Podzielenie grafu na części
Dokumentacja imlpementacyjna

Tsimafei Lukashevich, Paweł Solecki

27.03.2025

1. Algorytm dzielenia grafu na k części - METIS

1.1. Krótki opis algorytmu

Celem algorytmu METIS jest podział nieskierowanego grafu $G = (V, E)$ na k części, minimalizując liczbę przeciętych krawędzi. Algorytm składa się z trzech głównych etapów:

1.1.1. Coarsening (scalanie wierzchołków)

Hierarchiczne redukowanie grafu przez łączenie wierzchołków o silnych połączeniach. Siłę połączeń określamy następująco:

- Wierzchołki są bezpośrednio połączone.
- Mają wielu wspólnych sąsiadów (np. wierzchołki w tej samej gęstej części grafu).

1.1.2. Initial Partitioning (wstępny podział)

Podział najmniejszej wersji grafu na k części za pomocą wybranej metody.

1.1.3. Refinement (ulepszanie podziału)

Optymalizacja podziału, aby zminimalizować liczbę przeciętych krawędzi.

1.2. Opis szczegółów działania poszczególnych etapów

1.2.1. Coarsening (scalanie wierzchołków):

1. Utwórz kopię grafu $G_c \leftarrow G$.
2. Dopóki graf G_c jest zbyt duży:
    - Wybierz niesparowany wierzchołek $v \in V$.
    - Znajdź sąsiada u maksymalizującego wagę krawędzi $w(v, u)$.
    - Połącz v i u w jeden superwierzchołek.
    - Zaktualizuj listę sąsiedztwa.

1.2.2. Initial Partitioning (wstępny podział):

Podziel zredukowany graf G_c na k części za pomocą wybranej metody.

- Podziel zredukowany graf G_c na k części.
- Wykorzystaj jedną z metod - losową, algorytm spektralny lub heurystykę optymalizacyjną.
- Każdemu wierzchołkowi w G_c przypisz jedną z k części, minimalizując sumę wag przeciętych krawędzi.

1.2.3. Refinement (ulepszanie podziału):

Dla każdego wierzchołka $v \in V$ w pełnym grafie G:

- Sprawdź, czy przeniesienie v do innej części zmniejsza liczbę przeciętych krawędzi.
- Jeśli przeniesienie poprawia wynik, przenieś v do nowej części.

Wynik: Przypisanie wierzchołków do k części.

2. Formaty plików

2.1. Format wejściowy

2.1.1. Plik `.csrrg`

Tekstowy plik wejściowy z rozszerzeniem `.csrrg` opisujący graf w formacie macierzowym, gdzie:

- **1 linia** - Rozmiar macierzy (maksymalna liczba węzłów w wierszu)
- **2 linia** - Indeksy węzłów w poszczególnych wierszach
- **3 linia** - Wskaźniki na pierwszy węzeł w wierszu (odnosi sie do 2 linii)
- **4 linia** - Grupy połączonych węzłów - pierwszy węzeł z grupy jest połączony z resztą grupy
- **5 linia** - Wskaźniki na pierwszy węzeł grupy (odnosi się do 4 linii)

Przykład pliku `.csrrg`

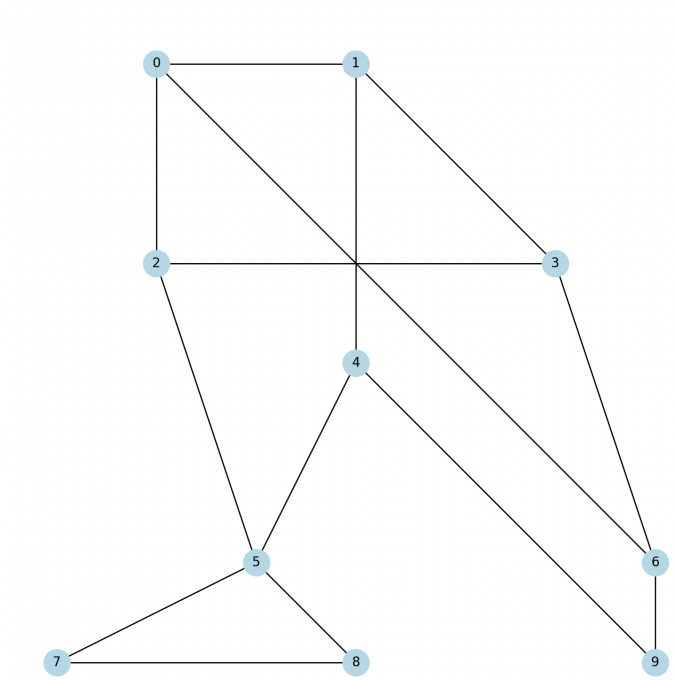

Figure 1: Graf opisany w pliku graf.csrrg

Plik `graf.csrrg`:

```
1 7
2 1;3;1;5;3;2;6;0;3;6
3 0;2;2;4;5;5;7;10
4 0;1;2;6;1;4;3;2;3;5;3;6;4;5;9;5;7;8;6;9;7;8
5 0;4;7;10;12;15;18;20
```

Wytłumaczenie rozmieszecznia węzłów w rzędach - opis linii 2 i 3

Pierwszy rząd

...

`1;3;`1;5;3;... - indeksy węzłów

`0;2;`2;4;5;5;... - zakres pierwszego rzędu

...

Drugi rząd

...

1;3;1;5;3;... - indeksy węzłów

0;`2;2;`4;5;5;... - zakres drugiego rzędu

...

Trzeci rząd

...

1;3;`1;5;`3;2;6;... - indeksy węzłów

0;2;`2;4;`5;5;... - zakres trzeciego rzędu

...

Wytłumaczenie grup połączonych węzłów - opis linii 4 i 5

Pierwsza grupa
...
`0;1;2;6;`1;4;3;2;3... - 0 jest połączone z 1, 2, 6
`0;4;`7;10;12;... - zakres grupy
...

Druga grupa
...
0;1;2;6;`1;4;3;`2;3... - 1 jest połączone z 4, 3
0;`4;7;`10;12;... - zakres grupy
...

2.1.2. Plik `.bin`

Plik binarny z rozszerzeniem `.bin` opsiujący graf. Przyjmowany format pliku jest identyczny jak binarny format wyjściowy opisany poniżej.

2.2. Format wyjściowy

Plik wyjściowy może być w wyżej pokaznym formacie `.csrrg` lub w formie binarnej `.bin`

Zapis binarny

Struktura pliku binarnego:

```
[naglowek] 14B
 |- endianness_flag 2B
 |- rozmiar_macierzy 2B
 |- offset_wskazniki_grup 2B
 |- offset_indeksy_wezlow 4B
 |- offset_grupy_krawedzi 4B
[wskazniki wierszy] (varint + delta)
[wskazniki grup] (varint + delta)
[indeksy wezlow] (varint)
[grupy krawedzi] (varint)
```

Struktura pliku binarnego jest podobna do pliku `.csrrg`. Linie pliku `.csrrg` odpowiadają:

- Linia 1 - `rozmiar_macierzy`
- Linia 2 - `[indeksy wezlow]`
- Linia 3 - `[wskazniki wierszy]`
- Linia 4 - `[grupy krawedzi]`
- Linia 5 - `[wskazniki grup]`

Dlaczego ta kolejność sekcji?

Nagłówek musi być pierwszy – zawiera kluczowe informacje do parsowania reszty pliku.
Offsety są zapisane przed danymi, aby parser mógł szybko obliczyć pozycje sekcji `[indeksy wezlow]` i `[grupy krawedzi]`.
Dane (indeksy węzłów i grupy krawędzi) są na końcu, ponieważ ich odczyt wymaga wcześniejszego wczytania offsetów.

Ta struktura gwarantuje minimalny rozmiar pliku i szybki dostęp do danych.

Nagłówek

`endianness_flag` (2B)

Pierwsze dwa bajty pliku stanowią znacznik kolejności bajtów (endianness_flag), który określa, czy plik został zapisany w tym samym systemie, w którym jest odczytywany. Domyślna wartość to **AB**.
- W systemie Little Endian (LE) liczby są zapisywane od najmniej znaczącego bajtu (LSB) do najbardziej znaczącego (MSB).
- W systemie Big Endian (BE) liczby są zapisywane odwrotnie – od MSB do LSB.

Obsługa odczytu:
1. Odczytaj pierwsze dwa bajty (endianness_flag).
2. Porównaj wartość:
- AB → Brak konieczności konwersji.
- BA → Wykonaj zamianę kolejności bajtów w polach wielobajtowych.
3. Przetwarzaj dane zgodnie z wykrytą kolejnością bajtów.

`rozmiar_macierzy` (2B)

Maksymalny rozmiar macierzy to 1024. Typ `uint16` (2B) wystarcza, ponieważ zakres 0–65 535 obejmuje maksymalną wartość 1024.

`offset_wskazniki_grup` (2B)

Wskaźnik na początek sekcji `[wskazniki grup]`

`offset_indeksy_wezlow` (4B)

Wskaźnik na początek sekcji `[indeksy wezlow]` **Maksymalna wartość**:
`rozmiar_nagłówka + rozmiar_wskaznikow_wierszy + rozmiar_wskaznikow_grup`

`offset_grupy_krawedzi` (4B)

Maksymalna wartość:
`offset_indeksy_wieszy + rozmiar_indeksy_wierszy`

W pliku nie ma informacji o tym kiedy zaczyna się sekcja `[wksazniki grup]`, ponieważ sekcja `[wskazniki wierszy]` ma zawsze `(rozmiar_macierzy + 1)` liczb.

Wskaźniki wierszy i grup

Delta + varint

1. Delta - kodowanie różnicowe

   Cel:
   Zmniejszenie rozmiaru danych poprzez zapisywanie różnic między kolejnymi wartościami (zamiast bezwzględnych wartości).

   Przykład:
   Zamiast zapisu: `0;4;7;10;12;15;18;20`
   Zapiszemy: `0;4;3;3;2;3;3;2`

2. Varint - (Variable-length Integer)

   Cel:
   Zmniejszenie rozmiaru liczb poprzez użycie 1–5 bajtów (w zależności od wielkości liczby).

   Zasady kodowania:
   Każdy bajt zawira **7 bitów danych** i **1 bit flagi** (MSB)
   `MSB = 1` - Następuje kolejny bajt
   `MSB = 0` - Ostatni bajt liczby

Indeksy węzłów i grupy krawędzi

Elementy indeksów węzłów i grup krawedzi są zapisywane jako **varint** (opisany powyżej).

3. Podział programu na moduły

3.1. Dane wejściowe

```
file_reader.h csrrg_reader.c bin_reader.c
```

3.2. Reprezentacja i przechowywanie grafu

```
graph.h graph.c
```

3.3. Scalanie wierzchołków

```
coarsening.h coarsening.c
```

3.4. Wstępny podział

```
partitioning.h partitioning.c
```

3.5. Ulepszanie podziału

```
refinement.h refinement.c
```

3.6. Dane wyjściowe

```
file_writer.h csrrg_writer.c bin_writer.c
```

3.7*. Współpraca modułów

```
main.c
```

4. Argumenty wywołania programu

Program akceptuje następujące argumenty wywołania:

- `-h / --help` - wyświetla insturkcję obługi programu
- `-o / --output` - umożliwia wybór nazwy pliku wyjściowego (np. --output wynik), domyślnie wynik jest zapisywany do pliku domyślnego w wybranym formacie.
- `-b / --binary` - określa format wyjściowy, zapisuje wynik do pliku binarnego. Domyślnie wynik jest zapisywany w formacie ASCII do pliku z rozszerzeniem `.csrrg`
- `-p / --parts` – liczba części podziału (np. -p 3, --parts 3), domyślnie 2.
- `-m / --margin` – maksymalny margines procentowy (np. -m 15, --margin 15), domyślnie 10%. Margines jest liczony od średniej (gdy graf zostanie podzielony na 4 i 6 węzłów, to margines błędu wynosi $20\% = 4/5 = 6/5$)
- `-f / --force` – pominięcie wbudowanego marginesu błędu, co pozwala na wykonanie podziału bez dodatkowych ograniczeń. W przypadku więcej niż jednego grafu w pliku program dzieli pierwszy graf (aby podzielić inny graf należy użyć `-g / --graph`
- `-g / --graph` - umożliwia wybór grafu do podzielenia w przypadku gdy w pliku jest więcej niż 1 graf
- `-v / --verbose` – szczegółowe logowanie przebiegu działania programu, w tym liczby iteracji i wyników pośrednich.

5. Przykładowe wywołania programu

Przykładowe wywołania programu:

- `./program graf.csrrg --parts 3 --margin 20 --output wynik`

  efektem będzie podzielenie grafu odczytanego z pliku `graf.csrrg` na `3` części tak, że liczba wierzchołków w powstałych częściach grafu nie będzie się różnić o więcej niż `20%`. Wynik zostanie zapisany do pliku `wynik.csrrg`.

- `./program graf.csrrg --parts 2 --force -]/-verbose --output wynik -b`

  efektem będzie podzielenie grafu na `2` części bez ograniczeń marginesu błędu, z aktywnym szczegółowym logowaniem i zapisaniem wyniku do pliku `wynik.bin` w formacie binarnym.

- `./program graf.bin --graph 2 --parts 4 --binary`

  efektem będzie podzielenie drugiego grafu z pliku `graf.bin` na `4` części, zapisując wynik w formacie binarnym.

6. Komunikaty błędów

Program nie przyjmuje argumentów w trakcie działania, więc mamy jedynie zadbać o poprawność danych wejściowych.

- [**Błąd 101**]: Niepoprawna definicja grafu: `Linia 29: Niepoprawna definicja krawędzi.`
- [**Błąd 102**]: Liczba podziałów poza zakresem: `Liczba podziałów musi być większa od 0 i mniejsza od liczby wierzchołków. Wczytano: "-1".`

- [**Błąd 103**]: Margines procentowy poza zakresem: `Margines procentowy musi być w zakresie 0-100. Wczytano: "123".`
- [**Błąd 104**]: Brak wymaganego argumentu wejściowego: `Nie podano pliku wejściowego zawierającego graf.`
- [**Błąd 105**]: Podany plik nie istnieje: `Nie można otworzyć pliku "graf.bin". Sprawdź ścieżkę dostępu.`
- [**Błąd 106**]: Wybór grafu poza zakresem: `W pliku znajduje się 5 grafów. Wczytano: "42". Popraw zakres numeracji.`