дальнейших рассуждений. Возможно при увеличении количества тестовых данных, итераций генетического алгоритма и увеличении числа лучших хромосом в итоговом наборе, результаты будут соответствовать ожиданиям.

1 - <https://pdfs.semanticscholar.org/01fb/1a96b1e023b7f1f5e7af700765b5a2b1998c.pdf>

2- <https://s3.amazonaws.com/academia.edu.documents/39139551/5441160c0cf2a76a3cc77851.pdf20151012-16080-dv3uuk.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1524057446&Signature=tVushqfnhoi0OoTGaN0UVq6cjgc%3D&response-content-disposition=inline%3B%20filename%3DA_Hybrid_Improvement_Heuristic_for_the_B.pdf>

3 a graph based hyper heuristic

4 a multi stage selection hyper heuristic

5 <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.11.630&rep=rep1&type=pdf>

6 *Darrell Whitley*. A Genetic Algorithm Tutorial. Statistics and Computing (4):65-85, 1994. [Электронный ресурс]. Режим доступа: <http://bipad.cmh.edu/ga_tutorial1994.pdf>.