Анализатор графов

Это приложение позволяет выделять сообщества в графах, находить ключевые вершины и делать раскладку. Оно поддерживает чтение и запись в `.csv` и `.db` (SQLite).

Установка и запуск

Приложение работает на Java SDK 15 версии.

- Установка:

```
git clone https://github.com/spbu-team-11/graph-analyzer-app
```

- Запуск:

Windows

PowerShell

```
./gradlew run
```

Command Prompt

```
gradlew run
```

Linux

```
./gradlew run
```

MacOS

```
./gradlew run
```

Выделение сообществ

Для выделения сообществ используется Leiden alghoritm, в качестве параметров ему передаются:

1. Количество итераций алгоритма. (Iteration)
2. Параметр отвечающий за размер выделяемых сообществ, более высокое значение ведет к большему количеству сообществ, а более низкое разрешение ведет к меньшему количеству сообществ. (Resolution)

Сообщества графически выделяются цветом, также можно посмотреть числовую метку сообщества (Labels → Community)

Выделение ключевых вершин

Для выделения ключевых вершин используется Harmonic centrality с алгоритмом Дейкстры и нормализованновым показателем centrality. Вершины выделяется размером, который считается по формуле:

$$R * \frac{2((ek)^x - (e\frac{k}{2})^x)}{k}$$

- x - показатель centrality
- R - стандартный размер вершины
- k - передаваемый параметр (Size-Ratio coefficient)

> Чем больше k, тем больше отношение размеров вершин с разным значением centrality.

Раскладка графа

Для раскладки графа используется алгоритм ForceAtlas2. В качестве параметров ему передаются:

- Количество итераций алгоритма. (Iteration)
- Сила притяжения вершин друг к другу. (Gravity)
- Активация/деактивация расчета с помощью логарифмической силы притяжения. (Logarithmic attraction mode)
- Активация/деактивация притяжения вершин к краям окна. (Outbound attraction mode)
- Активация/деактивация притяжения вершин к центру. (Strong gravity mode)

SQLite

Приложение имеет возможность сохранять и читать 2 типа SQLite баз данных (сохранение идет во 2 тип)

1) Необработанный граф. В таком случае в базе данных должно быть две таблицы

Vertices

id	element	community
integer	*text*	-1

Edges

id	element	first	second
integer	*text*	*int*	*int*

- `id` - идентификационный номер *(должен быть уникальным)*
- `first` - откуда идет ребро *(**id** вершины из первой таблицы)*
- `second` - куда идет ребро *(**id** вершины из первой таблицы)*

2. Обработанный граф. В таком случае должна быть третья таблица, а в таблице **Vertices** в поле **community** вместо -1 могут стоять непосредственно номера community

VerticesView

id	vertex	x	y	color
integer	*int*	*double*	*double*	*text*

- `id` должен быть уникален
- `vertex` - **id** вершины из таблицы **Vertices**
- `x, y` - координаты вершины
- `color` - цвет формата RGB в следующем виде "r/g/b", где r,g,b - double.

CSV-файлы

Приложение так же имеет возможность сохранять графы в файлы формата `.csv`, а также считывать их.

> Чтобы приложение могло считать граф, csv-файл должен соответствовать стандарту описанному ниже.

Заголовок:

```
isNode,name,x,y,color,radius,community,from,to
```

Остальные строчки должны представлять из себя заполненные (где это нужно) поля заголовка, разделенные запятой:

- `isNode` - значение true/false *(вершина либо ребро)*
- `name` - имя вершины/ребра *(уникальное поле)*
- `x`, `y` - координаты вершины *(необязательное поле в случае вершины, в случае ребра - пустое)*
- `color` - цвет вершины в формате "r/g/b", где "r", "g", "b" - числа из rgb представления цвета, а "/" - разделитель *(необязательное поле в случае вершины, в случае ребра - пустое)*
- `radius` - числовое представление радиуса вершины *(необязательное поле в случае вершины, в случае ребра - пустое)*
- `community` - номер сообщества вершины *(необязательное поле в случае вершины, в случае ребра - пустое)*
- `from`, `to` - имена вершин, которые соответствуют началу и концу ребра *(пустое для вершин поле)*