Paralelní grafové algoritmy

Značení

Minimální kostra grafu

Nejkratší cesta z jednoho uzlu

Nejkratší cesta mezi všemi dvojicemi uzlů

Použité značení

Definition

Buď $G = (V, E)$ graf. Pro libovolný uzel $u \in V$ budeme značit:

- $N_V(u)$ množinu všech sousedních uzlů k uzlu u, tj.

$$N_V(u) \equiv \{v \in V \mid \exists (u, v) \in E\},$$

- $N_E(u)$ množinu všech přilehlých hran k uzlu u, tj.

$$N_E(u) \equiv \{(u, v) \mid v \in V \wedge (u, v) \in E\}.$$

Minimální kostra grafu

Definition

Mějme souvislý graf $G(V, E)$. **Kostrou grafu** G nazveme podgraf $G'(V, E')$, který obsahuje všechny vrholy grafu G a je stromem. Má-li navíc graf G vážené hrany (jejich váhy budeme značit $w(e)$ pro každé $e \in E$), pak **minimální kostra** je taková kostra, pro kterou je suma

$$w(E') = \sum_{e \in E'} w(e),$$

minimální.

Minimální kostra grafu

Aplikace:

- návrh komunikačních sítí
- LDPC kódy pro korekce chyb
- zpracování obrazu (registrace obrazu pomocí Renyiho entropie)
- generování náhodného bludiště

Úlohu lze řešit pomocí Dijkstrova algoritmu se složitostí $O(n^2)$.

Jarníkův (Primův) algoritmus

1: **procedure** JARNÍK(V, E, w, r))
2: $V_F := \{r\}$
3: $E_T = \emptyset$
4: **for all** $v \in V \setminus V_F$ takové, že $(r, v) \in E$ **do**
5: $E_T := E_T \bigcup \{(r, v)\}$
6: **end for**
7: **while** $V_F \neq V$ **do**
8: najdi hranu $(u, v) \in E_T$ s minimální váhou a $u \in V_F$
9: $V_F := V_F \bigcup \{v\}$
10: $E_T := E_T \setminus (u, v)$
11: **for all** $u \in V \setminus V_F$ takové, že $(u, v) \in E$ **do**
12: $E_T := E_T \setminus (u, v)$
13: **end for**
14: **end while**
15: **return** V_F
16: **end procedure**

Paralelní Jarníkův algoritmus

- paralelizovat lze pouze hledání minimální hrany na řádku 8 a přidávání hran přílehlých množině V_F do E_T na řádku 11
- zbytek algoritmu je čistě sekvenční
- v případě řídkého grafu je to velmi málo paralelismu
  - množina E_T na řádku 8 je malá
  - for cyklus na řádku 11 je krátký
  - Bader A.B., Cong G., *Fast shared-memory algorithms for computing the minimum spanning forest of sparse graphs*, Journal of Parallel and Distributed Computing, 66, 2006, pp. 1366–1378.

Nejkratší cesta z jednoho uzlu

Definition

Mějme souvislý graf $G(V, E)$ a zvolme v něm jeden vrchol jako startovní. Chceme najít nejkratší cestu do všech ostatních uzlů.

Aplikace:

- navigace, jizdní řády
- zpracování obrazu
- řešení PDR - eikonální a Hamiltonovy-Jacobiho rovnice (fast-marching method)
- dynamické programování
- porovnávání řetězců v bioinformatice
- program `diff`

Dijkstrův algoritmus

1: **procedure** DIJKSTRA(V, E, w, s))
2: $V_F := \{s\}$, $V_T := \emptyset$, $l[s]$
3: **for all** $v \in V \setminus V_F$ **do**
4: **if** existuje hrana (s, v) **then**
5: $l[v] := w(s, v)$, $V_T := V_T \bigcup \{v\}$
6: **else**
7: $l[v] := +\infty$
8: **end if**
9: **end for**
10: **while** $V_F \neq V$ **do**
11: najdi $u \in V_T$, že $l[u] = \min_{v \in V_T} l[v]$
12: $V_F := V_F \bigcup \{u\}$
13: **for all** $v \in N_V(u) \setminus V_F$ **do**
14: $l[v] := \min\{l[v], l[u] + w(u, v)\}$
15: $V_T := V_T \bigcup \{v\}$
16: **end for**
17: **end while**
18: **return** l
19: **end procedure**

Paralelní Dijkstrův algoritmus

- paralelizace Dijkstrova algoritmu je obdobná paralelizaci Jarníkova algoritmu
- paralelizovat lze pouze řádky 11 a cyklus na řádku 13
- je-li graf V řídký, je to opět problém
  - množina V_T na řádku 11 malá
  - for cyklus na řádku 13 je krátký
  - Jasika N., Aslisphaic N., Elma. A, Ilvana K., Elma L., Nosovic N., *Dijkstra's shortest path algorithm serial and parallel execution performance analysis*, MIPRO, 2012 Proceedings of the 35th International Convention, 2012.

Nejkratší cesta mezi všemi dvojicemi uzlů

Místo pole $l[u]$ udávající délku nejkratší cesty z určitého uzlu nyní budeme hledat matici $\mathbb{D}$, pro niž platí, že $D_{u,v}$ je délka nejkratší cesty z uzlu u do uzlu v.

Dijkstrův algoritmus

Úlohu lze řešit pomocí n-krát aplikovaného Dijkstrova algoritmu, tj. se složitostí $O(n^3)$.
Paralelizaci lze provést dvěma způsoby:

- **paralelizace přes zdrojové vrcholy** – pro každý vrchol grafu G spustíme paralelně sekvenční verzi Dijkstrova algoritmu
- **paralelizace dekompozicí grafu** – postupně (sekvenčně) řešíme úlohu pro jeden uzel za druhým a použijeme paralelní verzi Dijkstrova algoritmu
- **kombinujeme oba přístupy**

Floydův algoritmus

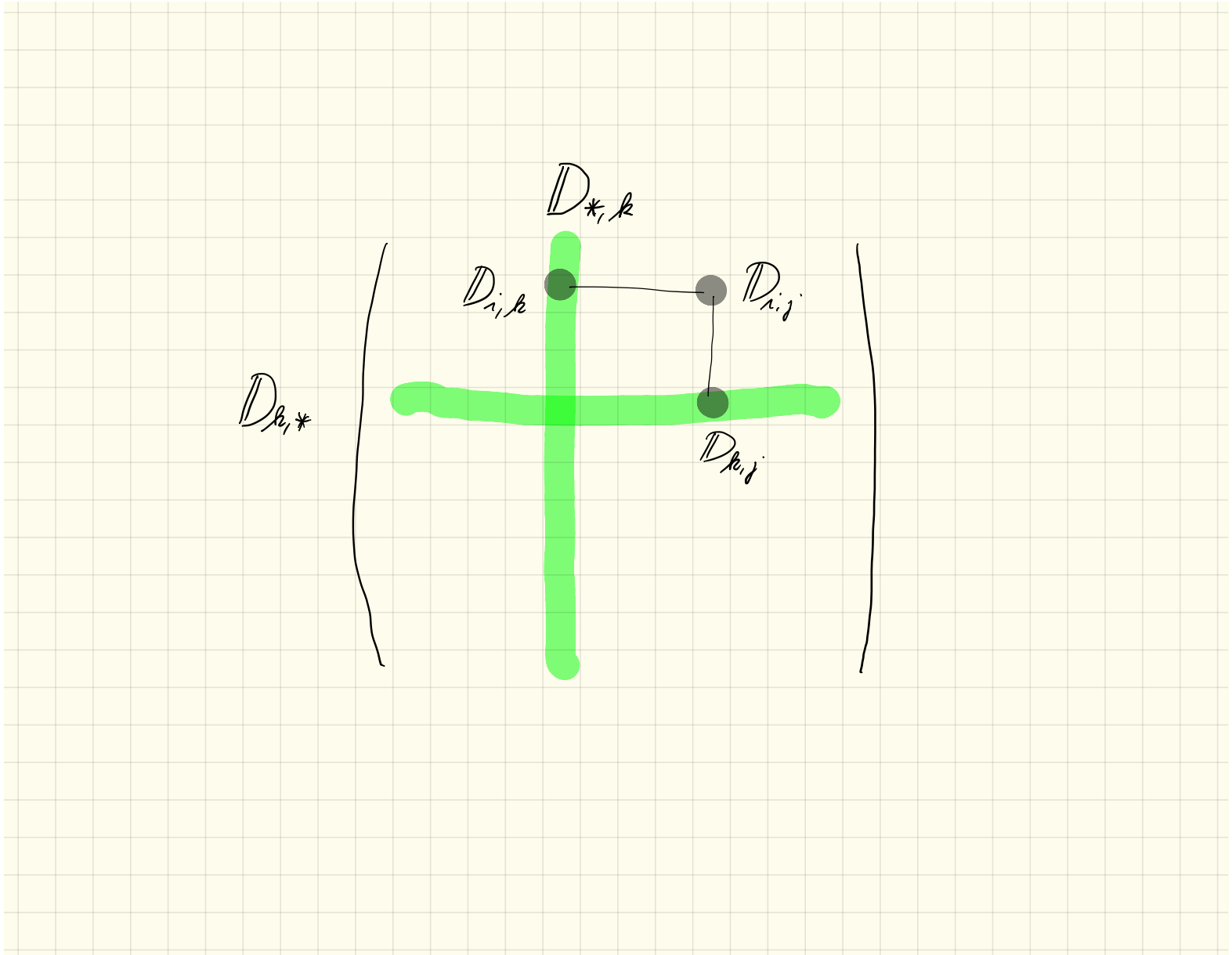
Floydův algoritmus je založen na následující myšlence:

- buď d_{uv} doposud nejkratší známá cesta mezi uzly u a v
- jinou kratší cestu můžeme najít tak, že objevíme uzel w, pro který platí $d_{u,w} + d_{w,v} < d_{u,v}$
- pak položíme $d_{uv} = d_{u,w} + d_{w,v}$
- začneme s adjacenční maticí $\mathbb{A}$ grafu G
  - ta obsahuje první odhad nejkratších cest mezi libovolnou dvojicí uzlů
  - uzly, mezi kterými nevede hrana, mají délku nejkratší cesty $+\infty$
- bereme jeden uzel po druhém a zkoušíme, které cesty v grafu lze s jeho pomocí zkrátit

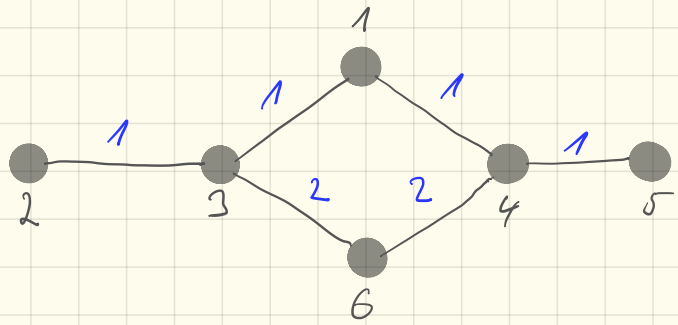
Floydův algoritmus

```
 1: procedure FLOYD( V, E))
 2:     D^0 := A_{G(V,E)}
 3:     for k := 1 to n do
 4:         for i := 1 to n do
 5:             for j := 1 to n do
 6:                 d^k_{i,j} := min{ d^{k-1}_{i,j}, d^{k-1}_{i,k} + d^{k-1}_{k,j} }
 7:             end for
 8:         end for
 9:     end for
10:     return D^n
11: end procedure
```

Floydův algoritmus



Floydův algoritmus



$$A = \begin{pmatrix} 0 & 1 & 1 & & & \\ 1 & 0 & & & & 2 \\ 1 & & 0 & 1 & 2 & \\ & & 1 & 0 & & \\ & & 2 & 2 & & 0 \end{pmatrix} = D^0$$

$$D^1 = \begin{pmatrix} 0 & 1 & 1 & & & \\ 1 & 0 & 1 & & & \\ 1 & 1 & 0 & 2 & & 2 \\ 1 & & 2 & 0 & 1 & 2 \\ & & & 1 & 0 & \\ & & 2 & 2 & & 0 \end{pmatrix} \quad D^2 = \begin{pmatrix} 0 & 1 & 1 & & & \\ 1 & 0 & 1 & & & \\ 1 & 1 & 0 & 2 & & 2 \\ 1 & & 2 & 0 & 1 & 2 \\ & & & 1 & 0 & \\ & & 2 & 2 & & 0 \end{pmatrix} \quad D^3 = \begin{pmatrix} 0 & 2 & 1 & 1 & & 3 \\ 2 & 0 & 1 & 3 & & 3 \\ 1 & 1 & 0 & 2 & & 2 \\ 1 & 3 & 2 & 0 & 1 & 2 \\ & & & 1 & 0 & \\ 3 & 3 & 2 & 2 & & 0 \end{pmatrix}$$

$$D^4 = \begin{pmatrix} 0 & 2 & 1 & 1 & 2 & 3 \\ 2 & 0 & 1 & 3 & 4 & 3 \\ 1 & 1 & 0 & 2 & 3 & 2 \\ 1 & 3 & 2 & 0 & 1 & 2 \\ 2 & 4 & 3 & 1 & 0 & 3 \\ 3 & 3 & 2 & 2 & 3 & 0 \end{pmatrix} \quad D^5 = \begin{pmatrix} 0 & 2 & 1 & 1 & 2 & 3 \\ 2 & 0 & 1 & 3 & 4 & 3 \\ 1 & 1 & 0 & 2 & 3 & 2 \\ 1 & 3 & 2 & 0 & 1 & 2 \\ 2 & 4 & 3 & 1 & 0 & 3 \\ 3 & 3 & 2 & 2 & 3 & 0 \end{pmatrix} \quad D^6 = \begin{pmatrix} 0 & 2 & 1 & 1 & 2 & 3 \\ 2 & 0 & 1 & 3 & 4 & 3 \\ 1 & 1 & 0 & 2 & 3 & 2 \\ 1 & 3 & 2 & 0 & 1 & 2 \\ 2 & 4 & 3 & 1 & 0 & 3 \\ 3 & 3 & 2 & 2 & 3 & 0 \end{pmatrix}$$

Floydův algoritmus paralelně

- paralelizovat lze for cykly na řádcích 4 a 5

Tranzitivní uzávěr

Definition

Graf je **tranzitvní** pokud platí

$$(u, v) \in E \wedge (v, w) \in E \Rightarrow (u, w) \in E.$$

Definition

Tranzitivní uzávěr grafu G je nejmenší tranzitvní graf, který obsahuje G jako podgraf.

Tranzitivní uzávěr lze sestrojit tak, že doplníme hrany mezi každou dvojici uzlů u a v, právě když v grafu G existuje cesta z u do v. To lze udělat pomocí Floydova algoritmu.