Параллельная обработка больших графов

Занятие 5

А.С. Семенов

dislab.org

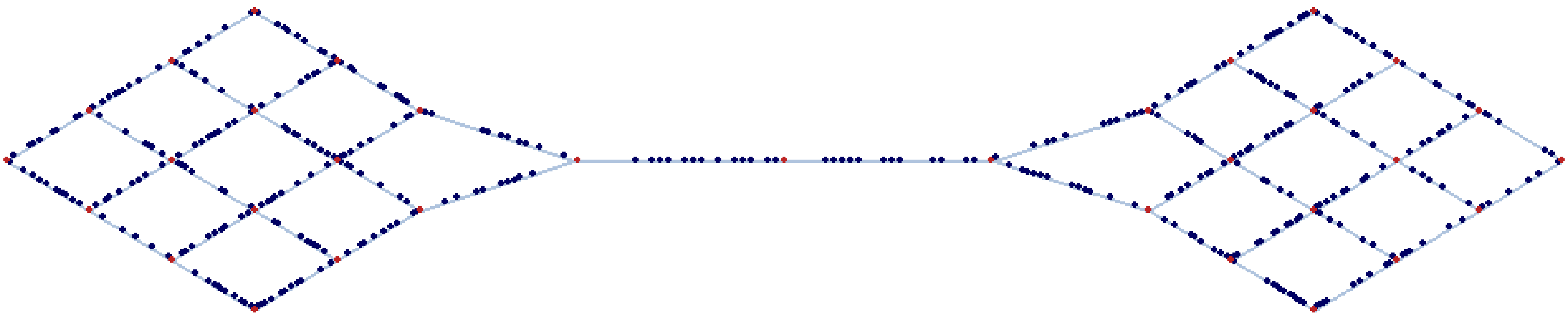
Betweenness centrality

- Связный неориентированный граф
- Допустимы циклы и кратные ребра
- w(e) > 0, e in E
- S_{st} – количество кратчайших путей между $s, t \in V$
- $S_{st}(v)$ – количество кратчайших путей, проходящих через v

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

Не удается отобразить рисунок.

Betweenness centrality



«Наивный» алгоритм

1. Подсчет количества и длины путей в графе между любыми двумя вершинами
2. Вычисление значения характеристики для каждой вершины

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

«Наивный» алгоритм
1. Подсчет количества путей

Множество предков вершины v на кратчайших путях от s:

$$P_s(v) = \{u \in V : (u, v) \in E, d_G(s, v) = d_G(s, u) + w(u, v)\}$$

Какое количество кр. путей σ_{sv} будет от *s* до *v*, если известно количество кр. путей σ_{su} от *s* до всех *u*, *u* $\in P_s(v)$?

«Наивный» алгоритм
1. Подсчет количества путей

Лемма. $$\sigma_{sv} = \sum_{u \in P_s(v)} \sigma_{su}$$

«Наивный» алгоритм
1. Подсчет количества путей

Лемма. $$\sigma_{sv} = \sum_{u \in P_s(v)} \sigma_{su}$$

Следствие. Из вершины *s* можно определить длины и количество всех кратчайших путей до всех остальных вершин за *O(M)* для графов без весов.

«Наивный» алгоритм
2. Подсчет характеристики

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

Как определить, что кратчайший путь проходит через вершину v?

«Наивный» алгоритм
2. Подсчет характеристики

Критерий Беллмана. Вершина v лежит на кратчайшем пути между вершинами s,t iff

$$d_G(s,t) = d_G(s,v) + d_G(v,t)$$

$$\sigma_{st}(v) = \begin{cases} 0 & \text{if } d_G(s,t) < d_G(s,v) + d_G(v,t) \\ \sigma_{sv} \cdot \sigma_{vt} & \text{otherwise} \end{cases}$$

«Наивный» алгоритм

1. Подсчет количества и длины путей в графе между любыми двумя вершинами
2. Вычисление значения характеристики для каждой вершины

«Наивный» алгоритм

1. Подсчет количества и длины путей в графе между любыми двумя вершинами
2. Вычисление значения характеристики для каждой вершины

Сложность

– Память **$O(N^2)$**
– Время **$O(N^3)$**

Алгоритм U. Brandes (2001)

<u>BC-Brandes</u> (G) // G – связный неориентированный без весов
for all v∈V **do** BC[v] = 0 **end for**
for all s∈V **do**
    BrandesInitS (s)
    ShortestPathsCounting (s)
    DependencyAccumulation ()
end for

Алгоритм Brandes, инициализация

BrandesInitS (s)
initEmptyStack(S) // S - пустой стек
for all t∈V **do**
    initEmptyList(P[t]);
    $s[t]=0; d[t]=-1; d[v]=0$
end for
$\sigma[s]=1; d[s]=0$

Подсчет числа кратчайших путей

```
ShortestPathsCounting (s)
initQueue(Q, s) // Q – очередь
while Q <> {}
    v = deque(Q);   push(S, v)
    for all w∈Adj[v]
        if d[w] < 0
            enqueue(Q, w)
            d[w] = d[v]+1
        end if
        if d[w] == d[v]+1
            s[w] = s[w]+s[v]
            append(P[w], v)
        end if
    end for
end while
```

Суммирование

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

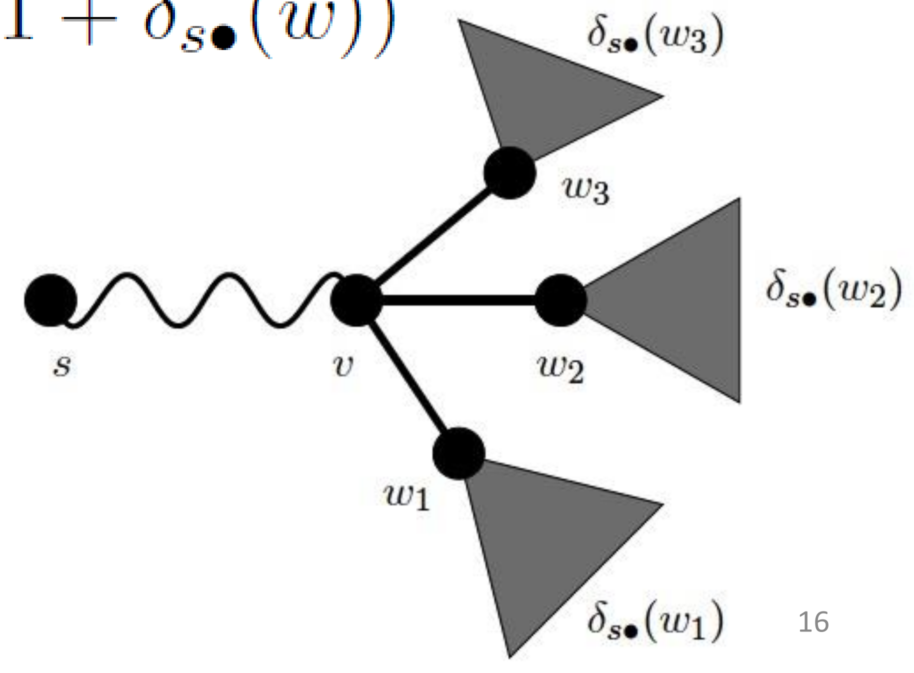
$$C_B(v) = \sum_{s \neq v \neq t \in V} \delta_{st}(v)$$

Суммирование

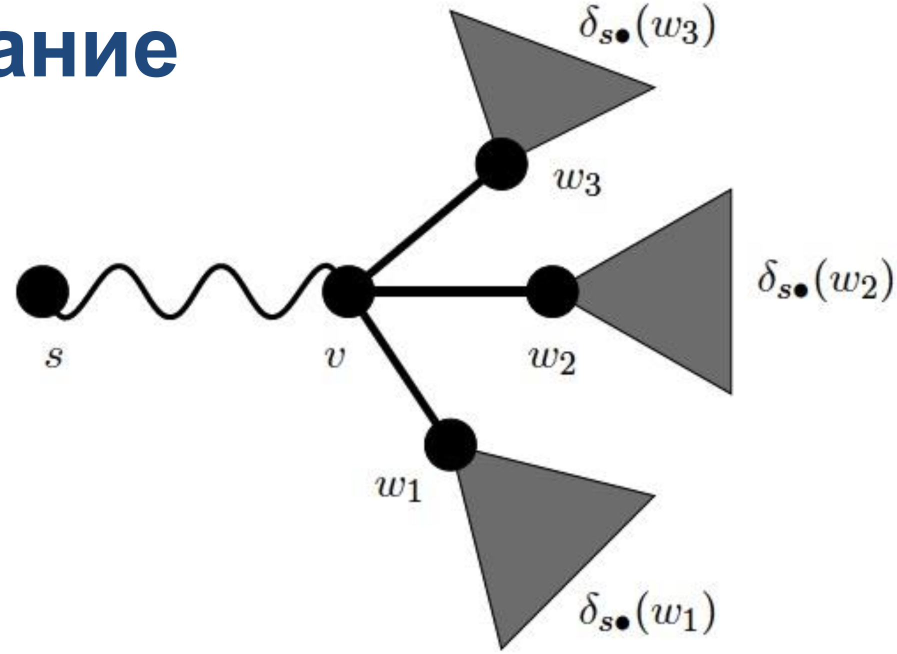
$$\delta_{st}(v) = \frac{\sigma_{st}(v)}{\sigma_{st}} \qquad \delta_{s\bullet}(v) = \sum_{t \in V} \delta_{st}(v)$$

Лемма. Если существует только один кратчайший путь из s в t, то

$$\delta_{s\bullet}(v) = \sum_{w\,:\,v \in P_s(w)} (1 + \delta_{s\bullet}(w))$$



Суммирование



$$W = \{w : v \in P_s(w)\}$$

$$\delta_{s\bullet}(v) = \sum_{t \in V} \delta_{st}(v) =$$

$$\sum_{w \in W} \delta_{sw}(v) + \sum_{w \notin W} \delta_{sw}(v) =$$

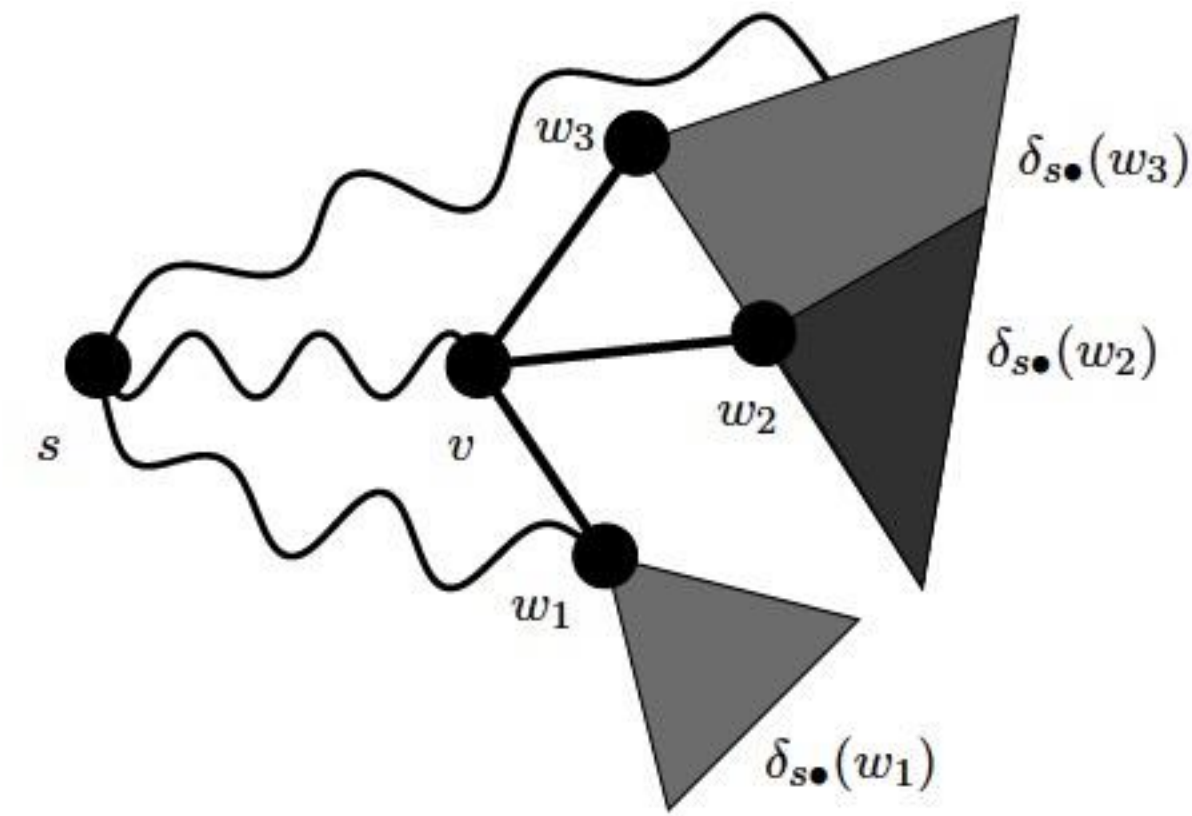
$$|W| + \sum_{w \in W} \delta_{s\bullet}(w) =$$

$$\sum_{w \in W} (1 + \delta_{s\bullet}(w))$$

Суммирование

Теорема.

$$\delta_{s\bullet}(v) = \sum_{w\,:\,v \in P_s(w)} \frac{\sigma_{sv}}{\sigma_{sw}} \cdot (1 + \delta_{s\bullet}(w))$$



Суммирование

<u>DependencyAccumulation</u> ()

for all $v \in V$ **do** $\delta[v] = 0$

while S <> {}

    w = pop(S)

    for all *v*∈P[*w*]

$$\delta[v] = \delta[v] + \frac{\sigma[v]}{\sigma[w]} \cdot (1 + \delta[w])$$

    end for

    if *w* <> *s*

$$BC[w] = BC[w] + \delta[w]$$

    end if

end while

Сложность алгоритма Brandes

BC-Brandes (G) // G – связный неориентированный без весов

for all v∈V **do** BC[v] = 0 **end for**

for all s∈V **do**

    BrandesInitS (s)

    ShortestPathsCounting (s)

    DependencyAccumulation ()

end for

Сложность:

- Время **O(N M)**
- Память **O(N+M)**