Identifikácia zhlukov v grafovej reprezentácii genómov

Študent: Eva Herencsárová
Školiteľka: doc. Mgr. Bronislava Brejová, PhD.

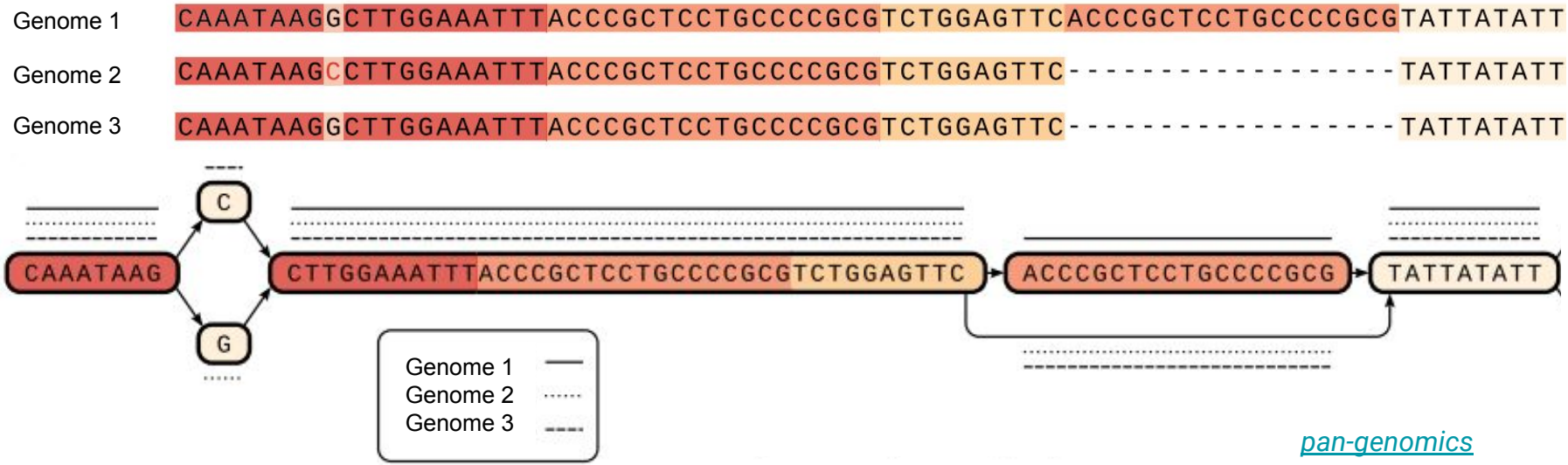
Motivácia — terminológia

- Genóm
  - postupnosť znakov *A, C, G, T*
  - referenčný genóm
- Pangenóm
  - súbor genómových sekvencií používaných spoločne ako referencia
  - prirodzene pridaná diverzita

Motivácia – terminológia

- Pangenóm

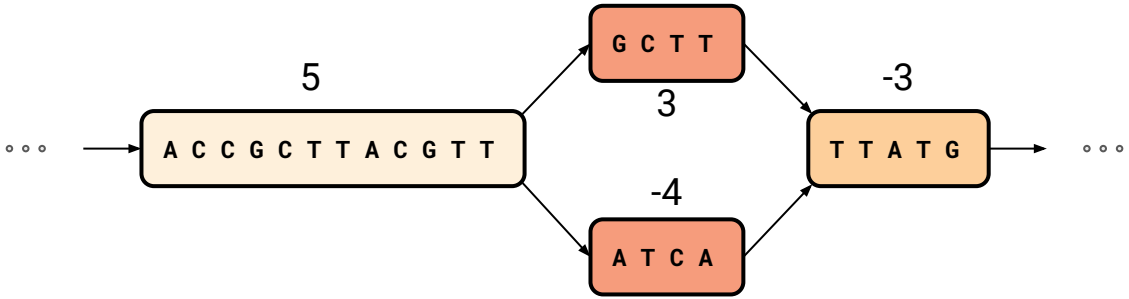
- **graf:** pangenóm (množina genómov)
- **vrchol:** časť sekvencie
- **hrana:** susednosť medzi sekvenciami
  - zlúčenie: identické časti
  - rozdelenie: rozdielne časti



[pan-genomics]

Motivácia

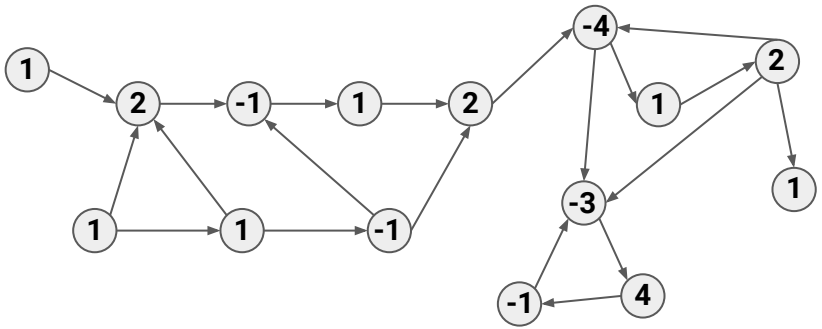
- Pridanie váh vrcholov v grafovom pangenóme
  - označujú významné miesta (mutácie/špecifická biologická funkcia)



- Cieľ:
  - nájsť zhluky, t.j. časti s vysokým skóre

Definícia problému

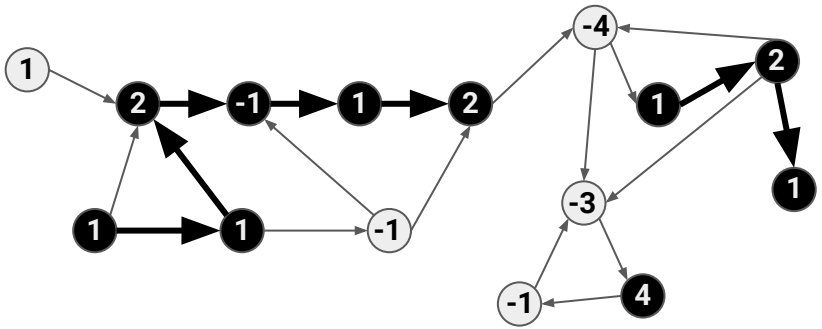
- Chceme nájsť disjunktné cesty v orientovanom grafe, ktoré maximalizujú celkový súčet skór vybraných ciest, pričom výber cesty je penalizovaný



penalty = 3

Definícia problému

- Chceme nájsť disjunktné cesty v orientovanom grafe, ktoré maximalizujú celkový súčet skór vybraných ciest, pričom výber cesty je penalizovaný



Definícia problému

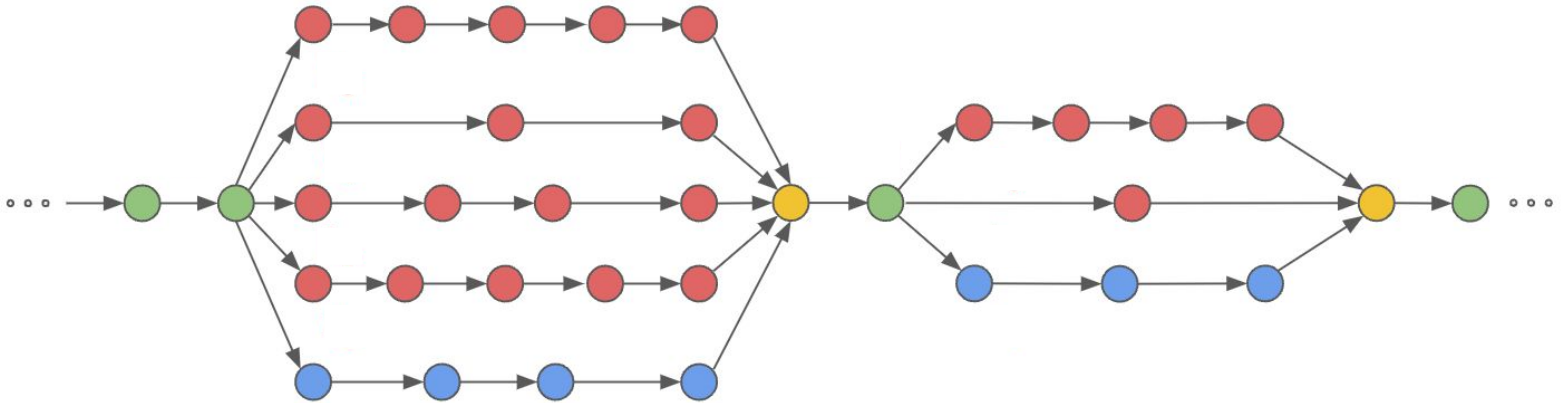
- Tento problém pre všeobecné orientované grafy je NP-ťažký

  ⇒ navrhneme algoritmy pre niektoré triedy orientovaných grafov

Bublinové grafy

≈ elastic-degenerate strings

- ***n*-poschodový bublinový graf** dostaneme tak, že v postupnosti vrcholov $u_1, \dots, u_k$ spojíme každý pár u_i a u_{i+1} s orientovanou cestou alebo *b*-poschodovou bublinou
- ***b*-poschodová bublina** zodpovedá *b* vrcholovo disjunktným cestám medzi zdrojovým vrcholom *s* a cieľovým vrcholom *t*



Bublinové grafy — idea

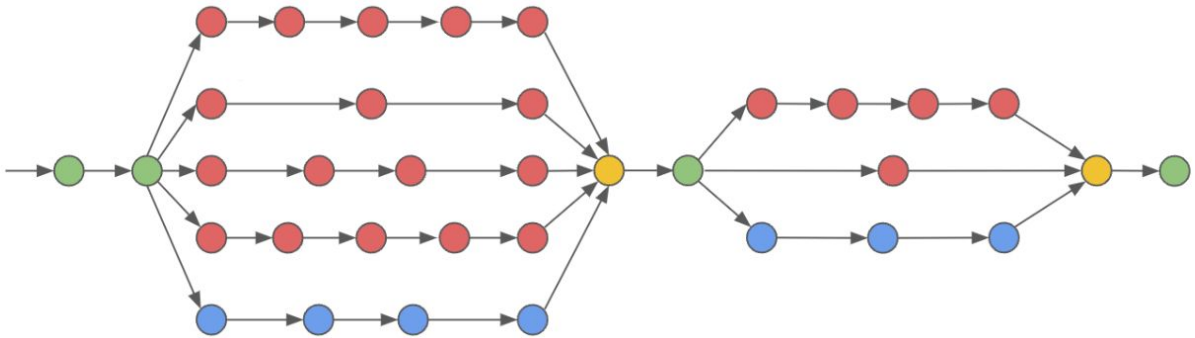
- *M. Csuros: Hľadanie "zhlukov" v lineárnych sekvenciách* (zodpovedá ceste)
  - dynamické programovanie: *W(i, 0/1)* je maximálne skóre pre sekvenciu *[start, i]*,

    kde *i* je/nie je vybraté
    - ***W(i, 0) = max{ W(i - 1, 0), W(i - 1, 1) }***
    - ***W(i, 1) = w_i + max{ W(i - 1, 0) - penalta, W(i - 1, 1) }***
  - zložitosť: *O(n)* *n* je dĺžka sekvencie

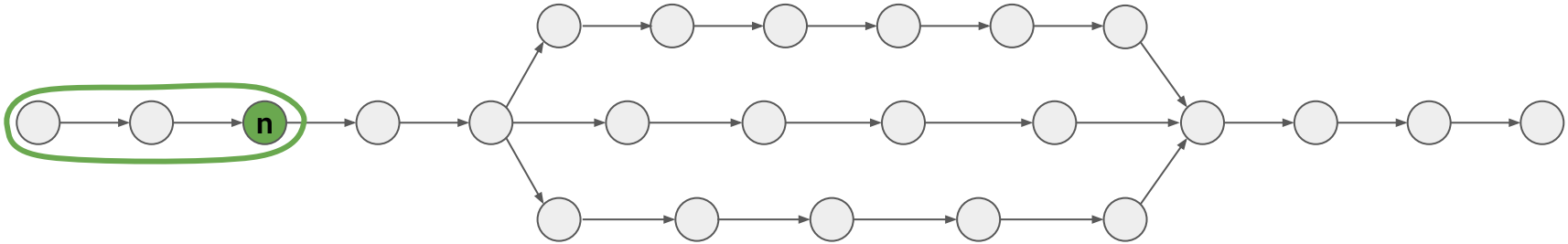
Csuros 2004

Bublinové grafy

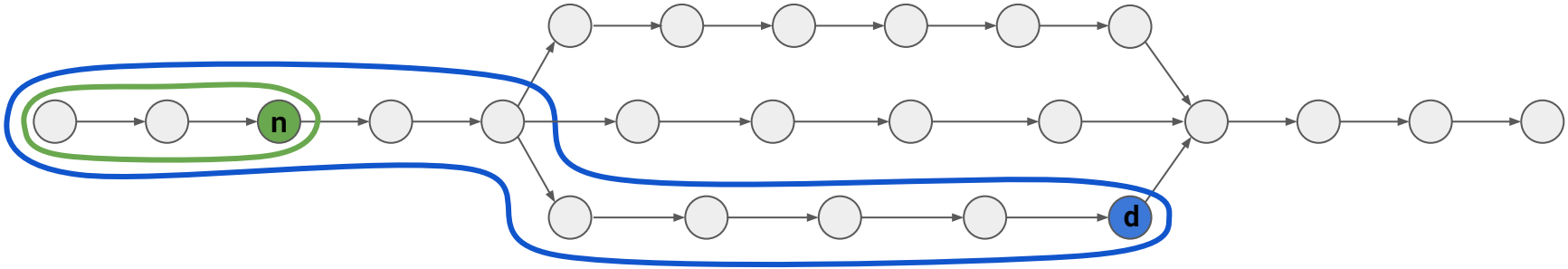
- Navrhneme rozšírenú verziu algoritmu, ktorá beží v $O(V)$ čase, kde V je počet vrcholov



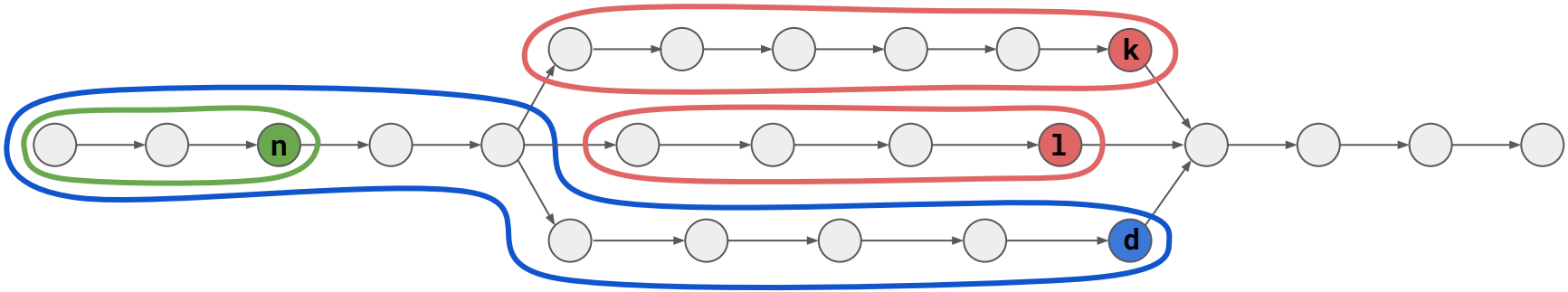
Bublinové grafy



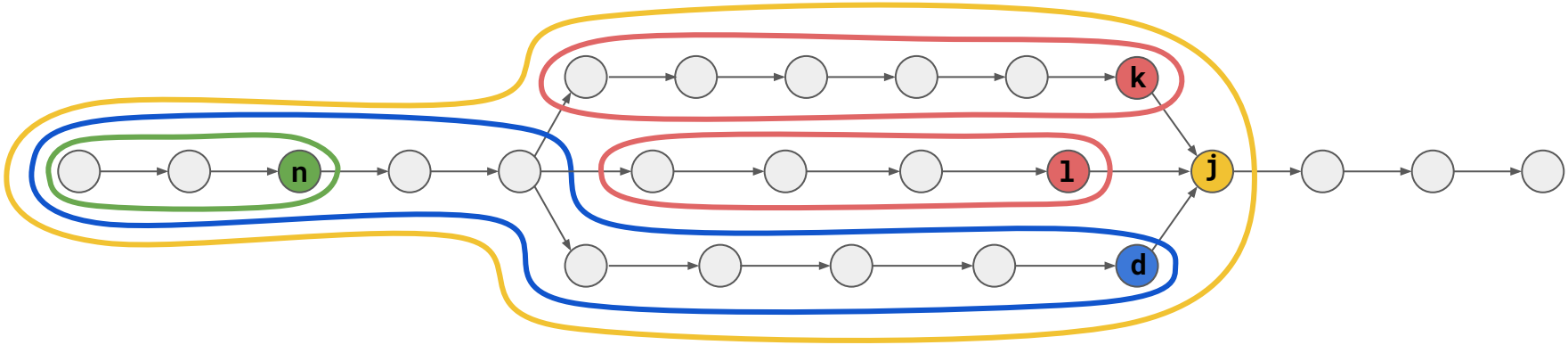
Bublinové grafy



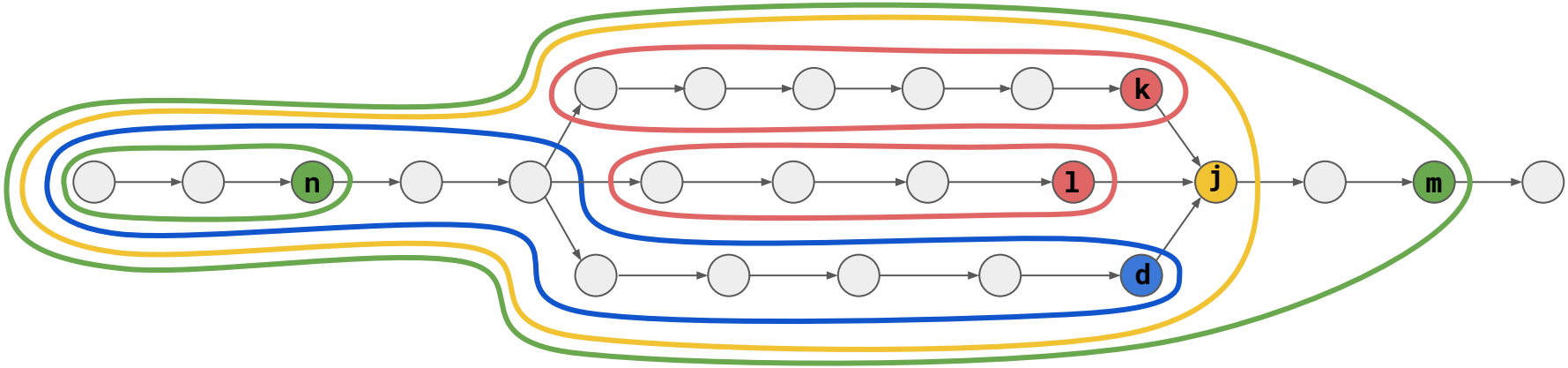
Bublinové grafy



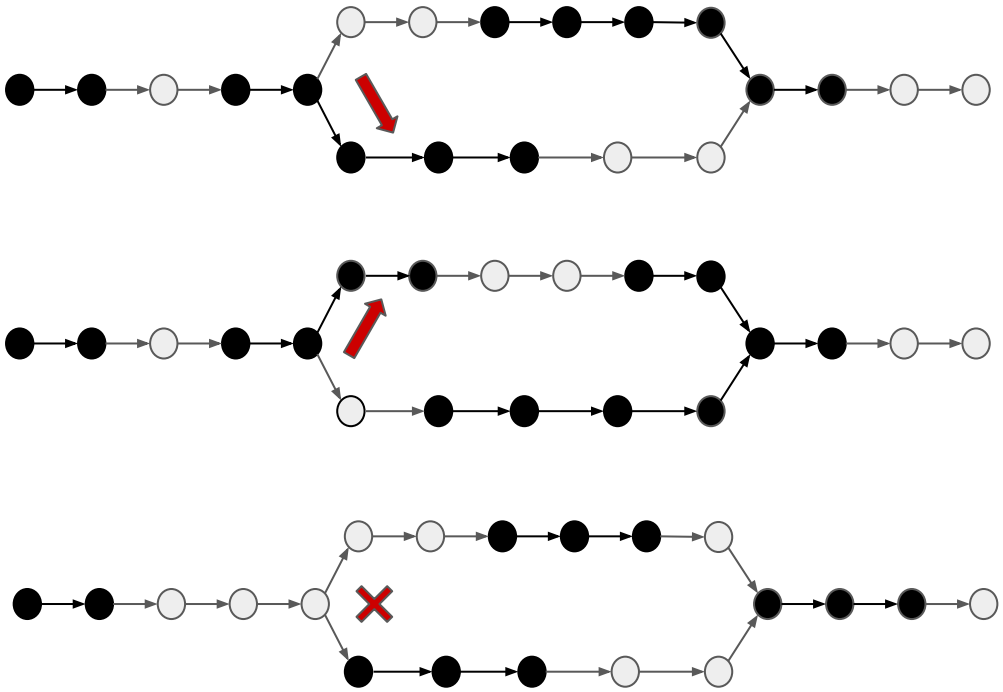
Bublinové grafy



Bublinové grafy



Bublinové grafy

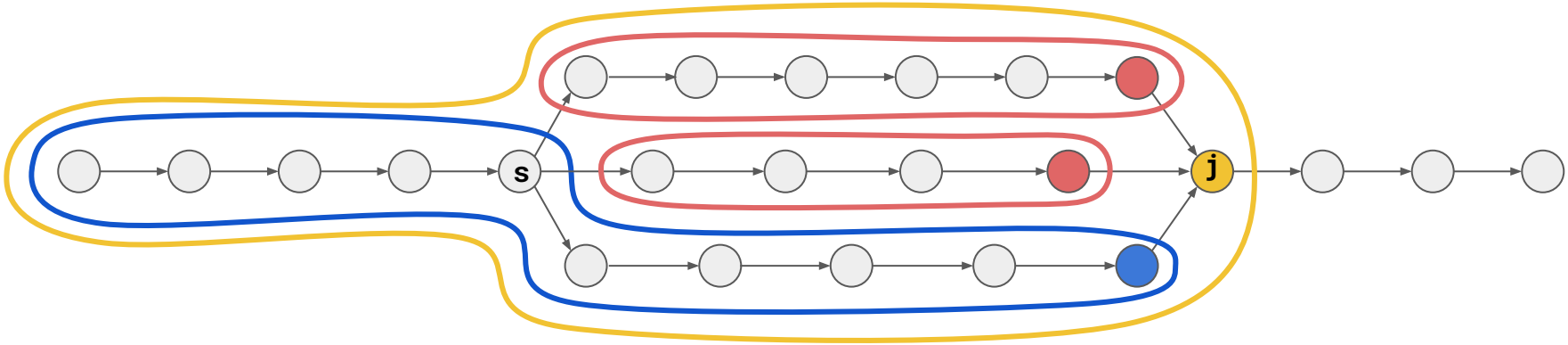


Rozšírená verzia *W(i, 0/1)*:

W(v, s, l):

- vrchol *v*
- *s ∈ {0, 1}* selekčná hodnota:
  - *1:* maximálne skóre, ak *v* je vybraté
  - *0:* maximálne skóre (všeobecne)
- *l ∈ {l, E}* pokračovanie cesty pred bublinou:
  - *l*: pokračuje na danom poschodí
  - *E*: nepokračuje na tomto poschodí

Bublinové grafy



Pravidlo pre vrchol *j*:

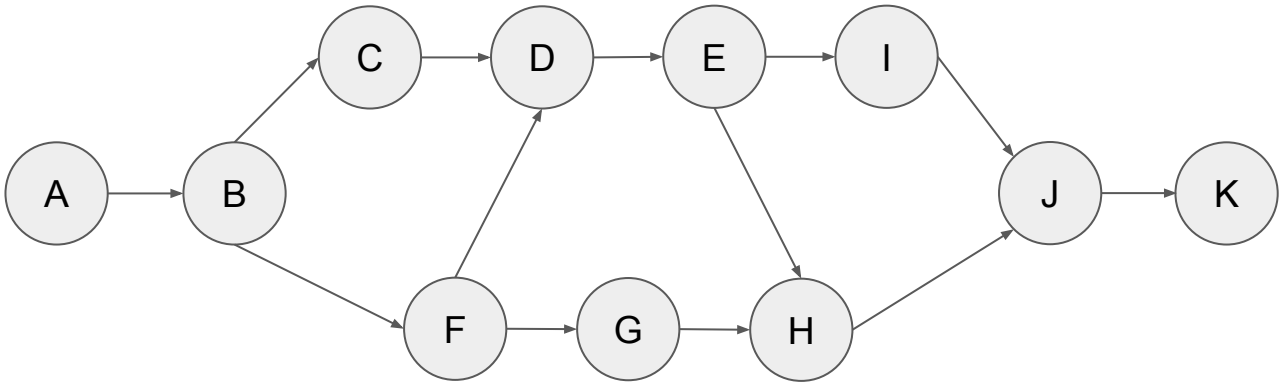
- vyskúšať všetky možné pokračovania cesty z vrchola *s*
- ak *j* je vybraté, môže pokračovať nejakú cestu z priameho predchodcu?

⇒ pri *b*-poschodovej bubline máme $O(b^2)$ kombinácií ciest, ale vieme zrátať v čase $O(b)$

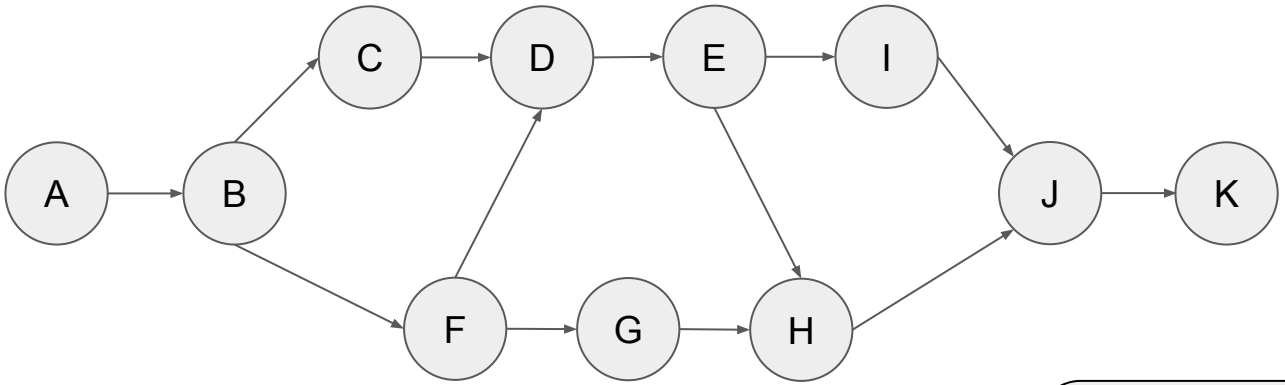
Bublinové grafy

- Algoritmus sme implementovali a použili na hľadanie GC bohatých oblastí v pangenóme baktérie *E. coli*
- Pozreli sme sa, ako počet a dĺžky vybraných ciest sa líšili pri zmene počtu genómov v pangenóme alebo pri zmene penalizačných hodnôt

Komplexnejšie bubliny

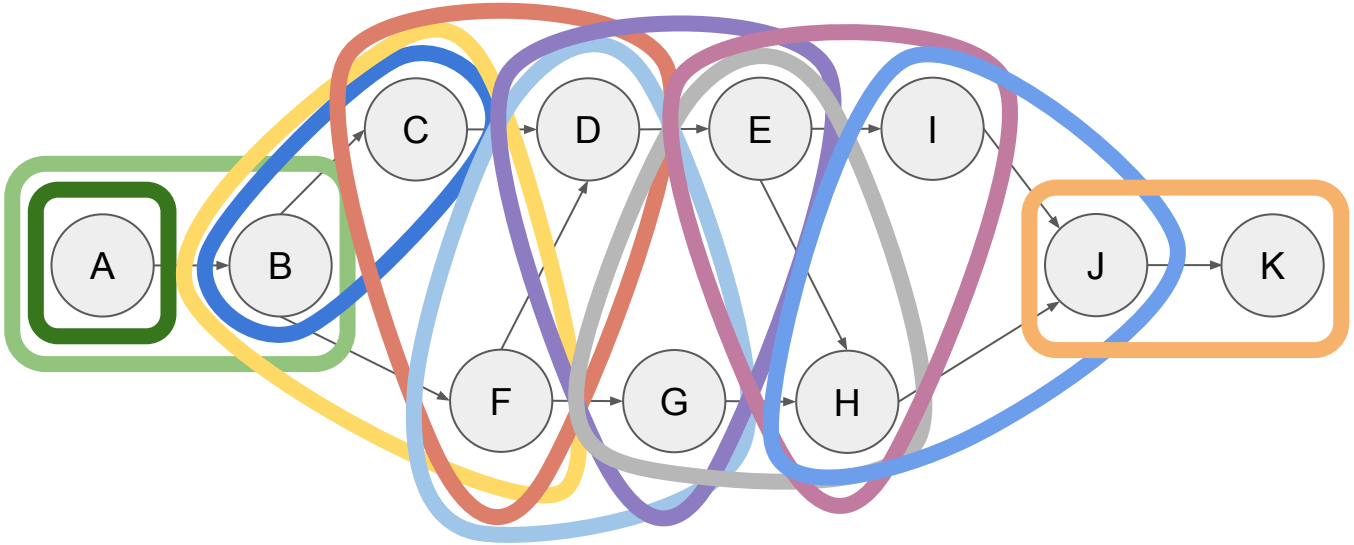


Komplexnejšie bubliny



Spravíme dekompozíciu grafu na postupnosť podmnožín vrcholov

DAGy s ohraničenou *šírkou*



DAGy s ohraničenou *šírkou*

- Algoritmus dostane ako vstup DAG *G* a jeho špeciálnu dekompozíciu (postupnosť množín vrcholov)
- Množiny dekompozície sú spracované po jednom v poradí postupnosti:
  - Algoritmus pre každý vrchol v každej množine vyskúša, či daný vrchol je alebo nie je súčasťou vybranej cesty
  - Ak vrchol je súčasťou vybranej cesty, algoritmus zváži, čo ak vrchol bude alebo nebude koncom nejakej vybranej cesty
  - Podľa skór predošlej množiny vypočíta skóre pre všetky tieto možnosti
- Výsledný algoritmus v čase $O(V \cdot 2^{width} \cdot width)$ nájde cesty v DAGu, ktoré maximalizujú penalizované skóre (width je šírka dekompozície)

Zhrnutie

- Biologický problém hľadania zhlukov s vysokým skóre sme zadefinovali ako problém hľadania disjunktných ciest, ktoré maximalizujú penalizované skóre
- Vytvorili sme algoritmus bežiaci v čase $O(V)$ pre bublinové grafy
  - zodpovedajú elastic-degenerate string-om, ktoré sú používané na reprezentáciu pangenómov
  - algoritmus sme implementovali a spravili sme experimenty na reálnych dátach
- Vytvorili sme algoritmus bežiaci v čase $O(V \cdot 2^{width} \cdot width)$ pre DAGy, kde $width$ je šírka dekompozície

Ďakujem za pozornosť!

Otázky

1. ***Existuje veľa možností ako definovať rôzne príbuzné dekompozície pre orientované grafy. Ako ste sa pri riešení vášho problému dopracovali práve k tej vašej modifikovanej cestovej dekompozícii (def. 2.6.4 v práci)?***

   Naša definícia 2.6.4 bola rozšírením existujúcej definície dekompozície pre neorientované grafy. Naša zmena bola pridanie podmienky *(iii)*, ktorá hovorí, že predchodcovia vrchola sa musia nachádzať v aktuálnej alebo skoršej množine dekompozície.

   Táto definícia umožňuje, aby graf bol spracovaný pomocou dynamického programovania. Dekompozícia sa skladá z postupností množín vrcholov. Podmienky z definície zaistia, aby spracovanie jednej množiny sa dalo spraviť na základe predchádzajúcej množiny.

Definition 2.6.4 (*Modified directed path-decomposition*). Let $G = (V, E)$ be a directed graph. The *modified directed path-decomposition* of G is a sequence of subsets $X_1, ..., X_n$ of V (we refer to them as *bags* of vertices), with three properties:

(i) For each arc of G, there exists an $i \in \{1, ..., n\}$ such that both endpoints of the arc belong to bag X_i.

(ii) For every three indices $1 \leq i \leq j \leq k \leq n$, $X_i \cap X_k \subseteq X_j$.

(iii) If vertex $v \in X_j$ then for each of v's predecessors p exists a bag X_i containing p where $i \leq j$.

Otázky

2. ***Je pre váš algoritmus naozaj potrebná vaša podmienka (i) („pre každú (orientovanú) hranu v grafe existuje aspoň jeden bag obsahujúci oba jej konce“)? Mohla by stačiť aj slabšia verzia tejto podmienky, pri ktorej len požadujeme, aby každý vrchol grafu ležal v aspoň jednom bagu?***

   Slabšia podmienka by umožňovala vytvárať dekompozíciu s jednoprvkovými množinami na základe štandardného topologického usporiadania. Náš algoritmus by na takejto dekompozícii nefungoval, keďže by nevedel spracovať množinu len na základe predchádzajúcej množiny. Takže iba výmena tohto pravidla je nedostatočná, bolo by nutné pridať ešte ďalšie pravidlá alebo zmeniť algoritmus.