Współbieżny Algorytm Brandesa

Michał Kuźba

8 stycznia 2017

1 Idea

Pośrednictwo wierzchołka w grafie wyraża się wzorem:

$$BC(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

gdzie σ_{st} - liczba najkrótszych ścieżek między s i t, a $sigma_{st}(v)$ - liczba najkrótszych ścieżek między s i t, przechodzących przez v. Pośrednictwo wszystkich wierzchołków w sieci można obliczyć przy pomocy algorytmu Brandesa (Brandes 2001), którego złożoność obliczeniowa dla grafu nieważonego wynosi $O(|V| \cdot |E|)$. Ze względu na dużą niezależność przetwarzania wierzchołków algorytm Brandesa, może działać współbieżnie na wielu wątkach.

2 Rozwiązanie

2.1 Specyfikacja problemu

Wejście : nieważony graf skierowany, liczba wątków.
Wyjście: wskaźnik pośrednictwa, dla wszystkich wierzchołków, obliczony przy użyciu zadanej liczby wątków.

2.2 Implementacja

Zastosowane podejście wzoruje się na idei przedstawionej w (Bader and Madduri 2006, p. 539-550). Każdy wątek przetwarza oddzielny wierzchołek pobrany z kolejki oczekujących wierzchołków. Przetwarzanie odbywa się niezależnie, bez współdzielenia danych. Jedynym wyjątkiem jest mapa wskaźników pośrednictwa, która może być modyfikowana przez różne wątki, nawet dla tych samych kluczy. Stąd konieczność zastosowania wykluczania dostępu. W rozwiązaniu operacje na mapie są synchronizowane przy użyciu mutexa. Dynamiczny przydział zadań pobieranych z kolejki także wymusza synchronizację operacji na niej. Ważne jest również zapewnienie aby operacje odczytu danych z grafu, który

jest współdzielony nie powodowały żadnych zmian w strukturze - uzycie iteratorów dla hashmapy, tak aby zachować własność thread-safe.

Alternatywnie można przydzielić statycznie wierzchołki do odpowiednich wątków, co jednak może być wrażliwe na nierównomierny przydział zadań - wierzchołek o dużym stopniu wyjściowym wymaga więcej operacji niż wierzchołek izolowany.

W (Bader and Madduri 2006, p. 539-550) zaproponowano również współbieżne wykonanie obliczeń wewnątrz funkcji przetwarzającej wierzchołek. Dokładniej, wewnątrz części BFS, różne krawędzie wychodzące z wierzchołka mogą być przetwarzane oddzielnie. Podejście to ma jednak kilka wad, które wpływają na jego wydajność:

- konieczność współdzielenia danych - wzajemne wykluczanie
- niewielki koszt pojedynczej operacji - stąd relatywnie duży koszt przydzielania i synchronizacji

Ponadto, w zadaniu liczba wątków jest dokładnie określona, co utrudnia dynamiczny przydział zadań. Ostatecznie powyższe rozwiązanie jest mniej efektywne. Przyjęte rozwiązanie gwarantuje dobre przyspieszenie wraz z rosnącą liczbą wątków, (patrz sekcja 3), jednak posiada kilka wad:

- Użycie pamięci - przy przetwarzaniu każdego wierzchołka przechowujemy dane liniowe względem rozmiaru sieci - odległości, liczbę najkrótszych ścieżek. Stąd dla dużej liczby równoległych obliczeń, pojawić się mogą problemy z brakiem pamięci.
- Brak współbieżności wewnątrz przetwarzania pojedynczego wierzchołka.

3 Wydajność

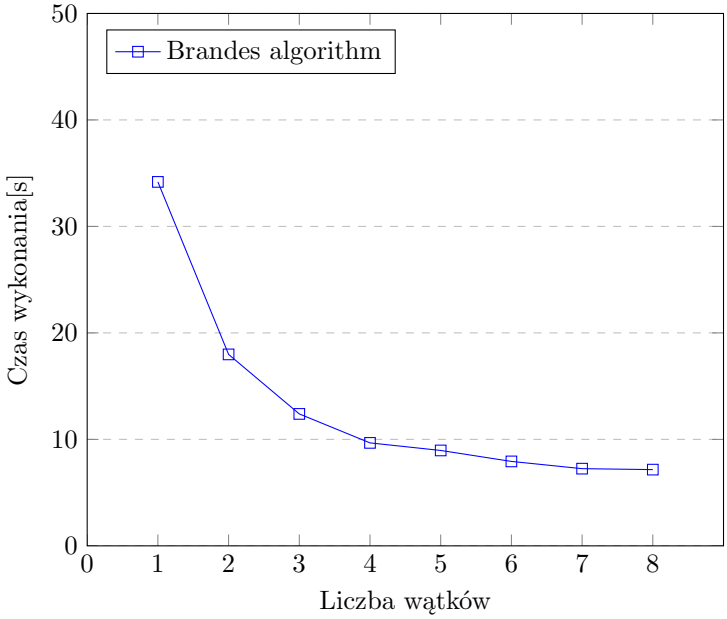
Ze względu na niezależność przetwarzania wierzchołków, co stanowi główną operację całego programu, dla dużych sieci, dostajemy znaczące przyspieszenie.

Można wyodrębnić kilka przyczyn, dla których przyspieszenie jest niższe niż liniowe (proporcjonalne do liczby wątków):

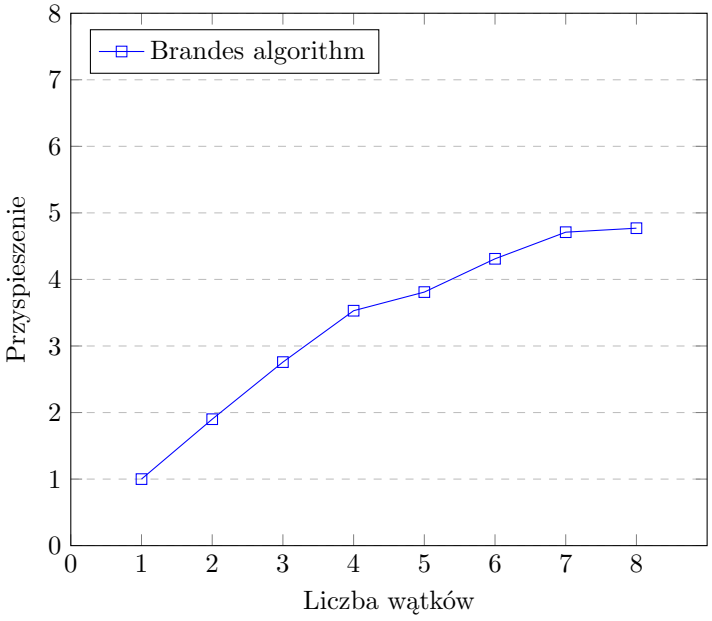
- przetwarzanie wierzchołków nie jest jedyną częścią wykonywania programu, tzn. sekwencyjne pozostają: wczytanie i wypisanie wyników.
- koszt synchronizacji mapy rezultatów oraz kolejki wierzchołków do przetworzenia - wykluczanie przy użyciu mutexa, czyli sekwencyjne przydzielanie zadań oraz aktualizacja wyników.

W przykładzie (rzadki graf, ok. 7,000 wierzchołków i 100,000 krawędzi) speed-up dla liczby wątków od 1 do 8:

Czas wykonania w zależności od liczby wątków



Przyspieszenie w zależności od liczby wątków



4 Podusmowanie

Zaproponowana implementacja pozwala kilkukrotnie szybciej obliczać pośrednictwo przy użyciu algorytmu Brandesa. Można ją rozszerzyć o dodatkowy poziom współbieżności lub przyspieszyć rozwiązanie standardowych problemów kolekcji thread-safe.

References

Bader, David A. and Kamesh Madduri (2006). "Parallel Algorithms for Evaluating Centrality Indices in Real-world Networks". In: *Proceedings of the 2006 International Conference on Parallel Processing*. ICPP '06. Washington, DC, USA: IEEE Computer Society, pp. 539–550. ISBN: 0-7695-2636-5. DOI: 10.1109/ICPP.2006.57. URL: http://dx.doi.org/10.1109/ICPP.2006.57.

Brandes, Ulrik (2001). "A Faster Algorithm for Betweenness Centrality". In: *Journal of Mathematical Sociology* 25, pp. 163–177.