МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

им. Н.Э. Баумана

Кафедра «Систем обработки информации и управления»

ОТЧЕТ

**Лабораторная работа №4 по курсу**

Методы поддержки принятия решений

«Решение оптимизационных задач с помощью генетических алгоритмов»

|  |  |  |
| --- | --- | --- |
| ИСПОЛНИТЕЛЬ: | |  |
| студенты группы ИУ5-73 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | |
| Мелузов Олег  Рылёва Александра | "\_\_"\_\_\_\_\_\_\_\_\_2018 г. | |

ПРЕПОДАВАТЕЛЬ:

Терехов В.И.,

к.т.н., доцент

Кафедра ИУ-5. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва - 2018

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

Оглавление

[Цель работы 3](#_Toc532322662)

[Задание 3](#_Toc532322663)

[Описание предметной области и выбранной задачи в терминах генетических алгоритмов 3](#_Toc532322664)

[Формулировка задачи 3](#_Toc532322665)

[Описание программы 3](#_Toc532322666)

[Протоколы экспериментов 3](#_Toc532322667)

[Выводы 3](#_Toc532322668)

[Используемая литература 3](#_Toc532322669)

# Цель работы

1.1. Целью лабораторной работы является углубление и закрепление теоретических знаний, полученных на лекциях, приобретение практических навыков самостоятельной работы при решении оптимизационных задач больших размерностей с помощью генетических алгоритмов.

1.2. В процессе выполнения лабораторной работы по теме «Решение оптимизационных задач с помощью генетических алгоритмов» на примере задачи поиска кратчайшего пути для информационного пакета (сообщения) в компьютерной сети студенты решают следующие задачи (задания):

- описывают предметную область;

- определяют исходные данные задачи;

- формулируют задачу и исходные данные в терминах генетических алгоритмов;

- определяют последовательность работы генетического алгоритма;

- разрабатывают компьютерную программу;

- исследуют работу генетического алгоритма и полученное решение.

# Задание

2.1. Сформулировать задачу и описать исходные данные в терминах генетических алгоритмов.

2.2. Разработать программу, которая осуществляет поиск кратчайшего пути для информационного пакета (сообщения) в компьютерной сети с помощью генетического алгоритма.

2.3. При проведении серии экспериментов (не меньше 10) по исследованию работы генетического алгоритма программа должна позволять пользователю задавать топологию сети (пропускные способности каналов связи), содержащей не менее 10 компьютеров (серверов), а также указывать компьютер-отправитель и компьютер-получатель. Должны отображаться все решения (хромосомы) одного поколения до и после применения каждого оператора (скрещивания, селекции, редукции и мутации). Переход к следующему поколению должен осуществляться: в автоматическом режиме в соответствии с заданным критерием; в ручном режиме.

# Описание предметной области и выбранной задачи

В качестве решаемой задачи была выбрана задача поиска кратчайшего пути между двумя точками взвешенного неориентированного графа. Это одна из основных задач теория графов для которой уже существует множество алгоритмов решений, выполняющих в, своей основе, перебор различных комбинаций вершин. В случае, когда в графе много вершин и/или рёбер, такие алгоритмы могут выполняться достаточно долгое время. В такой ситуации, генетический алгоритм может выполнить несколько итераций и найти локальный или даже глобальный минимум задачи – кратчайший путь между двумя точками. Также, из-за общности решаемой задачи, полученный алгоритм можно использовать в различных предметных областях, элементы которых могут быть представлены в виде графов.

# Формулировка задачи в терминах генетических алгоритмов

В терминах генетических алгоритмов геном будет являться вершина графа (её номер или название), а хромосомой – набор вершин-генов, представляющий собой путь из первой вершины в последнюю. Популяция – несколько хромосом – несколько различных вариантов путей из первой вершины в последнюю.

Целевую функцию пригодности удобнее всего выразить как 1/длина\_пути\_в\_графе, т.е. чем меньше путь, закодированный в хромосоме, тем больше вероятность, что эта хромосома будет участвовать в селекции.



# Описание программы

# coding: utf-8

import networkx as nx

import numpy as np

import matplotlib.pyplot as plt

from matplotlib import colors as mcolors

import seaborn as sns

from tqdm import tqdm\_notebook

from IPython.display import clear\_output

# # Define functions

def generate\_connections(n\_nodes, con\_p=0.5, min\_length=10, max\_length=50):

connections = {}

for i in range(n\_nodes):

for j in range(i+1, n\_nodes):

if np.random.rand()<con\_p:

v = np.random.randint(min\_length, max\_length+1)

connections[(i,j)] = v

return connections

def generate\_matrix(connection\_dict, n\_nodes):

A = np.ones((n\_nodes,n\_nodes)) \* np.inf

for i in range(n\_nodes):

A[i,i]=0

for coord, val in connection\_dict.items():

A[coord]=val

A[coord[::-1]] = val

return A

def plot\_graph(G, best\_edges=None, title=None, colors=None):

pos = nx.circular\_layout(G)

nx.draw(G, pos, with\_labels=True, node\_color=colors, width=1)

labels = nx.get\_edge\_attributes(G,'weight')

labels = {k:int(v) for k,v in labels.items()}

if best\_edges:

best\_edges\_w = {e:int(A[e]) for e in best\_edges}

nx.draw(G, pos, with\_labels=True, node\_color=colors, edgelist=best\_edges, width=3, edge\_color='g')

nx.draw\_networkx\_edge\_labels(G, pos, edge\_labels=best\_edges\_w)

else:

nx.draw\_networkx\_edge\_labels(G, pos, edge\_labels=labels)

if title: plt.title(title)

plt.show()

def generate\_path(path\_len, start, stop, n\_nodes):

return np.hstack([start, np.random.randint(0, n\_nodes, path\_len-2), stop])

def get\_path\_len(path):

return sum([A[a,b] for a,b in zip(path, path[1:])])

def get\_fitness(chromo):

return 1/get\_path\_len(chromo)

def select\_pairs(paths):

return np.array([(paths[i], paths[j]) for i,j in np.random.randint(0, len(paths), size=(N//2, 2))])

def crossover(a, b):

return np.array([i if np.random.rand()<0.5 else j for i,j in zip(a,b)])

def mutation(chromos):

mutated = np.array([i if np.random.rand()<1-MUTATION\_P else np.random.randint(N\_NODES) for i in chromos[1:-1]])

return np.hstack([chromos[0], mutated, chromos[-1]])

def reduction(old\_chromos, children, fits=None, is\_manual=False):

if is\_manual:

new\_chromos = np.vstack([old\_chromos[:1], children[:len(old\_chromos)-1]])

else:

old\_chromos\_ids = list(range(len(old\_chromos)))

old\_chromos\_fits\_ar = [(i, f) for i,f in zip(old\_chromos\_ids, fits)]

old\_chromos\_ids = [i for i,f in sorted(old\_chromos\_fits\_ar, key=lambda x: -x[1])]

old\_chromos = old\_chromos[old\_chromos\_ids]

new\_chromos = np.vstack([old\_chromos[:1],children[:len(old\_chromos)-1]])

return new\_chromos

def select\_manually(chromos):

to\_print = [(i, get\_path\_len(chromo), chromo,) for i,chromo in enumerate(chromos)]

to\_print = sorted(to\_print, key = lambda line: line[1])

ids\_map = {line[0]:idx for idx, line in enumerate(to\_print)}

ids\_map\_r = {v:k for k,v in ids\_map.items()}

if CLEAR\_OUTPUT: clear\_output()

print('Select best chromosomes')

for line in to\_print:

print("{id\_}: {l}, (len={length})".format(id\_=ids\_map[line[0]], l=line[2], length=line[1]))

print('Enter best chromosomes ids:')

ids = input().strip()

if ids=='stop':

raise StopIteration

if len(ids)>0:

ids = list(map(int, ids.split(' ')))

ids = [ids\_map\_r[i] for i in ids]

not\_ids = [ids\_map\_r[i] for i in ids\_map\_r if i not in ids]

parents = chromos[ids]

chromos = np.vstack([chromos[ids], chromos[not\_ids]])

else:

parents = chromos

print('-----------------\n')

return parents, chromos

def select\_auto(chromos, fits):

probs = np.exp(fits)/np.exp(fits).sum() #Softmax function

parents = np.array([chromos[i] for i in np.random.choice(range(len(chromos)), p=probs, size=len(chromos))])

return parents

def ga\_step(mx, old\_chromos, is\_manual=False):

chromos = old\_chromos.copy()

#if len(chromos)%2==1: raise Exception('chromos number should be odd')

# step2: calculate fitness

fits = np.array([get\_fitness(chromo) for chromo in chromos])

fits = fits/(fits.mean()+1e-10)

# step3: select parents vectors

if is\_manual:

parents, chromos = select\_manually(chromos)

else:

parents = select\_auto(chromos, fits)

# step4: set pairs. apply crossover and mutation

pairs = select\_pairs(parents)

children = np.array([crossover(pair[0],pair[1]) for pair in np.vstack([pairs,pairs])]) #Each pair gives 2 children

mutated = np.array([mutation(child) for child in children])

new\_chromos = reduction(chromos, mutated, fits, is\_manual)

stats = np.array([old\_chromos, parents, children, mutated, new\_chromos])

return new\_chromos, stats

# # Create graph

N\_NODES = 10

connection\_dict = generate\_connections(N\_NODES, con\_p=0.5)

A = generate\_matrix(connection\_dict, N\_NODES) # Матрица графа

G = nx.Graph(A) # Граф для рисования

edges\_to\_remove = [ (i,j) for i in range(N\_NODES) for j in range(N\_NODES) if np.isinf(A[i,j])]

G.remove\_edges\_from(edges\_to\_remove)

A\_to\_print = A.copy()

A\_to\_print[A\_to\_print==0] = None

A\_to\_print[A\_to\_print==np.inf] = None

sns.heatmap(A\_to\_print, cmap='winter\_r', annot=True, cbar\_kws={'label':'Вес'})

plt.title('Веса в графе')

plt.xlabel('Вершины')

plt.ylabel('Вершины')

plt.show()

start\_node, end\_node = 0, 1

colors = list(mcolors.get\_named\_colors\_mapping().keys())[:N\_NODES]

colors = ['xkcd:green' if i==start\_node else 'xkcd:red' if i==end\_node else colors[i] for i in range(len(colors))]

plot\_graph(G, title='Сгенерированный граф', colors=colors)

# # Manual optimization

N=10

chromo\_len = 10

is\_manual = True

CLEAR\_OUTPUT = False

MUTATION\_P = 0.3

chromos = np.array([generate\_path(chromo\_len, start\_node, end\_node, N\_NODES) for i in range(N)])

all\_chromos = np.array([chromos])

min\_path\_lens = np.array([min([get\_path\_len(i) for i in chromos])])

stats = []

for i in tqdm\_notebook(range(1000)):

try:

chromos, stat = ga\_step(A, chromos, is\_manual=is\_manual)

except StopIteration:

print("Stopped at",i)

break

stats.append(stat)

min\_path\_lens = np.append(min\_path\_lens, min([get\_path\_len(i) for i in chromos]))

all\_chromos = np.append(all\_chromos, [chromos], axis=0)

min\_len = min\_path\_lens[-1]

if not is\_manual and i>=early\_stopping\_steps and min\_len!=np.inf and all(min\_len==min\_path\_lens[- early\_stopping\_steps:]):

print('early stops at',i)

print('result', min\_len)

break

stats = np.array(stats)

plt.figure(figsize=(15,10))

plt.subplot(231)

sns.heatmap(stats[0][0], cmap=colors, cbar=False, annot=True)

plt.ylabel('Хромосомы')

plt.xlabel('Гены')

plt.title('Исходные хромосомы')

plt.subplot(232)

sns.heatmap(stats[0][1], cmap=colors, cbar=False, annot=True)

plt.ylabel('Хромосомы')

plt.xlabel('Гены')

plt.title('Селекционированные хромосомы')

plt.subplot(233)

sns.heatmap(stats[0][2], cmap=colors, cbar=False, annot=True)

plt.ylabel('Хромосомы')

plt.xlabel('Гены')

plt.title('Скрещенные хромосомы')

plt.subplot(234)

sns.heatmap(stats[0][3], cmap=colors, cbar=False, annot=True)

plt.ylabel('Хромосомы')

plt.xlabel('Гены')

plt.title('Мутировавшие хромосомы')

plt.subplot(235)

sns.heatmap(stats[0][4], cmap=colors, cbar=False, annot=True)

plt.ylabel('Хромосомы')

plt.xlabel('Гены')

plt.title('Редуцированные хромосомы')

plt.suptitle('Развитие хромосом на первой итерации')

plt.show()

plot\_data = np.array(min\_path\_lens)

#plot\_data[np.isinf(plot\_data)]=plot\_data[~np.isinf(plot\_data)].max()\*1.2 #Replace inf with finite numbers

plot\_data[np.isinf(plot\_data)]=3#plot\_data[~np.isinf(plot\_data)].max()\*1.2 #Replace inf with finite numbers

plt.plot(plot\_data)

plt.title('Минимальный найденный путь')

plt.ylabel('Минимальное расстояние')

plt.xlabel('Итерация')

#plt.yscale('log')

plt.show()

best\_chromo = chromos[np.argmin([get\_path\_len(i) for i in chromos])]

best\_chromo = best\_chromo[np.append(best\_chromo[:-1]!=best\_chromo[1:], True)]

best\_edges = [(best\_chromo[i], best\_chromo[i+1]) for i in range(len(best\_chromo[:-1]))]

plot\_graph(G, best\_edges=best\_edges, title='Найденный путь', colors=colors)

# # Automatic optimization

N=1000

path\_len = 10

early\_stopping\_steps = 100

is\_manual = False

MUTATION\_P = 0.3

chromos = np.array([generate\_path(chromo\_len, start\_node, end\_node, N\_NODES) for i in range(N)])

all\_chromos = np.array([chromos])

min\_path\_lens = np.array([min([get\_path\_len(i) for i in chromos])])

stats = []

for i in tqdm\_notebook(range(1000)):

try:

chromos, stat = ga\_step(A, chromos, is\_manual=is\_manual)

except StopIteration:

print("Stopped at",i)

break

stats.append(stat)

min\_path\_lens = np.append(min\_path\_lens, min([get\_path\_len(i) for i in chromos]))

all\_chromos = np.append(all\_chromos, [chromos], axis=0)

min\_len = min\_path\_lens[-1]

if not is\_manual and i>=early\_stopping\_steps and min\_len!=np.inf and all(min\_len==min\_path\_lens[- early\_stopping\_steps:]):

print('early stops at',i)

print('result', min\_len)

break

stats = np.array(stats)

plot\_data = np.array(min\_path\_lens)

plot\_data[np.isinf(plot\_data)]=plot\_data[~np.isinf(plot\_data)].max()\*1.2 #Replace inf with finite numbers

plt.plot(plot\_data)

plt.title('Минимальный найденный путь')

plt.ylabel('Минимальное расстояние')

plt.xlabel('Итерация')

#plt.yscale('log')

plt.show()

best\_chromo = chromos[np.argmin([get\_path\_len(i) for i in chromos])]

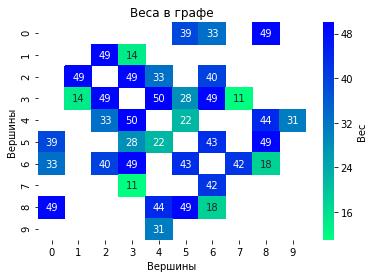
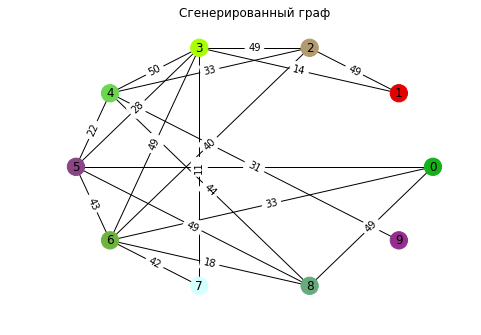
best\_chromo = best\_chromo[np.append(best\_chromo[:-1]!=best\_chromo[1:], True)]

best\_edges = [(best\_chromo[i], best\_chromo[i+1]) for i in range(len(best\_chromo[:-1]))]

plot\_graph(G, best\_edges=best\_edges, title='Найденный путь', colors=colors)

# Протоколы экспериментов

Первый сгенерированный граф:



## Эксперимент 1) Ручное управление.

Величина популяции N=10, так как человеку сложно анализировать больший объём информации.

Протокол выбора хромосом:

Select best chromosomes

0: [0 5 3 6 6 2 6 3 1 1], (len=259.0)

1: [0 6 9 7 0 7 3 7 2 1], (len=inf)

2: [0 9 6 4 2 7 7 0 0 1], (len=inf)

3: [0 4 5 8 1 7 9 9 7 1], (len=inf)

4: [0 9 3 7 3 7 4 1 5 1], (len=inf)

5: [0 5 2 5 2 9 4 4 9 1], (len=inf)

6: [0 7 4 0 7 8 6 8 1 1], (len=inf)

7: [0 9 1 0 7 6 7 8 7 1], (len=inf)

8: [0 0 4 4 3 3 1 4 2 1], (len=inf)

9: [0 0 0 6 3 0 0 1 0 1], (len=inf)

Enter best chromosomes ids:

0 5

-----------------

Select best chromosomes

0: [0 5 3 6 6 2 6 3 1 1], (len=259.0)

1: [0 5 4 2 2 2 9 4 9 1], (len=inf)

2: [0 7 2 5 2 9 4 4 9 1], (len=inf)

3: [0 5 2 6 2 9 4 4 9 1], (len=inf)

4: [0 5 2 0 2 1 3 4 9 1], (len=inf)

5: [0 5 2 6 2 8 6 4 2 1], (len=inf)

6: [0 5 2 6 2 1 6 8 5 1], (len=inf)

7: [0 4 2 5 2 9 4 4 9 1], (len=inf)

8: [0 5 5 5 7 2 6 2 1 1], (len=inf)

9: [0 5 2 5 2 9 4 4 9 1], (len=inf)

Enter best chromosomes ids:

0 8

-----------------

Select best chromosomes

0: [0 8 6 6 6 7 6 3 1 1], (len=214.0)

1: [0 5 3 6 6 2 2 3 1 1], (len=219.0)

2: [0 5 3 6 6 2 1 3 1 1], (len=233.0)

3: [0 5 3 6 6 2 6 3 1 1], (len=259.0)

4: [0 5 3 5 7 2 6 3 1 1], (len=inf)

5: [0 5 3 5 9 2 6 3 1 1], (len=inf)

6: [0 5 3 6 5 2 1 3 2 1], (len=inf)

7: [0 5 5 7 1 3 6 2 1 1], (len=inf)

8: [0 5 3 8 4 2 6 4 1 1], (len=inf)

9: [0 5 3 5 3 2 6 2 8 1], (len=inf)

Enter best chromosomes ids:

1 2 6

-----------------

Select best chromosomes

0: [0 5 3 6 6 2 2 3 1 1], (len=219.0)

1: [0 5 3 3 6 7 3 3 2 1], (len=267.0)

2: [0 5 3 6 5 3 2 3 2 1], (len=383.0)

3: [0 5 3 6 8 2 2 3 2 1], (len=inf)

4: [0 5 3 6 6 2 2 5 2 1], (len=inf)

5: [0 5 4 6 5 2 2 3 1 1], (len=inf)

6: [0 5 3 6 5 2 1 3 0 1], (len=inf)

7: [0 3 4 2 5 2 2 3 2 1], (len=inf)

8: [0 5 3 6 5 2 4 3 6 1], (len=inf)

9: [0 5 3 0 6 6 1 3 4 1], (len=inf)

Enter best chromosomes ids:

0 1 6

-----------------

Select best chromosomes

0: [0 5 3 6 6 2 2 3 1 1], (len=219.0)

1: [0 5 5 3 6 3 5 3 2 1], (len=319.0)

2: [0 5 3 7 6 0 6 3 2 1], (len=333.0)

3: [0 5 3 6 5 2 1 1 7 1], (len=inf)

4: [0 2 3 1 6 7 1 6 0 1], (len=inf)

5: [0 5 4 3 5 2 2 3 0 1], (len=inf)

6: [0 5 3 6 8 2 2 3 1 1], (len=inf)

7: [0 5 9 6 5 2 1 3 0 1], (len=inf)

8: [0 5 7 6 5 2 1 3 1 1], (len=inf)

9: [0 5 3 6 5 7 5 2 0 1], (len=inf)

Enter best chromosomes ids:

0 4 3

-----------------

Select best chromosomes

0: [0 5 3 6 6 2 2 3 1 1], (len=219.0)

1: [0 8 3 6 5 2 1 1 1 1], (len=inf)

2: [0 7 3 1 6 1 2 7 0 1], (len=inf)

3: [0 5 3 5 4 7 1 6 0 1], (len=inf)

4: [0 5 3 6 4 0 2 2 1 1], (len=inf)

5: [0 5 3 6 6 3 2 2 7 1], (len=inf)

6: [0 5 3 6 6 2 2 8 3 1], (len=inf)

7: [0 3 3 1 5 3 1 3 0 1], (len=inf)

8: [0 0 3 1 2 2 2 1 1 1], (len=inf)

9: [0 5 5 6 5 2 1 9 1 1], (len=inf)

Enter best chromosomes ids:

0 6 3

-----------------

Select best chromosomes

0: [0 5 3 6 6 2 2 3 1 1], (len=219.0)

1: [0 5 3 5 5 2 1 5 1 1], (len=inf)

2: [0 5 3 6 6 2 2 8 5 1], (len=inf)

3: [0 5 3 6 4 7 2 6 0 1], (len=inf)

4: [0 5 3 5 4 2 2 6 0 1], (len=inf)

5: [0 5 3 5 7 2 1 2 0 1], (len=inf)

6: [0 8 3 6 4 7 2 3 0 1], (len=inf)

7: [0 5 8 6 2 2 2 0 7 1], (len=inf)

8: [0 5 3 5 4 2 1 8 3 1], (len=inf)

9: [0 5 3 6 3 7 9 2 7 1], (len=inf)

Enter best chromosomes ids:

0 1 9

-----------------

Select best chromosomes

0: [0 5 3 6 6 2 2 3 1 1], (len=219.0)

1: [0 5 3 4 2 2 4 3 1 1], (len=247.0)

2: [0 1 3 0 5 5 1 2 7 1], (len=inf)

3: [0 5 3 6 6 1 2 5 7 1], (len=inf)

4: [0 7 3 6 3 7 9 2 5 1], (len=inf)

5: [0 5 1 8 6 2 2 3 5 1], (len=inf)

6: [0 5 3 6 5 7 9 2 1 1], (len=inf)

7: [0 5 3 6 6 9 7 3 1 1], (len=inf)

8: [0 5 3 4 3 7 9 2 9 1], (len=inf)

9: [0 8 3 6 1 2 2 4 1 1], (len=inf)

Enter best chromosomes ids:

0 1 2

-----------------

Select best chromosomes

0: [0 6 2 6 6 6 2 3 1 1], (len=216.0)

1: [0 5 3 6 6 2 2 3 1 1], (len=219.0)

2: [0 5 3 6 6 3 2 3 0 1], (len=inf)

3: [0 5 3 4 2 2 0 3 1 1], (len=inf)

4: [0 5 0 6 6 2 7 4 1 1], (len=inf)

5: [0 9 6 6 1 6 2 3 3 1], (len=inf)

6: [0 5 3 3 4 4 9 0 9 1], (len=inf)

7: [0 5 3 1 6 2 2 3 2 1], (len=inf)

8: [0 5 9 6 6 2 2 1 1 1], (len=inf)

9: [0 5 3 0 6 2 2 3 8 1], (len=inf)

Enter best chromosomes ids:

0 5

-----------------

Select best chromosomes

0: [0 6 2 6 6 6 2 3 1 1], (len=216.0)

1: [0 5 6 8 3 6 2 3 3 1], (len=inf)

2: [0 6 2 9 6 6 2 3 1 1], (len=inf)

3: [0 9 6 5 3 6 8 3 3 1], (len=inf)

4: [0 9 6 0 1 6 2 3 2 1], (len=inf)

5: [0 6 6 6 6 3 0 9 1 1], (len=inf)

6: [0 9 6 6 1 6 2 3 3 1], (len=inf)

7: [0 6 2 6 4 6 7 0 1 1], (len=inf)

8: [0 9 6 6 1 6 2 3 3 1], (len=inf)

9: [0 9 4 6 1 6 2 3 3 1], (len=inf)

Enter best chromosomes ids:

0 5

-----------------

Select best chromosomes

0: [0 6 2 6 6 6 2 3 1 1], (len=216.0)

1: [0 6 2 6 6 1 3 3 4 1], (len=inf)

2: [0 1 6 6 6 3 2 0 3 1], (len=inf)

3: [0 6 7 7 6 1 2 9 1 1], (len=inf)

4: [0 6 9 6 6 6 2 3 1 1], (len=inf)

5: [0 5 6 6 6 3 3 3 4 1], (len=inf)

6: [0 6 2 6 6 6 2 9 1 1], (len=inf)

7: [0 6 2 4 6 3 2 3 1 1], (len=inf)

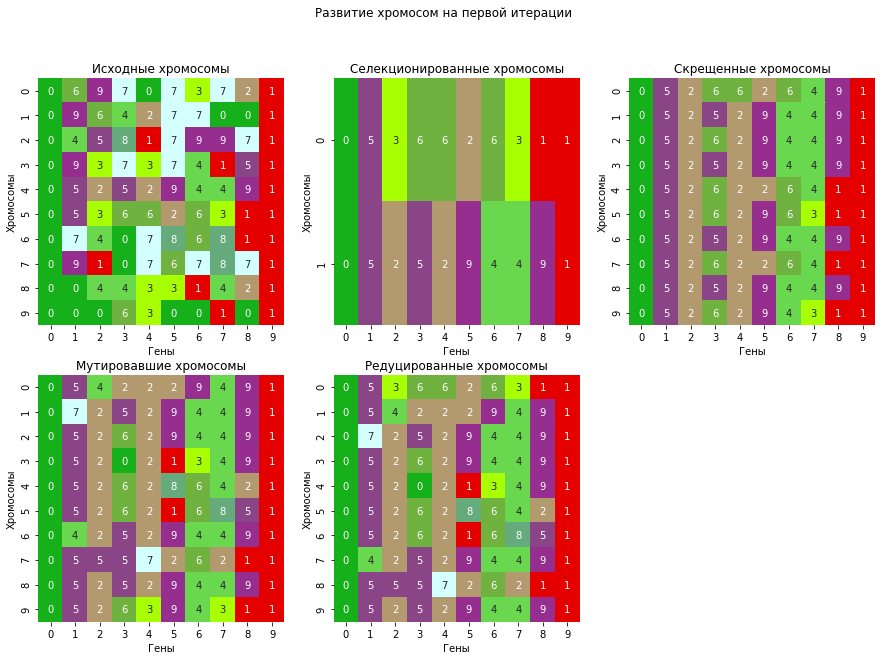
8: [0 6 6 9 6 6 8 5 3 1], (len=inf)

9: [0 6 1 6 6 6 5 3 1 1], (len=inf)

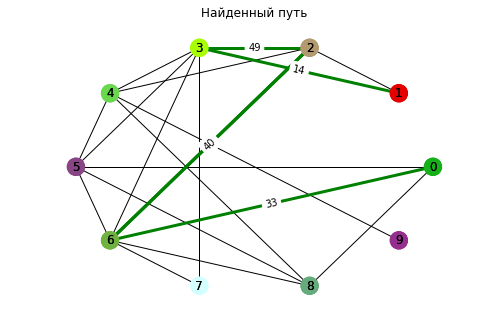
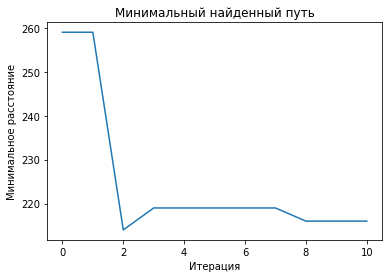
Enter best chromosomes ids:

stop

Stopped at 10

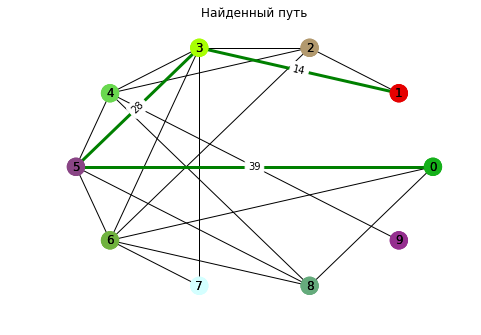
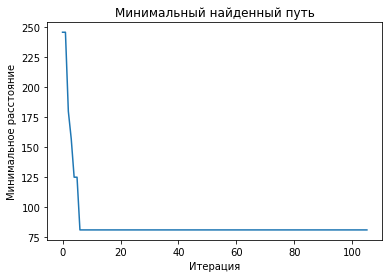
****

Минимальный найденный путь:



## Эксперимент 2) Автоматический поиск

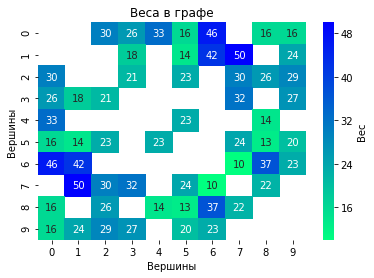
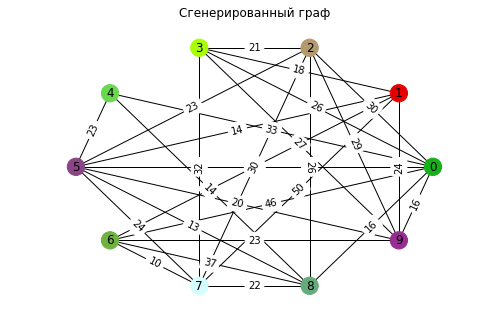
N=1000

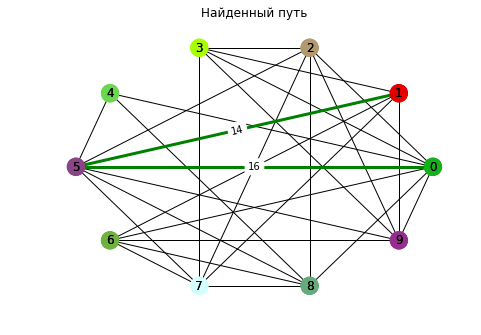
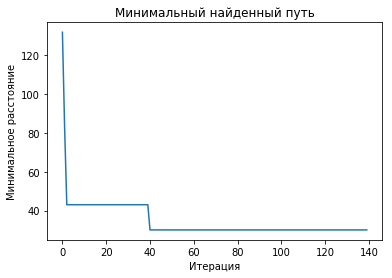


## Эксперимент 3) Автоматический поиск

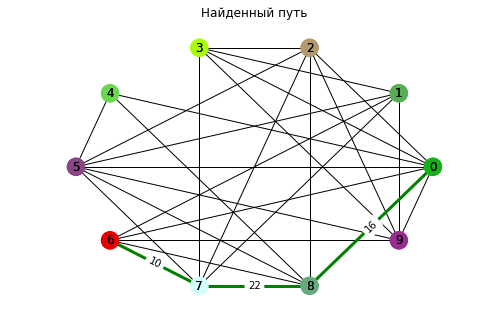
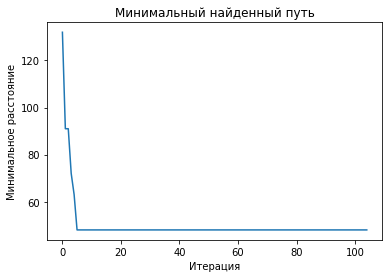
N=1000

Новый граф:

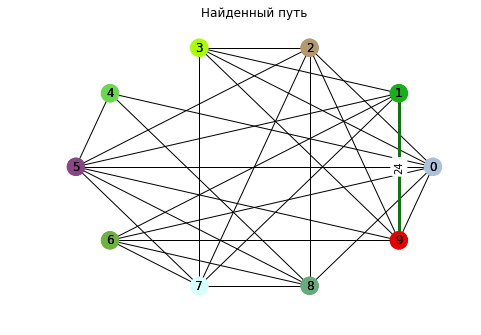


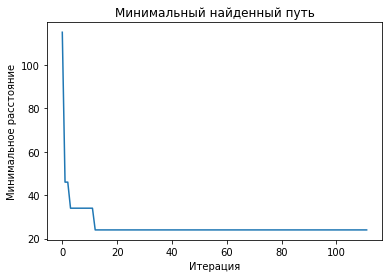


## Эксперимент 4) Автоматический поиск

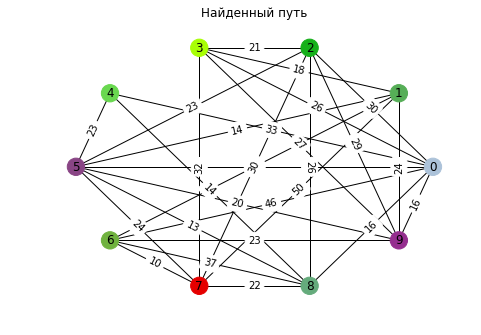
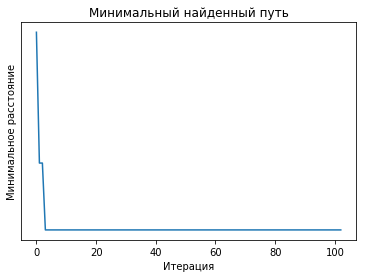


## Эксперимент 5) Автоматический поиск

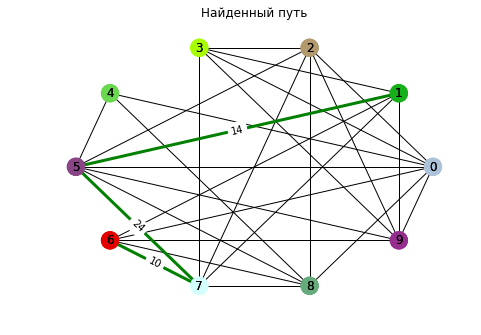
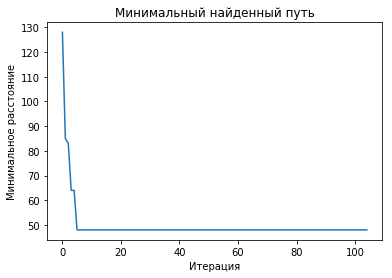




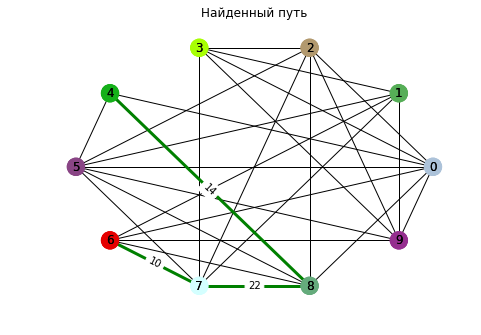
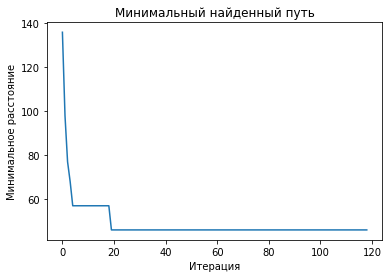
## Эксперимент 6) Автоматический поиск



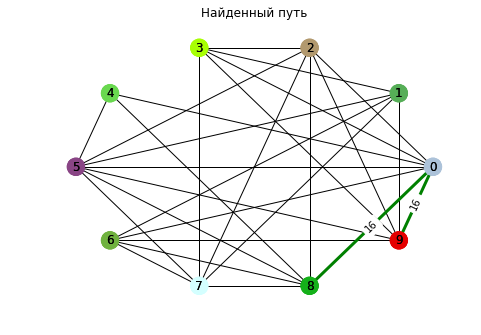
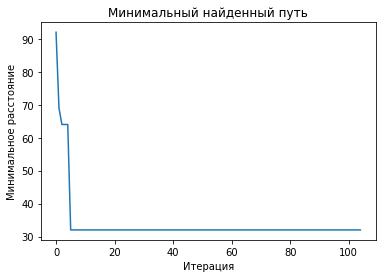
## Эксперимент 7) Автоматический поиск



## Эксперимент 8) Автоматический поиск

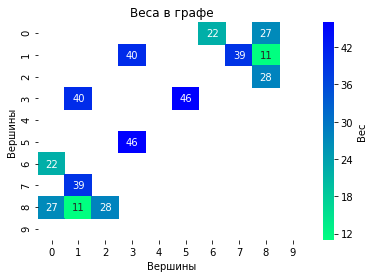
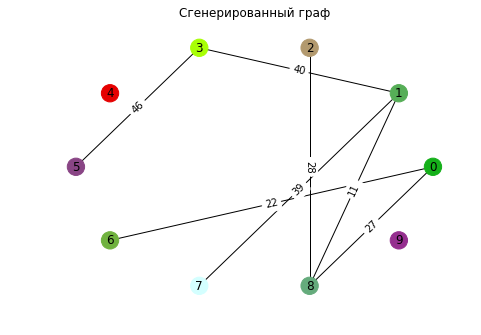


## Эксперимент 9) Автоматический поиск



## Эксперимент 10) Автоматический поиск

Граф:



## 

# Выводы

В ходе лабораторной работы мы углубили и закрепили теоретические знания, полученные на лекциях, приобрели практические навыки самостоятельной работы при решении оптимизационных задач больших размерностей с помощью генетических алгоритмов.

На данный момент методика построения генетических алгоритмов на данное время описана достаточно подробно, поэтому даже неподготовленному человеку уже не так сложно реализовать на практике подобный алгоритм. Однако достаточно сложно его интерпретировать.

Программа, автоматически выполняющая все операции, способна работать с данными очень большой размерности, из-за чего становиться сложно проанализировать её работу. С другой стороны, управление работой алгоритма в ручном режиме занимает много времени и требует малых размеров популяций, с которой человеку было бы удобно работать. При этом, результаты на каждой итерации в любом случае получаются слабо предсказуемыми, так как после выбором пользователем селекционных хромосом, происходят случайные операции скрещивания и мутации.

# Используемая литература

* Терехов В.И. Лекции по курсу Методы Поддержки Принятия Решений / МГТУ им. Н.Э. Баумана
* Д.В. Cперанский, Ю.А. Скобцов. Эволюционные вычисления / МГУПИ