1. Здравствуйте, тема моей ВКР Упаковка в контейнеры. Начнём с описания задачи. Она заключается в упаковке предметов заранее определённой формы в конечное число контейнеров предопределённой формы таким способом, чтобы количество используемых контейнеров было наименьшим. С математической точки зрения для одномерного случая данную задачу можно сформулировать как: дан конечный набор чисел О (размер предметов) и константа С (размер контейнера). Вопрос: найти разбиение множества О на N подмножеств таких, чтобы сумма элементов в каждом подмножестве не превосходила константу С и N было минимальным. В своей работе рассматривала именно одномерный случай задачи. Интерес к решению проблем такого класса растёт, потому что 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, эта проблема не нова и уже существует много различных подходов к её решению. Самый простой и популярный – это метаэвристики, которые дают приближённое решение. К сожалению, эти решения часто далеки от отптимальных. Одним из способов увеличения точности решения является использование не одной отдельной метаэвристики, а использование комбинации эвристик. Эвристический метод поиска, направленный на автоматизацию процесса выбора, комбинирования, обобщения или адаптации нескольких более простых эвристик для эффективного решения вычислительной задачи называется гиперэвристикой. В своей работе я использую именно гипер-эвристику для решения задачи упаковки в контейнеры.

2. В своей курсовой работе я уже использовала гиперэвристику для решения выбранной задачи. Алгоритм был создан на основе \* статьи. В ней предлагалось каждому состоянию задачи, которое определялось по n параметрам, сопоставлять номер эвристики, которая в данный момент даст лучшее решение. В статье использовался генетический алгоритм, чтобы создать гиперэвристику, с помощью которой потом решались задачи. Получалось, что каждое состояние задачи, это точка в n-мерном пространстве, которой соответствует эвристика. Когда задача решалась, определялось её текущее состояние – точка, затем вычислялась самая близкая точка, которой соответствует эвристика, которая в дальнейшем и применялась для решения. В этом году, я хочу рассмотреть гиперэвристику, которая сразу определяет последовательность эвристик. Существует множество статей и исследований, целью которых является именно создание такой последовательности.

3. Я просмотрела большое количество статей на эту тему, среди которых \*\*\*. В этих статьях авторы пытаются создавать универсальные алгоритмы поиска гиперэвристики. Чтобы можно было менять низкоуровневые эвристики на эвристики подходящие конкретной задачи, запускать их алгоритм и он будет создавать гиперэвристики в равной степени успешные для решения разных задач. В каждой статье в основе лежит в той или иной степени локальный поиск. В результате анализа этих статей я выделила 2 алгоритма, которые с некоторыми изменениями и поправками могут быть применены к моей задаче и показать хорошие результаты.

4. Первый из алгоритмов поиска последовательностей эвристик прост: по сути это простой локалсёрч, на каждой его итерации решаются все тестовые задачи. В качестве окрестностей используются: последовательное добавление всех возможных низкоуровневых эвристик во все возможные позиции; последовательное удаление одного элемента; swap - всевозможные попарные перестановки, и полный переворот последовательности эвристик. Из функции локального поиска выход осуществляется при обнаружении первого улучшения. В качестве шэйкинга используется рандомное удаление.

5. Второй алгоритм это multistage level. Здесь для поиска гиперэвристики используются два шага - stage. Изначально набор low level эвристик(LLH) состоит из всех выбранных эвристик и всех возможных их попарных комбинаций. Каждой такой low level эвристике сопоставлен счёт. Изначально у одинарных эвристик этот счёт 1, у комбинированных 0. Этот счёт определяет вероятность использования эвристики в решении. Также определены PS2HH – он определяет вероятность перехода на второй шаг этого multi-stage алгоритма. Также определён круговой массив C, в дальнейшем значения из этого массива используются для выхода из локального минимума, дальше будет видно, как именно этот массив С используется. И определён counter который определяет перемещения по массиву С.

Алгоритм начинается со своего первого шага – stage. Здесь выбирается рандомная эвристика, которая дальше будет использоваться на этом шаге. Выбирается она конечно не прям вот чтобы рандомно, а с использование рулетки. Выбор основан на скоре каждой эвристики и вероятность выбора эвристики это её счёт делённый на сумму всех счетов. Затем если у нас довольно долго не было улучшений, мы апдэйтим значение элипсон, которое помогает нам выйти из локального минимума. Здесь видно, что если вдруг решение текущей последовательности хуже лучшего в пределах элипсон, то мы не будем отвергать такое решение, а будем перемещаться туда, чтобы выйти из локального минимума, если мы в нём. Сама элипсон расчитывается вот по такой формуле \* это формула - !!!. Собственно здесь applyheuristic добавляет к уже имеющейся последовательности эвристик нашу рандомную эвристику и проверяется целевая функция – улучшилось ли решение относительно текущего. Применяем эту эвристику тау раз, и выходим из первого шага алгоритма, проапдэйтив лучшие результаты, если что-то было улучшено. Выйдя с этого шага алгоритма проверяется коунтер, если он дошёл до конца массива С, то апдэйтим его.

Затем с вероятностью PS2HH начинается второй шаг алгоритма. Во второй части алгоритма, мы опять пересчитываем эпсилон. Здесь задача решается с помощью текущей последовательности эвристик и последовательным добавлением к ней всех вариантов эвристик из LLH по тау раз. Значения всех лучших решений и количество шагов, сколько раз мы применили эвристику без ухудшения результата подряд, сохраняются в ParetoArchive.

Возвращаемся к основному алгоритму. Апдэйтим счета эвристик основываясь на ParetoArchive. Из этого архива берутся 3 лучшие эвристики, которые показали хороший результат и при этом могут применятся большее число раз не ухудшая решения, чем другие. Затем, если эвристика встречается 1 раз, то её счёт сэтится в 1, если два, то в 2. Пример Если у нас лучшие эвристики LLH1, LLH2; LLH1; LLH3 – то счета будут 2, 1 и 1. Если второй шаг не применяется, то у всех одиночных эвристик счёт становится 1, а у двойных 0. Выполнение multi-stage алгоритма продолжается заданное число раз, и на выходе получаем лучшую последовательность эвристик.

6. В результате своей работы собираюсь сравнить результаты двух только что описанных алгоритмов и гиперэвристики полученной с помощью генетического алгоритма, реализованной в том году. Есть большая вероятность, что описанные только что алгоритмы будут ещё подвергаться изменениям.