**Софийски университет**

**„Св. Климент Охридски”**

## Факултет по математика и информатика

## Проект по дисциплината

## “****Извличане на знание от данни****”



Проект на:

Мария Пламенова Паскова фн. 61559

Николай Станиславов Стойков фн. 61587

Съдържание

[Факултет по математика и информатика 1](#_Toc442457908)

[Проект по дисциплината 1](#_Toc442457909)

[“Извличане на знание от данни” 1](#_Toc442457910)

[1. Мотивация и задача на курсовата работа 3](#_Toc442457911)

[1.1. Основни цели и задачи 3](#_Toc442457912)

[2. Визуализация на граф 3](#_Toc442457913)

[3. Проблем 3](#_Toc442457914)

[4. Подход и използвани алгоритми 4](#_Toc442457915)

[5. Бъдещи идеи за развитие 4](#_Toc442457916)

[6. Изводи 5](#_Toc442457917)

# Мотивация и задача на курсовата работа

### Основни цели и задачи

Графите са една незаменима част от математиката и информатиката. Изучаването им започва още от първи курс и продължава да се изучава още няколко години.

Предметите, които са пряко или косвено свързани с графите са много. На всеки един език може да се построи граф и да се решават задачи с него.

Ние решихме да направим визуализацията на граф по-добра, за да бъде от помощ на хората, които се опитват да реализират такива или чрез графи да представят някакви данни.

Целите, които си поставихме бяха:

* Визуализация на граф
* Изпълняване на алгоритми върху граф
* Подреждане на създаден граф

За да можем да подреждаме добре един граф, то трябва първо той да се изрисува от потребителя. Това беше първата част от проекта.

# Визуализация на граф

За визуализацията на графа използваме canvas, в който потребителят може да изчертае върховете и ребрата на графа. За цялата физика за подреждането на графа не е използвана никаква библиотека – всичко е направено математически.

Върховете, между които има ребра се привличат, върховете, които не са свързани, се отдалечават. По този начин се постига относително добра визуализация на графа.

Има възможност за ръчно преместване на върхове. Може да се хване един връх и да се премести, като това не променя структурата на графа.

# Проблем

Ако построи голям граф с много пресичания, то той ще бъде трудно възприеман от потребителя.

Това е причината да решим, че би било хубаво ако графът изглежда добре. Под добре изглеждащ граф ние дефинираме – минимално пресичащи се ребра, нодове и възприемчиво разстояние между съседните върхове.

# Подход и използвани алгоритми

Решението, с което подходихме е да приложим генетичен алгоритъм с някои модификации. Най-напред започнахме с оптимизиране на броя на пресичанията между ребрата на графа, но това не доведе до много добри резултати тъй като имаше проблеми с пресичането на ребра и върхове. За да решим този проблем приложихме оптимизация и върху това да се минизира броя на пресичанията между върховете и ребрата. Но освен това имаше и още нещо, с което трябваше да се справим – пресичането между самите върхове (върховете са кръгове с определен радиус).

Една от модификациите, които приложихме бе да използваме мутацията като момент за подобряване на графа, а не просто рандомизиране на позициите и и вкарване на нови гени. Подобряването на графа се крие в това че при мутация ние проверяваме кои върхове са „проблемни“ и на базата на това ние знаем кои върхове би било най-подходящо да разместим.

Интересен момент е и генерирането на деца от 2-ма произволни родители. В началото, когато генерираме популацията създаваме много наброй графи с различни позиции за техните върхове. При генериране на децата се взимат позиции на върхове от 2-та родителски графа и тези позиции биват използвани за други върхове от дъщерния граф. Така едновременно взимаме „гени“ от родителите и променяме детето, но и вкарваме разновидности.

# Бъдещи идеи за развитие

**Идея номер 1: Търсене на мост в граф и разделяне на подграфи.**

Тук идеята е да се намери мост в графа, т.е. такова ребро, което когато се премахне разделя графа на два подграфа. По този начин може тези два графа да бъдат подредени по отделно в различни части от пространството, което имаме. Може да си представим пространството разделено на правоъгълници с определена големина, в която да се побира един подграф. След като бъдат подредени отделните графи отново ще бъдат добавени липсващите ребра и така ще се получи един по-добре разпределен в пространството граф.

**Идея номер 2: Търсене в ширина**

При тази идея реализацията може да се раздели по два начина. Един вариант е да се намери най-дългия път в графа и след това да се види кой е върхът, който е по средата. След това от този връх да се пусне търсене в ширина. Този подход ще позволи графът да бъде нареден в нещо като формата на кръг, ако този връх бъде поставен по средата тъй като всички съседи ще бъдат приблизително еднакъв брой от двете му страни.

Другият подход е да се пуска BFS от различни върхове и да се дефинира „най-добре изглеждащ“ граф. Така графа би изглеждал на нива спрямо това как е обходен от BFS алгоритъма. Този подход би предоставил граф, който се доближава до начина по който един нормален човек би го подредил.

# Изводи

Резултатите които получаваме при изпълнението на имплементирания генетичен алгоритъм водят до извода че при по-големи и по-гъсти (т.е. с повече ребра) графи решението не се справя особено добре. Тъй като алгоритъма се базира на рандомизация на позициите на върховете успешността на решението зависи до голяма степен от това начално генериране на произволни позиции. В някои случаи се случва решението да отнема дълго време преди да успее да намери задоволителен резултат.

Този алгоритъм работи добре, когато графа има сравнително малко ребра като броя на върховете не е от толкова голямо значение. За по-малки графи резултата от алгоритъма е задоволителен и се получава в кратко време.

Важно е да се отбележи също така, че когато се увеличи толерантността (т.е. решение с повече пресичания на ребра да се приеме за правилно) на алгоритъма може да се получи по-бързо решение, което обаче да не изглежда толкова добре.