Kapitola 1

Složitost

V této kapitole ukážeme několik výsledků ohledně časové a prostorové složitosti. Problém klastrové rovinnosti patří do třídy NP z pohledu časové složitosti a z hlediska prostového se dá řešit v prostoru $\mathcal{O}(n)$ na deterministiském stroji. Jako výchozí model uvažujeme RAM, případně jeho nedeterministickou verzi nRAM.

1.1 Datová reprezentace

Nejprve uvedeme možnosti reprezentace klastrového grafu a ujasníme vzhledem k čemu budeme vztahovat příslušnou složitost. Velikost grafu na vstupu je $\mathcal{O}(n + m + \|C\|)$, kde C je klastrová hierarchie. Jelikož graf musí být rovinný, platí pro m, že $m = \mathcal{O}(n)$. Nejprve musíme určit kolik klastrů se v klastrové hierarchii může nacházet. Pro zjednodušení budeme předpokládat, že klastr může obsahovat jediný vrchol nebo i všechny.

Tvrzení 1. *Maximální počet klastrů v grafu G s n ($n \geq 1$) vrcholy je 2n-1.*

Důkaz. Důkaz indukcí podle n:
Základ indukce : n=1
Zjevně platí.
Indukční předpoklad: Tvrzení platí pro $|V| <$ n.
Indukční krok:
BÚNO: každý vrchol je minimálně obsažen v klastru obsahující pouze jej. Díky tomuto pak každý klastr, který obsahuje aspoň dva vrcholy se rozkládá aspoň na dva menší obsahující méně vrcholů.

Máme graf s n vrcholy. Podle předpokladu máme klastr K obsahující všechny vrcholy. Ten obsahuje k vzájemně disjunktních podklastrů (takových, že už jediný klastr, ve kterém jsou obsaženy je K). Velikost i-tého klastru nechť je k_i Každý z těchto klastrů obsahuje méně než n vrcholů. Platí pro ně tedy indukční předpoklad. Máme tedy:

$$\#\text{ max. počet klastrů} = 1 + \sum_{i=1}^{k}(2 * k_i - 1) = 1 + 2 * \sum_{i=1}^{k} k_i - k = 2n - k + 1$$

K maximalizování dojde pokud bude vždy k=2. □

Pro reprezentaci klastrové hierarchie se nabízí dvě možnosti.

1. Seznamy vrcholů
2. Strom, kde listy představují vrcholy a vnitřní uzly představují klastry
   - Zde předpokládejme, že kořen tohoto stromu reprezentuje klastr obsahující všechny vrcholy.

První varianta má za následek, že klastrová hierarchie zabírá prostor až $\mathcal{O}(n^2)$. Příkladem takové klastrové hierarchie je graf, kde klastry jsou postupně do sebe vnořené. První klastr obsahuje všechny vrcholy, druhý o vrchol méně, třetí o další vrchol, ... Druhá varianta naproti tomu dává prostor $\mathcal{O}(n)$. Stačí tedy určovat složitost (časovou a paměťovou) vzhledem k počtu vrcholů vstupního grafu.

1.2 časová složitost

Hlavním výsledkem této části je lineární nedeterministický algoritmus pro klastrovou rovinnost.

Tvrzení 1. *Problém rozhodnotí existence rovinného klastrového nakreslení patří do třídy NP.*

Důkaz. Využíváme toho, že ekvivalentním problémem ke klastrové rovinnosti, je existence saturátoru. Ten nám zajistí, že klastry jsou souvislé. Saturátor dostaneme jako certifikát. Vzhledem k tomu, že klastrů je polynomiálně mnoho, tak ověření saturátoru se dá provést v polynomiálním čase (Například otestování souvislosti každého klastru zvláště). Dále jsou algoritmy testují klastrovou rovinnost v polynomiálním čase (TODO doplnit reference), pokud klastry jsou souvislé. □

Tvrzení 1. *Pro problém klastrové rovinnosti je nedeterministický algoritmus, jehož časová složitost je* $\mathcal{O}(n)$.

Důkaz. Důkaz tohoto tvrzení je pouze doplněním důkazu, že klastrová rovinnost je v NP. Pro důkaz je třeba ukázat, že umíme ověřit souvislost všech klastrů v čase $\mathcal{O}(n)$, a pak že klastrová rovinnost se dá otestovat v lineárním čase pokud jsou klastry souvislé. Druhá část viz (TODO doplnit reference)

Prosté otestování všech klastrů zvlášť na souvislost vede na algoritmus s časovou složitostí $\mathcal{O}(n^2)$, protože klastrů je až lineárně mnoho. Pro zlepšení půjdeme cestou, kdy budeme kontrahovat hrany, to využijeme k tomu, že budeme kontrahovat vrcholy menších klastrů a postupně tak zkontrahujeme všechny klastry. Jako datovou strukturu pro klastrovou hierarchii stromovou reprezentaci.

Při testování klastru na souvislost použijeme klasický algoritmus na testování souvislosti (po zjištění, že klastr je souvislý jej zkontrahujeme), jen hrany, které vedou ven (při průchodu si je zapamatujeme) z klastru aktualizujeme (nasměrujeme je do nového vrcholu vzniklého kontrakcí klastru). Časová složitost pro jeden klastr C je $\mathcal{O}(n_C+m_C+\#\text{počet hran ven z klastru})$, kde n_C je počet vrcholů klastru a m_C je počet hran mezi vrcholy klastru. Problémem je, že tohle stále vede na algoritmus s kvadratickou časovou složitostí (TODO obrázek klastrového klastru dosvědčující tuto složitost). Problémem je, že se hrany můžou aktualizovat příliš často.

Zdánlivě jsme si nepomohli, pokud bychom dovedli aktualizaci hrany provést tak, aby jsme při příští navštěvě hrany ji zkontrahovali. To provedeme následovně: Pro každý klastr budeme mít pomocný graf, kde vrcholy představují podklastry daného klastru (maximální na inkluzi). Hrany v těchto pomocných grafech představují hrany jdoucí mezi klastry. Hrany se budou nastavovat jakožto ty aktualizované. Abychom mohli určit, kam aktualizovaná hrana patří, tak potřebujeme určit nejmenší klastr, který sdílejí vrcholy příslušné hrany. Navíc také potřebujeme znát podklastry, kam vrcholy patří. To je ale problém nejmenšího společného předka v zakořeněném stromu, kdy potřebné dotazy se provádí v konstantním čase a lineárním předvýpočtem a využívající lineární prostor. Jednoduchou úpravou získame i ty potřebné informace.

Náš algoritmus tedy napřed provede DFS, přičemž se hrany rozdělí do pomocných grafů. Následně se pro každý pomocný graf provede test souvislosti. První část algoritmu pracuje v čase $\mathcal{O}(n)$ díky tomu, že aktualizaci umíme provést v konstantním čase a hran je pouze $\mathcal{O}(n)$. Druhá část algoritmu pra-

cuje v čase $\sum_{C \in \mathcal{C}} (n_C + m_C) = \sum_{C \in \mathcal{C}} n_C + \sum_{C \in \mathcal{C}} m_C \leq |\mathcal{C}| + m = \mathcal{O}(n) + \mathcal{O}(n) = \mathcal{O}(n)$ □

1.3 prostorová složitost

Z výsledků o časové složitosti můžeme říci, že můžeme klatrovou rovinnost rozhodovat v nedeterministickém prostoru o velikosti $\mathcal{O}(n)$. Ze Savitchovy věty (doplnit ref) plyne, že v deterministickém prostoru stačí nejvýše prostor velikosti $\mathcal{O}(n^2)$. Lepšího výsledku, ve smyslu, že potřebujeme méně prostoru, dosáhneme využítím vztahu tříd NTIME a DSPACE, který je $NTIME(t(n)) \subseteq DSPACE(t(n))$. Jelikož máme nedeterministický algoritmus pro klastrovou rovinnost pracující v lineárním čase, tak díky předešlému víme, že na existuje deterministický algoritmus využívající pouze lineárně mnoho prostoru.

Tvrzení 1. *Klastrová rovinnost lze rozhodovat na RAMu s lineárně omezeným prostorem.*

Důkaz. □