МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Нижегородский государственный университет им. Н.И. Лобачевского

Е.А.Неймарк

Лабораторные работы по курсу «Эволюционно-генетические алгоритмы»

Учебно-методическое пособие

Рекомендовано методической комиссией ИИТММ
для студентов ННГУ, обучающихся по направлению подготовки
090303 «Прикладная информатика»

Нижний Новгород
2018

УДК 004.02
Н-45

Н-45 Неймарк Е.А Лабораторные работы по курсу «Эволюционно-генетические алгоритмы».

Данное пособие направлено на выработку практического навыка программирования алгоритмов, проведения численного эксперимента и составления отчета, согласно требованиям.

В пособии представлено 8 алгоритмов для реализации. Данные алгоритмы применяются для поиска решения дискретных задач оптимизации. Каждый алгоритм представлен в виде пошаговой схемы выполнения. Студент в ходе работы должен предоставить программную реализацию каждого алгоритма, причем программа в ходе выполнения обязательно выводит промежуточные данные работы алгоритма (требования к каждой реализации представлены в задании). По результатам выполнения работы должен быть представлен отчет. При оценке качества выполнения работы оценивается как программная реализация, так и отчет.

Десятым алгоритмом для реализации является эволюционно-генетический алгоритм. Его схема и подробный разбор производится в лекциях. Для тестирования эволюционно-генетического алгоритма предоставлены дискретные задачи оптимизации 2х типов: задача коммивояжера (симметричная евклидова) и задача о целочисленном ранце. По результатам реализации и тестирования эволюционно-генетического алгоритма пишется подробный отчет, включающий подробное описание реализованного алгоритма, описание проведенного эксперимента, приведение сырых данных по эксперименту, обработки данных и составление выводов по результатам эксперимента.

Итогом выполнения практических занятий является 10 отчетов о выполненных работах. В ходе работы отрабатываются навыки понимания алгоритмического подхода и реализации алгоритмов на высокоуровневых языках программирования. Навыки работы с требованиями к реализации и их выполнение. Навыки проведения численного эксперимента с целью верификации работы программы, реализующей алгоритм. А также навыки составления плана эксперимента, обработки экспериментальных данных и составления выводов по результатам эксперимента.

Пособие предназначено для студентов ИИТММ направления подготовки «Прикладная информатика», изучающих курс «Эволюционно-генетические алгоритмы».

Оглавление

Занятие 1. Метод Монте-Карло

Метод Монте-Карло является методом случайного поиска. Существенно сокращая вычислительные затраты, можно получить достаточно неплохое решение. Однако, никаких оценок качества полученного решения нет.

Суть метода – получить несколько случайно выбранных из пространства поиска решений, оценить каждое и выбрать из них наилучшее. Объём выборки является параметром алгоритма. Полученное таким образом решение считается общим решением задачи.

Метод является случайным и не точным.

Для начала работы нам надо задать пространство поиска (ПП) $\boldsymbol{S = \{s_i, i = \overline{1, n}\}, |S| = n,}$ пространство писка состоит из кодировок $s_i, i = \overline{1, n}$. **L** – длина кодировки $s_i \in S$. **B** – алфавит кодирования, если не указано иное **B={0,1}** – бинарный алфавит.

А также надо задать ландшафт приспособленности, т.е. каждой кодировке поставить в соответствие некоторое значение приспособленности $\boldsymbol{\mu(s_i)}$, $\boldsymbol{i = \overline{1, n}}$. По определению приспособленности: $\mu(s_i) > 0$.

N – параметр алгоритма, обозначает количество шагов алгоритма до получения решения, считаемого приемлемым.

Алгоритм:

1. i=0, L ,N, max=0, maxS – пустая кодировка
2. **если** i<N
   то $s_i \in S$ – **случайный выбор, переход** =>3
   иначе переход =>выход
3. **если** max<$\mu(s_i)$
   то max=$\mu(s_i)$, maxS=s_i
4. i=i+1, **переход** =>2

выход: искомое решение: **maxS**, его приспособленность **max.**

Задание:

1. Нарисовать по алгоритму блок-схему
2. Задать пространство поиска. L=15. Кодировки s_i задаются как бинарные значения (строки) соответствующих натуральных чисел $x_i = \overline{0, 2^L - 1}$.
3. Задать ландшафт приспособленности
   a. Первый вариант: случайное задание приспособленностей (в десятичном счислении)
   b. Второй вариант: приспособленность соответствует натуральному значению бинарного кода
   c. Третий вариант: приспособленность вычисляется как квадратичная функция: $\mu(s_i) = (x_i - 2^{L-1})^2$
4. Запрограммировать метод Монте-Карло. При запуске должны выводиться следующие данные: для каждого шага – номер шага $\boldsymbol{i}$, текущие ***max*** и ***maxS***, выбираемое $\boldsymbol{s_i}$, если происходит смена ***max*** и ***maxS***, показать это.
5. Сделать 5-10 запусков программы. Сделать отчет.

В программной реализации оцениваются следующие пункты:

1. соответствие алгоритму
2. вывод на консоль:
   1) задан ландшафт поиска (достаточно вывести 32 кодировки + их приспособленности в виде списка)
   2) выводится номер шага
   3) выводится текущий максимум приспособленности
   4) выводится текущая лучшая кодировка
   5) выводится выбираемая на шаге кодировка
   6) выводится приспособленность выбираемой на шаге кодировки
   7) показана смена максимума
   8) выведено итоговое решение алгоритма

Отчет содержит следующие пункты:

1. блок-схема алгоритма
2. выводится ландшафт, достаточно 32 кодировки + их приспособленности
3. результаты запусков (минимум 5) – копии результатов с экрана
4. все лучшие решения по запускам – в виде таблицы
5. выделено лучшее решение среди всех запусков (кодировка и ее приспособленность)
6. сделаны выводы по результатам работы алгоритма
7. соответствие кода и результатов исполнения
8. приложен код

Занятие 2. Метод восхождения на холм. Поиск в глубину

Методы восхождения на холм (МВХ) также являются методами поисковой оптимизации. Они относятся к случайным и также, как и метод Монте-Карло, не гарантируют получения оптимального решения и не дают оценки качества получаемых решений.

Методы являются одношаговыми, т.е. в каждый момент времени у нас есть текущее решение, при остановке это решение считается лучшим найденным. Эта группа методов относится также к методам слепого поиска, т.е. алгоритм не имеет сведений о характере поведения оптимизируемой функции и получает ее только с помощью замеров.

Суть метода восхождения на холм – на каждом шаге выбирается текущая кодировка, оцениваются приспособленности кодировок в ее окрестности, если найдена кодировка с приспособленностью выше, чем у текущей, то она становится текущей на следующем шаге.

Поиск в глубину Метод поиска в глубину является одной из разновидностей МВХ. В данном случае для текущей кодировки на каждом шаге происходит оценка случайной кодировки из ее окрестности, если ее приспособленность выше, чем у текущей, то она делается текущей на следующем шаге, если нет – на следующем шаге берется следующая случайная кодировка из окрестности. Если все кодировки в окрестности текущей исчерпаны (она лучшая в своей окрестности) – метод останавливается.

Метод является случайным, не точным.

Также, как и для метода Монте-Карло, надо задать пространство поиска и ландшафт. Воспользуемся старыми обозначениями.

N – параметр алгоритма, обозначает количество шагов алгоритма до получения решения, отметим, что реальное число шагов алгоритма может быть меньше, если на каком-то шаге мы нашли локальный оптимум.

Для работы МВХ необходимо определить способ задания окрестности решения (кодировки). Поскольку мы используем бинарный метод кодирования, то мерой близости выберем хэммингово расстояние $h(s_i, s_j) = \sum_{k=1}^{L}(s_i \oplus s_j)_k$ –фактически, это число различающихся битов у двух кодировок. В качестве окрестности $\Omega(s_i)$ для текущей кодировки выберем подмножество кодировок, имеющих всего один отличающийся бит, $\Omega(s_i) = \{s_j \in S, h(s_i, s_j) = 1\}$.

Алгоритм:

1. i=0, L ,N, $s_i \in S$ – **случайный выбор,** maxS=s_i, max=μ(maxS), **определяем** Ω(maxS)
2. **если** i<N
   то переход =>3**,**
   иначе переход =>выход
3. **если** |Ω(maxS)|>0
   то i=i+1, $s_i \in \Omega(\text{maxS})$ – **случайный выбор,** $\Omega(\text{maxS}) = \Omega(\text{maxS})\backslash\{s_i\}$, **переход** =>4 **,**
   иначе переход =>выход
4. **если** max<$\mu(s_i)$
   то maxS=s_i, max=μ(maxS) , **определяем** Ω(maxS)
5. **переход** =>2.

выход: искомое решение: **maxS**, его приспособленность **max.**

Задание:

1. Нарисовать по алгоритму блок-схему
2. Задать пространство поиска. L=5. Кодировки s_i задаются как бинарные значения натуральных чисел $x_i = \overline{0, 2^L - 1}$.
3. Задать ландшафт приспособленности
   a. Первый вариант: приспособленность соответствует натуральному значению бинарного кода
   b. Второй вариант: приспособленность вычисляется как квадратичная функция: $\mu(s_i) = (x_i - 2^{L-1})^2$
   c. Третий вариант: случайное задание приспособленностей
4. Запрограммировать метод восхождения на холм. При запуске должны выводиться следующие данные: для каждого шага – номер шага ***i***, текущие ***max*** и ***maxS***, окрестность $\boldsymbol{\Omega(maxS)}$ – все кодировки и их приспособленности, выбираемое $\boldsymbol{s_i}$**;** если происходит смена ***max*** и ***maxS***, показать это.
5. Сделать 5-10 запусков программы. Сделать отчет.

В программной реализации оцениваются следующие пункты:

1. соответствие алгоритму
2. вывод на консоль:

1) задан ландшафт поиска (достаточно вывести 32 кодировки + их приспособленности в виде списка)
2) выводится номер шага
3) выводится текущий максимум приспособленности
4) выводится текущая лучшая кодировка
5) выводится выбираемая на шаге кодировка
6) выводится приспособленность выбираемой на шаге кодировки
7) выводится полученная на шаге окрестность (все кодировки и их приспособленности)

Отчет содержит следующие пункты:

1. блок-схема алгоритма
2. выводится ландшафт, достаточно 32 кодировки + их приспособленности
3. результаты запусков (минимум 5) – копии результатов с экрана
4. все лучшие решения по запускам – в виде таблицы
5. выделено лучшее решение среди всех запусков (кодировка и ее приспособленность)
6. сделаны выводы по результатам работы алгоритма
7. соответствие кода и результатов исполнения
8. приложен код

Занятие 3. Метод восхождения на холм. Поиск в ширину

Метод поиска в ширину также является разновидностью МВХ. В этом случае для текущей кодировки на каждом шаге происходит оценка не одной случайной кодировки из ее окрестности, а всех кодировок окрестности. Движение продолжается в сторону наилучшей кодировки из всей окрестности. Если все кодировки в окрестности текущей не лучше нее, то пришли в локальный оптимум – метод останавливается.

Метод является случайным, не точным.

Стоит отметить, что в отличие от предыдущего алгоритма, метод поиска в ширину производит не N оценок приспособленности, а больше.

Алгоритм:

1. **i=0,** L ,N, $\mathbf{s_i} \in \mathbf{S}$ **– случайный выбор,** maxS=s_i, max=μ(maxS), определяем Ω(maxS), N
2. **если** i<N
   то maxΩ = $\{s_j : \mu(s_j) = \max\limits_{\forall s_k \in \mathbf{\Omega(maxS)}} \mu(s_k)\}$, **,** i=i+1, **переход** =>3
   иначе переход =>выход
3. **если** max<μ(maxΩ)
   то max=μ(maxΩ), maxS=maxΩ **,** определяем Ω(maxS) **переход** =>2**,**
   иначе переход => выход.

выход: искомое решение: **maxS**, его приспособленность **max**.

Задание:

1. Нарисовать по алгоритму блок-схему

2. Задать пространство поиска. L=5. Кодировки s_i задаются как бинарные значения натуральных чисел $x_i = \overline{0, 2^L - 1}$.
3. Задать ландшафт приспособленности
    a. Первый вариант: приспособленность вычисляется как квадратичная функция: $\mu(s_i) = (x_i - 2^{L-1})^2$, где x_i – значение бинарного кода
    b. Второй вариант: случайное задание приспособленностей
    c. Третий вариант: приспособленность соответствует натуральному значению бинарного кода
4. Запрограммировать метод восхождения на холм. При запуске должны выводиться следующие данные: для каждого шага – номер шага ***i,*** текущие ***max*** и ***maxS***, окрестность $\boldsymbol{\Omega(maxS)}$– все кодировки и их приспособленности, выбираемое $\boldsymbol{s_i}$**;** если происходит смена ***max*** и ***maxS***, показать это.
5. Сделать 5-10 запусков программы. Сделать отчет.

<u>В программной реализации оцениваются следующие пункты:</u>

1. соответствие алгоритму
2. вывод на консоль:
    1) задан ландшафт поиска (достаточно вывести 32 кодировки + их приспособленности в виде списка)
    2) выводится номер шага
    3) выводится текущий максимум приспособленности
    4) выводится текущая лучшая кодировка
    5) выводится выбираемая на шаге кодировка
    6) выводится приспособленность выбираемой на шаге кодировки
    7) выводится полученная на шаге окрестность (все кодировки и их приспособленности)

<u>Отчет содержит следующие пункты:</u>

1. блок-схема алгоритма
2. выводится ландшафт, достаточно 32 кодировки + их приспособленности
3. результаты запусков (минимум 5) – копии результатов с экрана
4. все лучшие решения по запускам – в виде таблицы
5. выделено лучшее решение среди всех запусков (кодировка и ее приспособленность)
6. сделаны выводы по результатам работы алгоритма
7. соответствие кода и результатов исполнения
8. приложен код

<u>Занятие 4. Метод многократного запуска</u>

Поскольку методы восхождения на холм зависят от выбора начальной точки, для улучшения качества получаемого решения используется метод многократного запуска.

То есть методы запускаются несколько раз (**N** раз), из всех запусков выбирается лучшее решение. Каждый метод делает по **n** шагов. Таким образом, решается **N*n** задач распознавания.

N- количество запусков разных методов. $\mathbf{maxS}_i$, $\mathbf{max}_i$ – решение и его приспособленность, полученные из методов.

Алгоритм:

1. **i=0, N,** max=0, maxS – пустая кодировка
2. **если** i<N
   то $k \in \{0,1,2\}$ – случайное число, i=i+1, **переход** =>3
   иначе переход =>выход
3. **если** $k = 0$
   то запустить метод Монте-Карло
   иначе
   если $k = 1$
   то запустить метод поиска в глубину
   иначе запустить метод поиска в ширину
4. **если** $\max_i > max$
   то max=$\max_i$, maxS=$\max S_i$
5. **переход** =>2
6. **выход**: искомое решение: **maxS**, его приспособленность **max**.

Задание:

1. Нарисовать по алгоритму блок-схему
2. Задать пространство поиска. L=7. Кодировки s_i задаются как бинарные значения натуральных чисел $x_i = \overline{0, 2^L - 1}$.
3. Задать ландшафт приспособленности, приспособленность вычисляется как функция: $\mu(s_i) = 5sin(x_i) + ln(x_i)$

В программной реализации оцениваются следующие пункты:

1. соответствие алгоритму
2. вывод на консоль:
   1) задан ландшафт поиска (достаточно вывести 32 кодировки + их приспособленности в виде списка)
   2) Приспособленности: число с 2мя знаками после запятой
   3) показан выбираемый метод

для каждого метода:

1) выводится номер шага
2) выводится текущий максимум приспособленности
3) выводится текущая лучшая кодировка
4) выводится выбираемая на шаге кодировка
5) выводится приспособленность выбираемой на шаге кодировки
6) выводится полученная на шаге окрестность (все кодировки и их приспособленности) - кроме М-К
7) выводится показана смена максимума

для всех методов

8) показана смена максимума после завершения каждого алгоритма
9) есть итоговый максимум

Отчет содержит следующие пункты:

1. блок-схема
2. ландшафт (32 кодировки из всех)
3. результаты запусков (минимум 3) – копии результатов с экрана
4. таблица запусков (№запуска, лучшее решение - алгоритм, получивший это решение)
5. выделено лучшее решение среди всех запусков и алгоритм, получивший его
6. сделаны выводы
7. соответствие кода и результатов исполнения
8. приложен код

Занятие 5. Метод ближайшего соседа

Метод ближайшего соседа является жадным методом, использующимся для поиска решения задачи коммивояжера (ЗК). Суть жадных методов в пошаговом построении решения, таким образом, чтобы на каждом шаге целевая функция как можно больше увеличивалась (в случае задачи на максимум). Таким образом, при использовании жадного метода, мы не имеем готового решения на промежуточных шагах алгоритма, а только некоторую часть окончательного решения. Жадные методы не гарантируют нахождения оптимального решения, однако, для некоторых алгоритмов можно получить оценки качества решений.

В методе ближайшего соседа принцип жадности реализуется следующим образом: на каждом шаге следующий город выбирается из всех невключенных в обход так, чтобы он был ближайшим к последнему посещенному.

Метод относится к детерминированным не точным.

В данном случае элемент случайности можно внести при помощи выбора первого города.

Примем следующие обозначения N – число городов в задаче. $X = \{x_1, x_2 \dots x_N\}$ – множество городов (обычно, используются просто порядковые номера городов). $S = (s_1, s_2 \dots s_K)$ – получаемый обход городов (решение задачи), где s_i- номер i-го в обходе города, на начальных этапах решения $K<N$, в конце $K=N$.

Алгоритм:

1. i=1, $x_i \in X$ – **случайный выбор, изменяем** $X = X\backslash\{x_i\}$, $s_i = x_i$, i=i+1
2. **если** $|X| > 0$

   то $x_i = \{x_j \in X: \rho(s_{i-1}, x_j) = \min_{\forall x_k \in X} \rho(s_{i-1}, x_k)\}$, $X = X\backslash\{x_i\}$, $s_i = x_i$, i=i+1

   переход =>2

   иначе переход =>выход

выход: искомое решение: $S = (s_1, s_2 \dots s_N)$, $Q = \sum_{i=1}^{n-1} \rho(s_i, s_{i+1}) + \rho(s_N, s_1)$.

Задание:

1. Нарисовать по алгоритму блок-схему.
2. Запрограммировать метод ближайшего соседа. На вход подается матрица расстояний между городами (пожалуйста, не вводите матрицу вручную – это долго!!). На выходе получаем последовательность обхода городов. На каждом шаге

отображается выбираемый город и расстояние от последнего города до него. Вывести решение.

3. Задать матрицу расстояний размера 5X5 с очевидным решением. Протестировать на своей матрице. Сделать 2-3 запуска. Оценить правильность решения.
4. Сделать 5-10 запусков программы с матрицей из приложения согласно своему варианту. Сделать отчет.

В программной реализации оцениваются следующие пункты:

1. соответствие алгоритму
2. вывод на консоль:

Для матрицы расстояний (5X5)

1) выводится номер шага
2) выводится текущий обход
3) выводится выбираемый на каждом шаге город
4) выводится расстояние от предыдущего до выбранного
5) выводится полученная длина текущего обхода
6) выводится итоговый цикл
7) выводится итоговая длина цикла

Отчет содержит следующие пункты:

1. блок-схема

   матрица расстояний (5X5)

   1) выведена матрица
   2) результаты запусков (копии экрана) -минимум 3 запуска
   3) выводы о корректности получаемого решения
   4) результаты запусков (итоговый цикл) в виде таблицы
   5) выделено лучшее решение среди всех запусков

   матрица расстояний (15X15)

   1) выведена матрица
   2) результаты запусков (таблица) -мин 5 запусков
   3) выделено лучшее решение среди всех запусков
2. сделаны выводы
3. соответствие кода и результатов исполнения
4. приложен код

Занятие 6. Метод ближайшего города

Метод ближайшего города также является жадным методом, применяемым к ЗК. В отличие от метода ближайшего соседа, вставка происходит не в хвост обхода, а в любое место (кроме начала).

Выбирается первый город, далее для каждого города находим кандидата на добавление: просматриваем все необойденные города и выбираем из них ближайший. Таким образом у нас для каждого города в обходе имеется кандидат на добавление и расстояние до кандидата. Затем просматриваем все пары и находим ту, в которой расстояние между городами наименьшее среди всех – кандидат из этой пары будет следующим добавлен в обход. Кандидат вставляется в обход после того города, с которым он образовал пару.

Метод относится к детерминированным не точным.

Так же как и в предыдущем случае элемент случайности можно внести при помощи выбора первого города.

К обозначениям, принятым в предыдущем алгоритме, добавим вектор $Z = (z_1, z_2 \dots z_K)$ – города, являющимися кандидатами на вставку. На каждом шаге $|Z| = |S|$.

Алгоритм:

1. i=0, $x_i \in X$ – **случайный выбор, изменяем** X $= X \backslash \{\boldsymbol{x_i}\}$, $s_i = x_i$**,** i=i+1
2. **если** $|X| > 0$

то $z_l = \left\{ x_j \in X : \rho_l(s_l, x_j) = \min\limits_{\forall x_k \in X} \rho(s_l, x_k), l = \overline{1.i} \right\}$ $l = \overline{1.i}$,

z $= \left\{ z_k \in X : \rho_k = \min\limits_{l=\overline{1.i}} \rho_l \right\}$, X $= X \backslash \{z\}$, $s_{t+1} = s_t$, $\forall t = \overline{k+1, i}$, $s_{k+1} =$ z,

i=i+1 **переход** =>2

иначе переход =>выход

выход: искомое решение: $S = (s_1, s_2 \dots s_N)$, $Q = \sum_{i=1}^{n-1} \rho(s_i, s_{i+1}) + \rho(s_N, s_1)$

Задание:

1. Нарисовать по алгоритму блок-схему.
5. Запрограммировать метод ближайшего города. На вход подается матрица расстояний между городами (пожалуйста, не вводите матрицу вручную – это долго!!). На выходе получаем последовательность обхода городов. На каждом шаге отображаются все города из построенного обхода с кандидатами на вставку и расстояниями до кандидата. Вывести решение.
2. Задать матрицу расстояний размера 5X5 с очевидным решением. Протестировать на своей матрице. Сделать 2-3 запуска. Оценить правильность решения.
3. Сделать 5-10 запусков программы с матрицей из приложения согласно своему варианту. Сделать отчет.

В программной реализации оцениваются следующие пункты:

1. соответствие алгоритму
2. вывод на консоль

матрица расстояний (5X5)

1) номер шага
2) все пары кандидатов на добавление с расстоянием в виде: город из обхода – город из не обойдённых, расстояние между ними
3) выбираемый на текущем шаге город
4) выводится текущий обход
5) выводится текущая длина обхода
6) выводится итоговый цикл
7) выводится итоговая длина цикла

Отчет содержит следующие пункты:

1. блок-схема

матрица расстояний (5X5)

1) выведена матрица
2) результаты запусков (копии экрана) -минимум 3 запуска
3) выводы о корректности получаемого решения
4) результаты запусков (итоговый цикл) в виде таблицы
5) выделено лучшее решение среди всех запусков

матрица расстояний (15X15)

1) выведена матрица

2) результаты запусков (таблица) -мин 5 запусков
3) выделено лучшее решение среди всех запусков
2. сделаны выводы
3. соответствие кода и результатов исполнения
4. приложен код

Занятие 7. Жадный алгоритм для ЗоР

Здесь предлагается один из жадных алгоритмов для поиска решения целочисленной задачи о ранце (ЗоР). Как уже было сказано выше, жадный алгоритм на каждом шаге строит локально оптимальное решение. Предлагаемый жадный алгоритм для задачи о ранце предлагает для максимизации целевой функции на каждом шаге класть в ранец предметы с максимальной стоимостью (ценностью). Ранец заполняется до тех пор, пока не нарушено весовое ограничение W_{max}.

Примем следующие обозначения N – число предметов в задаче. $S = (s_1, s_2 \dots s_K)$ – получаемый набор предметов (решение задачи), где $s_i \in \{0,1\}, i = \overline{1,N}$, если i-й предмет положен в ранец, то $s_i = 1$. Ценность i-го предмета и его вес обозначим c_i и w_i соответственно. Суммарный вес всех предметов, положенных в ранец, не должен превосходить весового ограничения W_{max}. Q – суммарная ценность всех предметов в ранце.

Алгоритм:

1. sumW=0, Q=0, $S = (s_1, s_2 \dots s_N)$, $s_i = 0,, i = \overline{1,N}$
2. $c_l = \max\limits_{\forall\, s_i=0} c_i \quad i = \overline{1.N}$
3. **если** sumW + $w_l \le W_{max}$,
   то Q = Q + c_l , sumW = sumW + w_l, $s_l = 1$ **переход** =>2
   иначе переход =>выход

выход: искомое решение: $S = (s_1, s_2 \dots s_N)$.

Задание:

1. Нарисовать по алгоритму блок-схему.
2. Запрограммировать жадный алгоритм. На вход подаются цена и вес предметов, а также дается весовое ограничение W_{max}. На выходе получаем строку, в которой представлены компоненты, соответствующие предметам, положенным в ранец, имеют значение 1, остальные - 0. На каждом шаге отображается выбираемый предмет (его порядковый номер), его цена, значение критерия и вес ранца. Вывести решение.
3. Задать ранец размера 5 и весовое ограничение с очевидным решением. Протестировать. Сделать 2-3 запуска. Оценить правильность решения.
4. Сделать 5-10 запусков программы с задачей из приложения согласно своему варианту. Сделать отчет.

В программной реализации оцениваются следующие пункты:

1. соответствие алгоритму

вывод на консоль:

1) выведены данные задачи (номер предмета, цена, вес, общий вес ранца) - задача 5 предметов
2) задача проверяется на корректность

3) выводится номер шага
4) выводится выбираемый предмет
5) выводится вес выбранного предмета
6) выводится цена выбранного предмета
7) выводится текущая цена ранца
8) выводится текущий вес ранца
9) выводится итоговое решение (предметы, цена, вес)

Отчет содержит следующие пункты:

1. блок-схема

тестовая задача 5 предметов

1) выведены данные задачи (номер предмета, цена, вес, общий вес ранца)
2) приведено пошаговое решение
3) приведено итоговое решение

тестовая задача 15 предметов

1) выведены данные задачи (номер предмета, цена, вес, общий вес ранца)
2) приведено пошаговое решение
3) приведено решение

2. сделаны выводы
3. соответствие кода и результатов исполнения
4. приложен код

Занятие 8. Жадный алгоритм (Данцига) для ЗоР

В том случае, если бы в решении ЗоР можно было брать не целые, а дробные части от предметов, то ее точное решение можно было бы найти при помощи теоремы Данцига, в этом случае решение находится за полиномиальное время. Целочисленная же задача является NP-трудной, однако принцип, положенный в основание этой теоремы можно использовать для жадного поиска решения.

Введем понятие удельной ценности предмета $\gamma_i = c_i / w_i$. На каждом шаге алгоритм добавляет в ранец предметы, имеющие максимальную удельную ценность. Ранец заполняется до тех пор, пока не нарушено весовое ограничение W_{max}.

Примем приведенные выше обозначения N – число предметов в задаче. $S = (s_1, s_2 \dots s_K)$ – получаемый набор предметов (решение задачи), где $s_i \in \{0,1\}, i = \overline{1,N}$, если i-й предмет положен в ранец, то $s_i = 1$. Ценность i-го предмета и его вес обозначим c_i и w_i соответственно, $\gamma_i = c_i / w_i$, $i = \overline{1,N}$ - удельная ценность. Суммарный вес всех предметов, положенных в ранец, не должен превосходить весового ограничения W_{max}. Q – суммарная ценность всех предметов в ранце.

Алгоритм:

1. sumW=0, Q=0, $S = (s_1, s_2 \dots s_N)$, $s_i = 0,, i = \overline{1,N}$
2. $\gamma_l = \max\limits_{\forall s_i = 0} \gamma_i \;\; i = \overline{1.N}$,
3. **если** sumW + $w_l \le W_{max}$,

   то Q = Q + c_l , sumW = sumW + w_l, $s_l = 1$ **переход** =>2**,**

   иначе переход =>выход

выход: искомое решение: $S = (s_1, s_2 \dots s_N), Q$

Задание:

1. Нарисовать по алгоритму блок-схему.

2. Запрограммировать жадный алгоритм. На вход подаются цена и вес предметов, а также дается весовое ограничение W_{max}. На выходе получаем строку, в которой представлены компоненты, соответствующие предметам, положенным в ранец, имеют значение 1, остальные - 0. На каждом шаге отображается выбираемый предмет (его порядковый номер), его цена, относительная стоимость, значение критерия и вес ранца. Вывести решение.
3. Задать ранец размера 5 и весовое ограничение с очевидным решением. Протестировать. Сделать 2-3 запуска. Оценить правильность решения.
4. Сделать 5-10 запусков программы с задачей из приложения согласно своему варианту. Сделать отчет.

<u>В программной реализации оцениваются следующие пункты:</u>

1. соответствие алгоритму
2. вывод на консоль
   1) выведены данные задачи (номер предмета, цена, вес, общий вес ранца)
      - задача 5 предметов
   2) выводится номер шага
   3) выводится выбираемый предмет
   4) выводится относительный вес выбранного предмета
   5) выводится цена выбранного предмета
   6) выводится текущая цена ранца
   7) выводится текущий вес ранца
   8) выводится итоговое решение (предметы, цена, вес)

<u>Отчет содержит следующие пункты:</u>

1. блок-схема

тестовая задача 5 предметов

   1) выведены данные задачи (номер предмета, цена, вес, общий вес ранца)
   2) приведено пошаговое решение
   3) приведено итоговое решение

тестовая задача 15 предметов

   1) выведены данные задачи (номер предмета, цена, вес, общий вес ранца)
   2) приведено пошаговое решение
   3) приведено решение

2. сделаны выводы
3. соответствие кода и результатов исполнения
4. приложен код

Занятие 9. Внедрение случайного механизма в детерминированный алгоритм

Данная лабораторная работа является подготовительной для применения жадного алгоритма при создании начальной популяции в вашем ЭГА.

Поскольку приведённые в предыдущих ЛР методы являются детерминированными, то применение их в качестве операторов формирования начальной популяции приведет к получению идентичных особей. А нам для работы необходимо генетическое разнообразие. Внесем его при помощи случайности.

Реализуем случайный механизм в виде рулетки, каждый сектор которой будет соответствовать вероятности выбора предмета или города (в зависимости от задачи).

В некоторых языках рулетка уже реализована, вы можете использовать эту функцию или запрограммировать свою.

Для ЗК формируем рулетку, которая выбирает следующий город с тем бОльшей вероятностью, чем меньшее расстояние от предыдущего мы имеем. Таким образом, для рулетки мы должны подавать значения таким образом, чтобы города, имеющие наименьшее расстояние получали бОльший сектор (например: обратные величины расстояний или *const*-расстояние; назовем их *преобразованные расстояния*).

Для ЗоР формируем рулетку, которая выбирает предмет, с тем бОльшей вероятностью, чем больше его ценность (относительная ценность).

На вход рулетки подаются значения, предназначенные для формирования вероятностей, на выходе получаем номер выпавшего сектора.

Рулетка формируется по принципу геометрической вероятности. Если $S = (s_1, s_2 \dots s_N)$ –это набор значений для формирования рулетки (ценности/относительные ценности предметов или преобразованные расстояния), то $\sum_{i=1}^{N} s_i$ – это размер рулетки и вероятность выбора j-го сектора равна $s_j \Big/ \sum_{i=1}^{N} s_i$, $j = \overline{1, N}$. Для выбора нам необходимо сгенерировать случайную величину ζ, равномерно распределенную на отрезке $[0, \sum_{i=1}^{N} s_i]$. Выбор определяется номером отрезка j, на который упала случайная величина: $\sum_{i=1}^{j-1} s_i < \zeta \leq \sum_{i=1}^{j} s_i$.

Данная функция рулетки может также использоваться в пропорциональной селекции.

Алгоритм:

1. **Вход**: $S = (s_1, s_2 \dots s_N)$,
2. Генерируем случайную величину ζ, равномерно распределенную на отрезке: $[0, \sum_{i=1}^{N} s_i]$,
3. Ищем номер отрезка j, на который упала случайная величина ζ
4. **Возвращаем** j

Задание:

1. Выберите массовую задачу, которую вы будете решать в ЛР №10
2. Запрограммируйте для нее жадные алгоритмы (для ЗК из ЛР 5-6, дляЗоР из ЛР 7-8) с внедрением случайного механизма.
3. На входе данные задачи

4. В меню осуществляется выбор одного из жадных алгоритмов (5 или 6 для ЗК и 7 или 8 для Зор)
5. Все шаги отображаются как в соответствующей ЛР.
6. На выходе – окончательное решение.
7. Для задачи размерности 5 и сделать 2-3 запуска. Оценить правильность решения.
8. Сделать 5-10 запусков программы с задачей из приложения согласно своему варианту. Сделать отчет.

В программной реализации оцениваются следующие пункты:

1. соответствие алгоритму
2. вывод на консоль (те же требования, как и в соответствующей ЛР)

Отчет содержит следующие пункты:

1. см. пункты к соответствующей ЛР

Занятие 10. Эволюционно-генетический алгоритм

Данное задание связано с решением дискретной задачи оптимизации при помощи эволюционно-генетического алгоритма.

В ходе работы необходимо реализовать сам эволюционно-генетический алгоритм, согласно приведенной блок-схеме (рис.1). При этом выбираются способы кодирования решения, конкретные операторы (кроссовер, мутация, селекция), стратегия формирования следующего поколения и условие остановки.

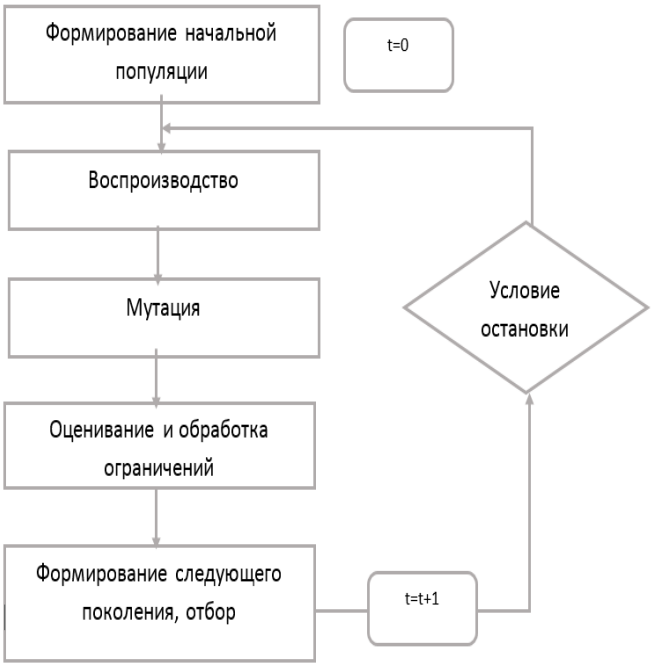

Рис.1. Блок-схема эволюционно-генетического алгоритма

При реализации алгоритма на стадии формирования начальной популяции нужно выбрать способ кодирования решения. В данной реализации кодировка зависит от решаемой задачи оптимизации.

Затем необходимо реализовать методы формирования начальной популяции.

Этап воспроизводства состоит из 2х стадий: 1я – выбор способа формирования родительской пары, 2я- работа оператора кроссовера.

Затем следует оператор мутации.

На следующем этапе происходит оценивание кодировок (если этого не произошло раньше), а также возможна коррекция кодировок и приспособленностей для обработки ограничений.

Дальше происходит формирование следующего поколения. Стратегия формирования определяет общую схему смены поколений, оператор селекции непосредственно производит отбор особей в следующую популяцию.

В программе, реализующей эволюционно-генетический алгоритм должен быть следующий набор операторов:

1. ***Оператор формирования начальной популяции – минимум 2 различных оператора.*** При этом первый может быть случайный, второй обязательно реализует какую-либо эвристику в построении допустимого решения (например, использует жадный алгоритм).
2. ***Оператор кроссовера - минимум 2 различных оператора.***
3. ***Мутация - минимум 2 различных оператора.***
4. ***Селекция - минимум 2 различных оператора.*** Оператор селекции является аналогом естественного отбора, его реализация производится при помощи вероятностного выбора особей, переносимых в следующее поколение. Каждая особь может дать более одной копии в следующем поколении. Операторы, которые просто отсеивают плохо приспособленные особи, не используя вероятностные схемы селекции, не являются реализацией естественного отбора.

В программной реализации оцениваются следующие пункты:

1. Программа реализована в консольном или GUI варианте на языке высокого уровня.
2. Выбор параметров (операторов и численности популяции) происходит при помощи меню (однократно, а не в каждом поколении)
3. Выводятся промежуточные результаты работы алгоритма:
   a. Номер поколения
   b. Лучшая особь в каждом поколении (особь, это кодировка+приспособленность), дополнительно: для ЗК выводится длина цикла, для ЗоР выводится вес кодировки
   c. Все особи в поколении (те же требования, что и к выводу лучшей особи)
4. Выводится итоговый результат работы алгоритма.

Составления отчета:

Отчет по этой работе составляется в виде мини-исследования и состоит из следующих частей:

1. Постановка массовой задачи, которую вы будете решать при помощи ЭГА
   a. Математическая
   b. Содержательная
2. Описание реализованного алгоритма
   a. Способ кодирования
   b. Методы формирования начальной популяции
   c. Способ выбора родительской пары
   d. Операторы кроссовера
   e. Операторы мутации
   f. Стратегия формирования следующего поколения
   g. Операторы селекции
   h. Условия остановки
   i. Способ обработки ограничений
3. Описание эксперимента.
   a. Основные цели эксперимента (что хотите показать) – формулируем гипотезу в отношении построенного алгоритма
   b. Параметры алгоритма (численность популяции, условия остановки, стратегия формирования поколения и т.п.)
   c. Параметры эксперимента (сколько запусков, какие операторы использовались и т.п.)
4. Результаты эксперимента (обработанные данные) в виде таблицы.
5. Выводы по результатам эксперимента
6. Приложения
   a. Сырые данные результатов эксперимента. (Описание параметров серии. Выдачи серии с зафиксированными параметрами.) – 4-5 запусков в серии.
   b. Программный код.

Важной частью отчета является **обработка экспериментальных данных.** Сырые данные результатов эксперимента приводятся только в приложении. Экспериментальными данными могут быть: итоговый результат работы алгоритма (решение), число поколений

или количество вычислений, затраченных на поиск, вероятность получения оптимального решения в серии, генетическое разнообразие в популяции и т.п.

Каким образом можно обработать серию экспериментов? Это зависит от формулируемой гипотезы, но в любом случае, необходимо статистически обработать выдачу от эксперимента. Например, в итоговой таблице можно привести лучшее, наиболее часто встречаемое решение (мода), среднее получаемое решение, разброс решений в серии и т.п.

Итоговая таблица с результатами обработки эксперимента должна быть устроена таким образом, чтобы было удобно делать выводы по работе алгоритма.

Таблица может быть **примерно** устроена следующим образом:

	Параметры алгоритма 1	….	Параметры алгоритма к
Лучшее решение			
Мода			
Разброс решений			
Среднее число поколений			
Вероятность получения оптимального решения			
…..			

Выводы формируются по итоговой таблице. Для проверки вышей гипотезы выделите из таблицы те данные, которые показывают/опровергают ее. Возможно, результаты эксперимента натолкнут вас на формулирование новой гипотезы.