Program implementuje algorytm B-adoratorów na wielu wątkach. Dla każdej z wartości [0, blimit] tworzy thread_count-1 wątków I wywołuje funkcję wyszukującą partnerów dla każdego wierzchołka.

Wierzchołki przechowywane są w dwóch atomowych kolejkach. Graf jest reprezentowany jako wektor wektorów sąsiedztwa. Wektory sąsiedztwa obudowane są klasą, która umożliwa częściowe sortowanie I pozyskiwanie w czasie stałym najcięższej krawędzi. Dla każdego wierzchołka przechowywana jest również kolejka priorytetowa jego partnerów oraz liczba wierzchołków, których brakuje mu do osiągnięcia bvalue.

Dostęp do kolejki partnerów ograniczony jest dla każdego wierzchołka mutexem. Liczba brakujących wierzchołków przechowywana jest jako atomic.

Podczas szukania odpowiedniego partnera wyciągane są kolejno najcięższe krawędzie z wektora sąsiedztwa. Jeśli wyciągnięty wierzchołek ma jeszcze miejsce lub jego najgorsza propozycja jest gorsza od naszej, to wierzchołek jest zwracany jako potencjalny dobry partner I zajmujemy jego mutexa. Po zajęciu mutexa ponownie sprawdzamy, czy wierzchołek może być dobrym partnerem. Jeśli tak jest, to obniżamy liczbę wierzchołków, których jeszcze nam brakuje, dodajemy swój wierzchołek do listy partnerów znalezionego partnera I ewentualnie wyrzucamy jego najgorszą propozycję, jeśli miał ich już pełną liczbę. Wówczas dodatkowo zwiększamy liczbę brakujących partnerów wierzchołkowi wyrzuconemu I wrzucamy go do drugiej kolejki. Po rozważeniu każdego sąsiada wyrzucamy go z listy sąsiadów.

Po wyciągnięciu wszystkich wierzchołków z pierwszej kolejki, zamieniamy ich zawartości I powtarzamy cały proces, aż obie nie będą puste.

Na końcu zliczamy wagi krawędzi, opróżniając jednocześnie listy partnerów.

Zastosowane optymalizacje

1. Lista sąsiadów dla każdego wierzchołka jest sortowana w trakcie tworzenia grafu. Dla każdego wierzchołka w czasie stałym mamy dostęp do jego najcięższego sąsiada.

2. Dla każdego wierzchołka każdy jego sąsiad może być rozważony jako adorator maksymalnie raz. Struktura przechowująca sąsiadów przechowuje iterator do ostatniego jeszcze nierozważonego elementu I przesuwa go przy sprawdzaniu kolejnych wierzchołków.

3. Sortowanie częściowe sąsiadów z odległością równą 2 * bvalue(v) dla każdego wierzchołka v.

Speed-up

Program testowany na grafie roadNet_PA z biblioteki SNAP (w grafie zostały dodane wagi I usunięte podwójne krawędzie), blimit = 20, na maszynie students.

Thread count	time	speed-up
1	2m0,089s	1

2	2m2,670s	0,978959811
3	2m1,959s	0,984666978
4	2m1,541s	0,988053414
5	1m56,458s	1,031178622
6	1m53,352s	1,059434328
7	1m49,989s	1,091827365
8	1m47,960s	1,112347166