МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭКОНОМИЧЕСКИЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ ИНФОРМАТИКИ И ПРИКЛАДНОЙ МАТЕМАТИКИ

КАФЕДРА ПРИКЛАДНОЙ МАТЕМАТАТИКИ И

ЭКОНОМИКО-МАТЕМАТИЧЕСКИХ МЕТОДОВ

**КУРСОВАЯ РАБОТА**

**по дисциплине**

**«Модели и методы исследования операций»**

Тема:

\_\_\_\_\_\_\_\_\_\_\_\_\_ Моделирование задачи о покрытии\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Направление: 01.03.01 - Экономика

Направленность: Математические методы и статистический анализ

Студент(ка)\_\_\_\_\_\_\_\_\_\_\_\_\_Перла\_Лев\_Аркадьевич\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*ФИО полностью*

Группа:\_\_\_\_\_\_Э-1613\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*номер группы подпись*

Проверил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Чернов В. П.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(Фамилия И. О. преподавателя)*

Должность: \_\_\_\_\_\_\_\_\_\_\_\_\_\_ профессор, д.э.н.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(уч. степень, уч. звание)*

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Дата: \_\_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Санкт-Петербург

2019 год

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc27159245)

[Глава 1. Математическая постановка задачи 4](#_Toc27159246)

[Глава 2. Жадный алгоритм 5](#_Toc27159247)

[Глава 3. Генетический алгоритм 6](#_Toc27159248)

[3.1 Описание 6](#_Toc27159249)

[3.2 Инициализация алгоритма 6](#_Toc27159250)

[3.3 Выбор родительских особей 7](#_Toc27159251)

[3.4 Кроссовер 8](#_Toc27159252)

[3.5 Мутация 9](#_Toc27159253)

[3.6 Алгоритм восстановления допустимости решения 10](#_Toc27159254)

[3.7 Замена особи в популяции на новую 12](#_Toc27159255)

[3.8 Переход к новой эпохе и вывод результата 13](#_Toc27159256)

[3.9 Тестирование и проверка на сходимость 14](#_Toc27159257)

[ЗАКЛЮЧЕНИЕ 15](#_Toc27159258)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 16](#_Toc27159259)

ВВЕДЕНИЕ

Задача о покрытии множества является классическим вопросом информатики и теории сложности. Данная задача обобщает NP-полную задачу о вершинном покрытии (и потому является NP-сложной) [1].

Данная задача имеет применение во многих сферах, она часто возникает при решении других оптимизационных задач и задач анализа данных. Ее приложения включают в себя задачи размещения, машинного обучения, распределения ресурсов, интеллектуального анализа данных, задачи, возникающие при проектировании интегральных схем и другие [5].

Приведем простейший пример взвешенной задачи о покрытии. Некоторой организации требуется нанять переводчиков с ряда языков на английский. Имеется пять кандидатур таких переводчиков A, B, C, D, E, делающих переводы только с части языков и требующих свою оплату труда.

Необходимо определить, каких переводчиков следует взять на работу, чтобы обеспечить возможность перевода со всех языков на английский и при этом гарантировать наименьшие расходы организации.

В данной работе мы рассмотрим решение задачи о покрытии с помощью жадного алгоритма, а также генетического алгоритма для решения задачи о взвешенном покрытии. Все алгоритмы написаны на языке программирования Python.

Глава 1. Математическая постановка задачи

Рассмотрим самую общую постановку задачи о покрытии:

Имеется конечное множество и его подмножества , такие что:

(1)

Требуется найти минимальный по числу подмножеств набор Sj, такой, что каждый элемент множества S принадлежит хотя бы одному из подмножеств этого набора.

Если каждому из подмножеств Sj поставлен в соответствие вес cj > 0, рассматривается задача о взвешенном покрытии.

Вводится матрица так что:

(2)

Предполагается, что каждый элемент входит в одно из подмножеств Sj. Вводятся булевы переменные xj, j = 1,2, …, n:

(3)

Тогда задача о покрытии множества состоит в нахождении покрытия с минимальной стоимостью по следующей математической модели:

(4)

При ограничениях:

(5)

(6)

Глава 2. Жадный алгоритм

Жадный алгоритм выбирает множества, руководствуясь следующим правилом: на каждом этапе выбирается множество, покрывающее максимальное число ещё не покрытых элементов.

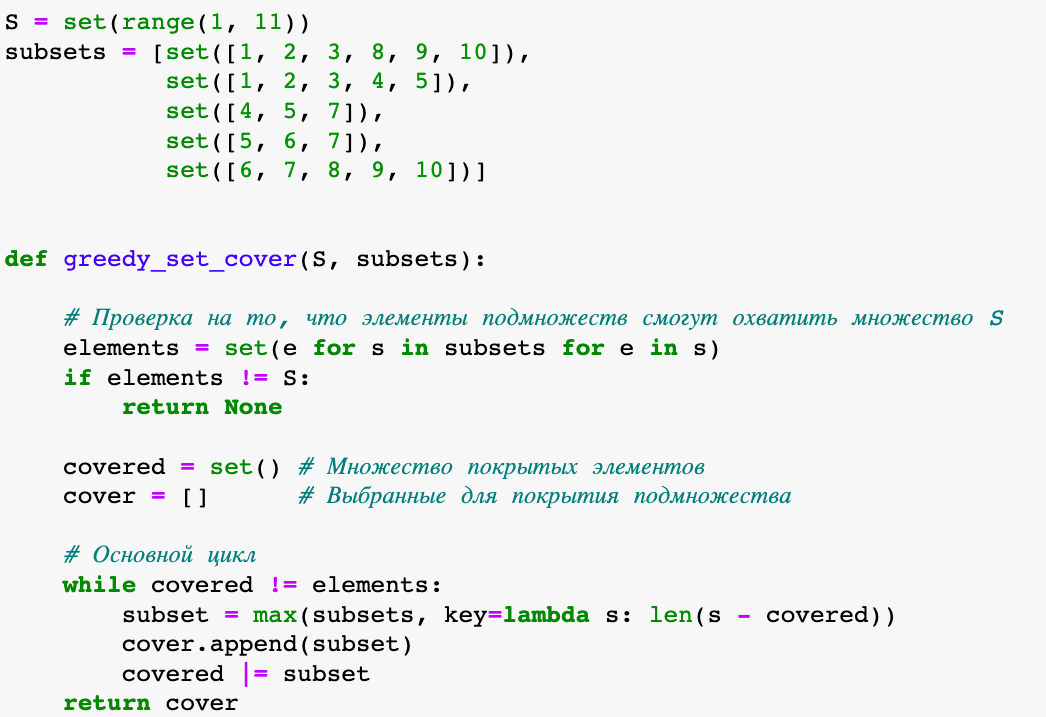


Рисунок 1 – Реализация жадного алгоритма

На рисунке 1 показана реализация жадного алгоритма для решения задачи о невзвешенном покрытии на языке Python [2].

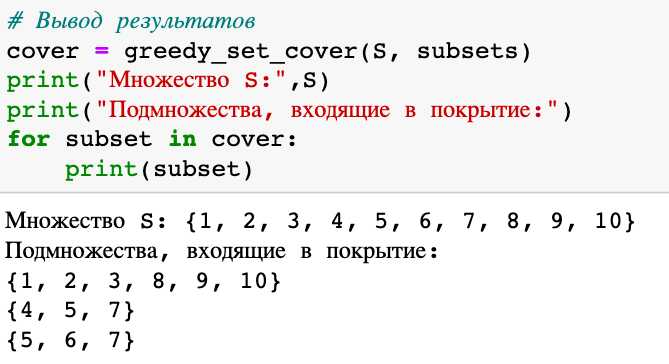


Рисунок 2 – Вывод ответа для заданных данных

Глава 3. Генетический алгоритм

3.1 Описание

Генетический алгоритм представляет собой эвристический метод случайного поиска, основанный на принципе имитации эволюции биологической популяции.

В общем случае в процессе работы алгоритма происходит последовательная смена популяций, каждая из которых является семейством покрытий, называемых особями популяции.

Покрытия начальной популяции строятся случайным образом. Наиболее распространённая и лучше всего зарекомендовавшая себя — стационарная схема генетического алгоритма, в которой очередная популяция отличается от предыдущей лишь одной или двумя новыми особями.

При построении новой особи из текущей популяции с учётом весов покрытий выбирается родительская пара особей и на их основе в процедуре кроссинговера формируется некоторый набор покрывающих множеств. Далее подвергается мутации, после чего из него строится особь, которая замещает в новой популяции покрытие с наибольшим весом. Обновление популяции выполняется некоторое (заданное) число раз, и результатом работы алгоритма является лучшее из найденных покрытий.

Далее приведем реализацию генетического алгоритма для решения задачи взвешенного покрытия множества, основанную на статье Нгуена Минь Ханга [3].

3.2 Инициализация алгоритма

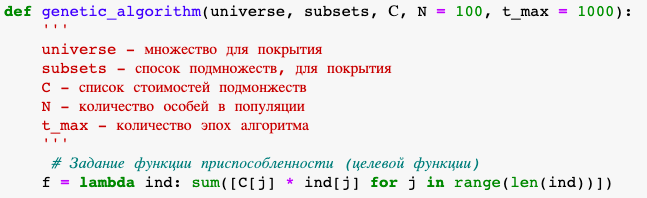


Рисунок 3 – Задание аргументов алгоритма и функции приспособленности

На рисунке 3 показано, как задаются аргументы нашего алгоритма, а также функция приспособленности, которая определяется по формуле 4.

Далее генерируется случайная начальная популяция (рисунок 4).

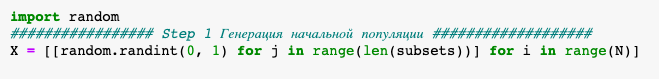


Рисунок 4 – Генерация начальной популяции

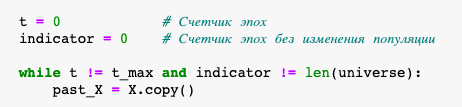


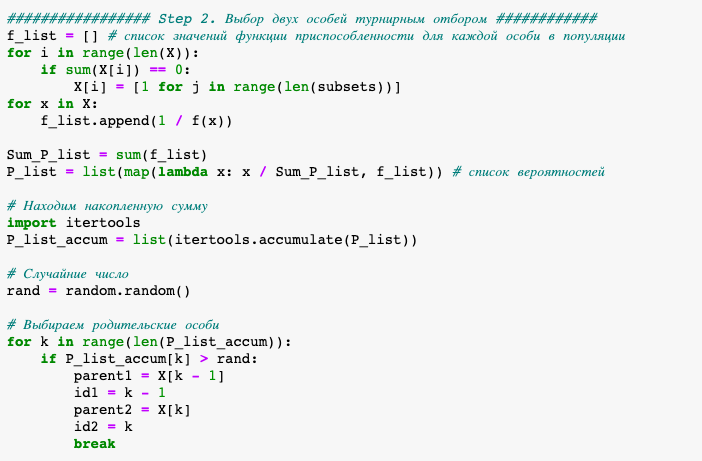
Рисунок 5 – Основной цикл

Основной цикл алгоритма будет продолжаться либо пока количество эпох не превысит установленного значения (t\_max), либо пока популяция не будет обновляться на количестве эпох, которое мы взяли как длина множества для покрытия.

3.3 Выбор родительских особей

Выбор родительских пар для скрещивания дает возможность индивидам возрождаться. Был предложен ряд методов выбора родительских пар. В данной работе применяется метод пропорционального выбора. Согласно этому методу, каждому индивиду будет присвоена вероятность выбора в соответствии с его степенью приспособленности и выбор будет производиться на основе этих вероятностей (рисунок 6). Вероятность выбора для каждого индивида рассчитывается по следующей формуле:

(7)

 Рисунок 6 – Выбор родительских особей

3.4 Кроссовер

Классические операторы кроссовера не зависят от степени приспособленности родительских особей. Поэтому они не могут моделировать доминирующие и рецессивные характеры соответствующих пар генов родителей. Для устранения этого ограничения мы предлагаем новый оператор кроссовер, создающий только 1 потомок от каждого пара родителей и позволяющий потомку унаследовать доминирующие гены от родителей.

Пусть fx1 и fx2 – степени приспособленности родительских особей x1 и x2, а new\_x – потомок, полученный в результате кроссовера. Так как гены принимают бинарные значения, обозначим как p0 частоту появления значения 0 и через p1 частоту появления значения 1 для гена в популяции.

Тогда для каждый j-ый ген (new\_xj) потомка мы определяем по следующему правилу:

1. Если j-ые гены у обоих родителей равны - new\_xj = x1j= x2j
2. Если j-ые гены у обоих родителей не равны:
   1. Если x1j = 0, присвоим new\_xj = x1j, если . Иначе присвоим new\_xj = x2j
   2. Если x1j = 1, присвоим new\_xj = x1j, если . Иначе присвоим new\_xj = x2j

Реализация кроссовера показана на рисунке 7.

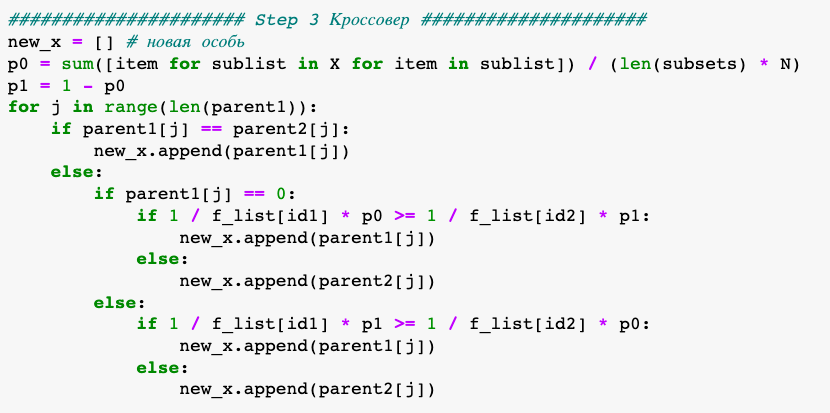


Рисунок 7 – Кроссовер

3.5 Мутация

Для проведения мутации над новой особью введем список H вероятностей изменения j-го гена. В реализации, представленной на рисунке 8, мы используем генератор случайного числа в промежутке от 0 до 2/N. По условию алгоритма вероятности мутации должны быть маленькими.

Гены, имеющие вероятность мутации больше 1/N, будут выбраны для мутации по следующему правилу замены:

1. Если Hj >1/n , new\_xj = 1, p1>p0, то new\_xj = 0.
2. Если Hj >1/n , new\_xj = 0, p0>p1, то new\_xj = 1.

Реализация алгоритма мутации представлена на рисунке 8.

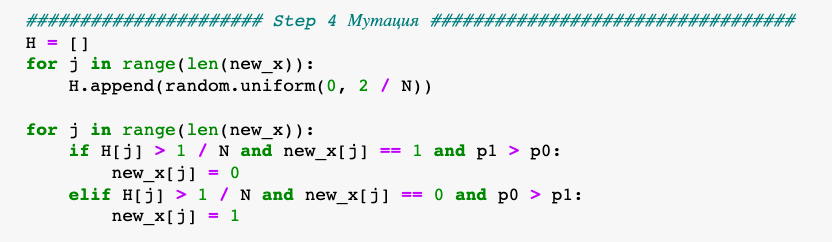


Рисунок 8 – Мутация

3.6 Алгоритм восстановления допустимости решения

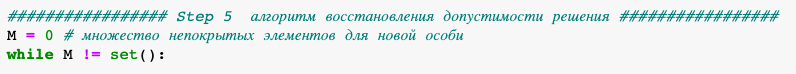


Рисунок 9 – Определение алгоритма допустимости решения

При применении операторов кроссовер и мутации над индивидами допустимость решения может быть нарушена (полученный набор не является покрытием множества). Для устранения этого явления предлагается эвристический алгоритм восстановления допустимости решения, одновременно являющийся локальным оптимизационным шагом для увеличения общей эффективности генетического алгоритма.

На рисунке 9 показано определение основного цикла алгоритма восстановления допустимости решения: он будет выполняться пока множество покрытых элементов не будет равно пустому множеству.

На первом шаге алгоритма восстановления допустимости решения, показанного на рисунке 10, происходит инициализация необходимых массивов:

* ;
* R – множество всех j, такие что Sj входит в покрытие;
* M – множество всех непокрытых элементов ;
* W – множество, в котором wi – количество подмножеств в решении, содержащих ,.

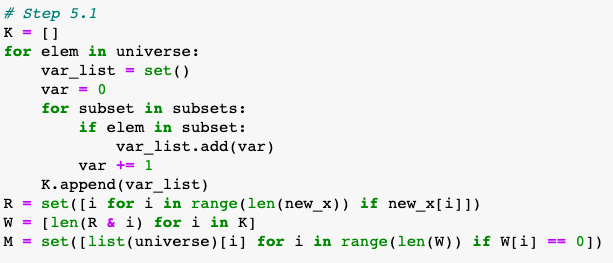


Рисунок 10 – Инициализация, шаг первый

На втором шаге (рисунок 11) для каждого по порядку нахождения в М:

1. Найти подмножество Sj по порядку возрастания в Kj, которое минимизирует:

(8)

1. Дополнить Sj в R и присвоить:
   * ;



Рисунок 11 – Шаг второй

На третьем шаге (рисунок 12) найденный вариант проверяется на избыточность.

Для каждого j по порядку убывания в R, если wi > 1 для всех, то присвоим:

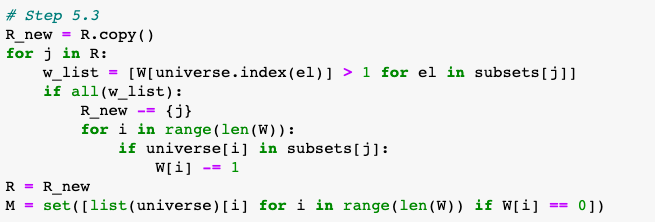


Рисунок 12 – Шаг третий

После окончания цикла алгоритма восстановления допустимости решения R является допустимым решением, не содержащим избыточных подмножеств. Восстанавливаем из R потомка (рисунок 13). Алгоритм закончен.

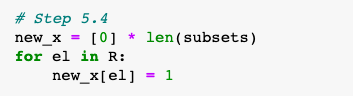


Рисунок 13 – Шаг четвертый

3.7 Замена особи в популяции на новую

Созданная особь – потомок будет использоваться для смены старой популяции по правилу стационарной замены. Особи со степень приспособленности выше среднего являются кандидатами на замену особями с меньшей степенью приспособленности.

В процессе смены популяции по правилу стационарной замены мы должны избежать повторного появления индивида в популяции.

Реализация данного процесса представлена на рисунке 14.

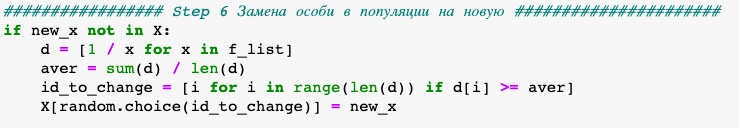


Рисунок 14 – Замена особи из популяции на новую

3.8 Переход к новой эпохе и вывод результата

На рисунке 15 изображен процесс перехода к новой эпохе. Счетчик эпох увеличивается на единицу. Если популяция за эпоху не изменилась indicator также увеличивается на единицу, в противном случае обнуляется.

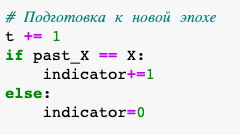


Рисунок 15 – Подготовка к новой эпохе

Результатом работы алгоритма будет особь с минимальным значением функции приспособленности в конечной популяции. Нахождение результата показано на рисунке 17.

Но в конечной популяции могут остаться особи из первоначальной случайно сгенерированной, поэтому их необходимо отфильтровать. Мы делаем это с помощью функции check, которая описана на рисунке 16.

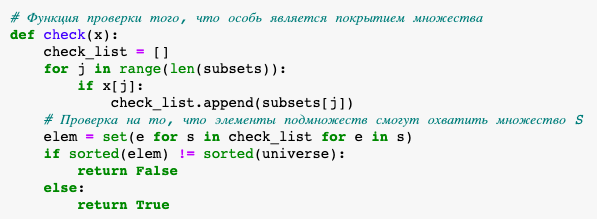


Рисунок 16 – Функция проверки покрытия

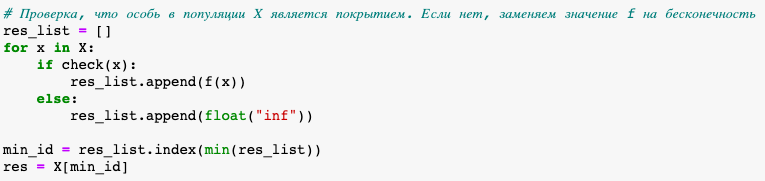


Рисунок 17 – Нахождение результата

3.9 Тестирование и проверка на сходимость

Мы сгенерировали множество из 20 элементов, его 15 подмножеств разной длины и вектор цен каждого подмножества (рисунок 18).

Также проверили, что возможно найти хотя бы одно покрытие из данных подмножеств.

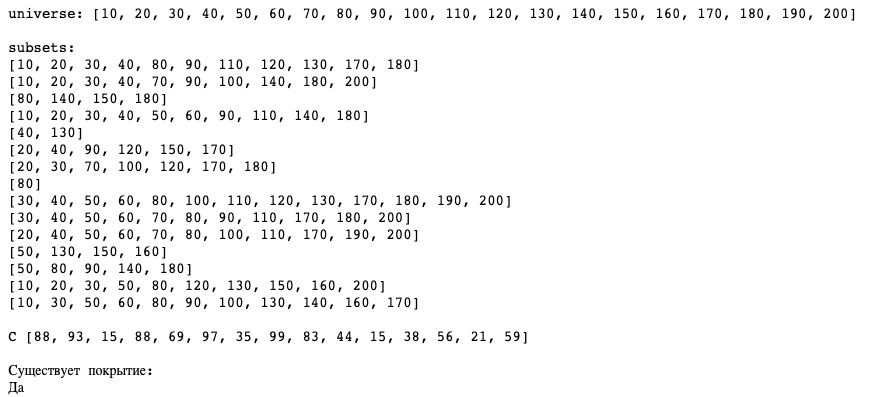


Рисунок 18 – Входные данные алгоритма

После мы запустили алгоритм и получили оптимальное решение (рисунок 19).

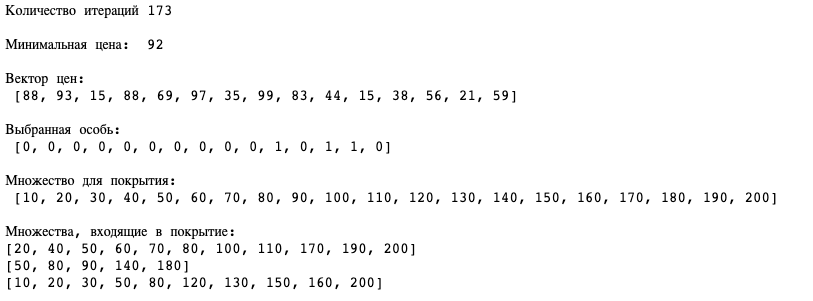


Рисунок 19 – Результат работы алгоритма

Также мы провели тест на сходимость. За 200 прогонов алгоритма по входным данным (рисунок 17) оптимальная особь находится в среднем за 130 итерации, при том каждый раз она была одна и та же.

ЗАКЛЮЧЕНИЕ

В данной курсовой работе была дана математическая постановка задачи покрытия множества. Также мы рассмотрели самые популярные эвристические алгоритмы для решения задачи покрытия множества, конкретно жадный и генетический алгоритмы.

Реализовали данные алгоритмы на языке программирования Python. Проверили результат выполнения алгоритма на сгенерированных тестах.

Данные алгоритмы можно применять в различных сферах, где необходимо решать задачи размещения объектов или назначения людей для выполнения каких-либо работ.

С другой стороны, нужно понимать, что данные алгоритмы являются эвристическими и на больших объемах данных могут сходиться к локальному минимуму, не давая оптимального решения.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Статья в интернет портале Wikipedia «Задача о покрытии множества». [Электронный ресурс]. Режим доступа:  
   ru.wikipedia.org/wiki/Задача\_о\_покрытии\_множества
2. Статья в интернет портале martinbroadhurst “Greedy set cover in Python”. [Электронный ресурс]. Режим доступа:  
   [martinbroadhurst.com/greedy-set-cover-in-python.html](http://www.martinbroadhurst.com/greedy-set-cover-in-python.html)
3. Гуен Минь Ханг "Применение генетического алгоритма для задачи нахождения покрытия множества"
4. С.А.Канцедал, М.В.Костикова «Один алгоритм решения задачи о покрытии»
5. А. В. Пролубников «Задача о покрытии множества с интервальными весами подмножеств и жадный алгоритм ее решения»