Resumo - A New Algorithm for Finding Trees with Many Leaves

Hugo K. K. Rosado

Universidade Estadual de Campinas
MO829 - Algoritmos Parametrizados

29 de novembro de 2016

Conteúdo

Definição

Seja $G = (V, E)$ um grafo direcionado, então um subgrafo acíclico T de G é dito out-tree se T possui uma raiz r tal que existe um caminho direcionado da raiz até cada folha. Dizemos que T é uma out-tree geradora se existe caminho direcionado entre r e cada vértice de G.

MLST

Definição

Seja $G = (V, E)$ um grafo direcionado, então um subgrafo acíclico T de G é dito out-tree se T possui uma raiz r tal que existe um caminho direcionado da raiz até cada folha. Dizemos que T é uma out-tree geradora se existe caminho direcionado entre r e cada vértice de G.

Definição

Maximum Leaf Spanning Tree *(MLST):*

Entrada: Um grafo $G = (V, E)$

Problema: Encontrar árvore T com maior número de folhas.

Caso Geral

- Aplicações práticas em network design.

Caso Geral

- Aplicações práticas em network design.
- O problema é classificado como APX-difícil [8].

Caso Geral

- Aplicações práticas em network design.
- O problema é classificado como APX-difícil [8].
- Alg. 2-aproximação [12].

Caso Geral

- Aplicações práticas em network design.
- O problema é classificado como APX-difícil [8].
- Alg. 2-aproximação [12].
- Alg. 3-aproximação em tempo quase linear [10].

Caso Geral

- Aplicações práticas em network design.
- O problema é classificado como APX-difícil [8].
- Alg. 2-aproximação [12].
- Alg. 3-aproximação em tempo quase linear [10].
- Alg. 3/2-aproximação para grafos cubos [3].

Versão de Parametrização

Definição

Maximum Leaf Spanning Tree *(MLST):*

Entrada: Um grafo $G = (V, E)$ *e* $k \in \mathbb{N}$

Parâmetro: k

Problema: G *contém uma árvore geradora com pelo menos* k *folhas?*

Versão de Parametrização

Definição

Maximum Leaf Spanning Tree *(MLST):*

Entrada: Um grafo $G = (V, E)$ *e* $k \in \mathbb{N}$

Parâmetro: k

Problema: G *contém uma árvore geradora com pelo menos* k *folhas?*

Definição

Directed Maximum Leaf Out-Tree *(DMLOT):*

Entrada: Um digrafo $G = (V, E)$ *e* $k \in \mathbb{N}$

Parâmetro: k

Problema: G *contém uma out-tree com pelo menos* k *folhas?*

Versão de Parametrização

Definição

Directed Maximum Leaf Spanning Out-Tree *(DMLST):*

Entrada: Um grafo $G = (V, E)$ *e* $k \in \mathbb{N}$

Parâmetro: k

Problema: G *contém uma out-tree geradora com pelo menos* k *folhas?*

Versão de Parametrização

Definição

Directed Maximum Leaf Spanning Out-Tree *(DMLST):*

Entrada: Um grafo $G = (V, E)$ *e* $k \in \mathbb{N}$

Parâmetro: k

Problema: G contém uma out-tree geradora com pelo menos k folhas?

- Existe algoritmo exato com tempo $O(1.996^n)$[5].

Versão de Parametrização

Definição

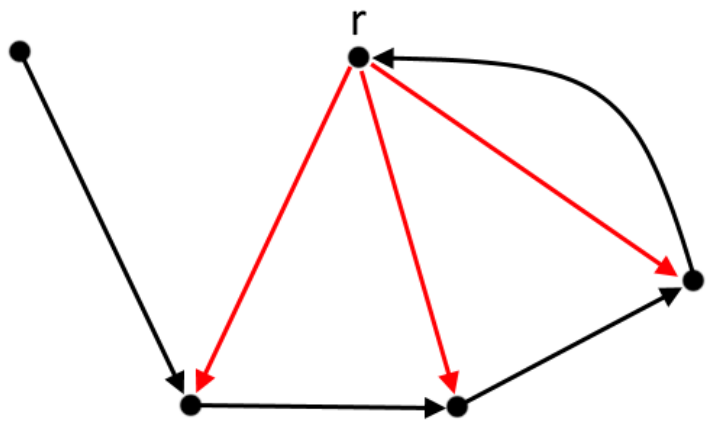
Directed Maximum Leaf Spanning Out-Tree *(DMLST):*

Entrada: Um grafo $G = (V, E)$ *e* $k \in \mathbb{N}$

Parâmetro: k

Problema: G contém uma out-tree geradora com pelo menos k folhas?

- Existe algoritmo exato com tempo $O(1.996^n)$[5].



MLST - Resultados Conhecidos

- MLST $\in$ *FPT* (exercício 6.15 da lista de exercícios 3).

MLST - Resultados Conhecidos

- MLST $\in FPT$ (exercício 6.15 da lista de exercícios 3).
- $O((17k^4)!|G|)$ usando treewidth por Boadlaender [1].

MLST - Resultados Conhecidos

- MLST $\in FPT$ (exercício 6.15 da lista de exercícios 3).
- $O((17k^4)!|G|)$ usando treewidth por Boadlaender [1].
- $O((2k)^{4k} poly(G))$ por Downey e Fellows [6].

MLST - Resultados Conhecidos

- MLST $\in FPT$ (exercício 6.15 da lista de exercícios 3).
- $O((17k^4)!|G|)$ usando treewidth por Boadlaender [1].
- $O((2k)^{4k} poly(G))$ por Downey e Fellows [6].
- $O(|G| + 14.23^k k)$ por Fellows, McCartin, Rosamond e Stege usando resultados de teoria de grafos muito sofisticados [11].

MLST - Resultados Conhecidos

- MLST $\in FPT$ (exercício 6.15 da lista de exercícios 3).
- $O((17k^4)!|G|)$ usando treewidth por Boadlaender [1].
- $O((2k)^{4k} poly(G))$ por Downey e Fellows [6].
- $O(|G| + 14.23^k k)$ por Fellows, McCartin, Rosamond e Stege usando resultados de teoria de grafos muito sofisticados [11].
- $O(|V|^3 + 9.4815^k k^3)$ por Bonsma, Brueggermann e Woeginger [2].

MLST - Resultados Conhecidos

- MLST $\in FPT$ (exercício 6.15 da lista de exercícios 3).
- $O((17k^4)!|G|)$ usando treewidth por Boadlaender [1].
- $O((2k)^{4k} poly(G))$ por Downey e Fellows [6].
- $O(|G| + 14.23^k k)$ por Fellows, McCartin, Rosamond e Stege usando resultados de teoria de grafos muito sofisticados [11].
- $O(|V|^3 + 9.4815^k k^3)$ por Bonsma, Brueggermann e Woeginger [2].
- Kernel de tamanho $3.75k$ por Estivill-Castro [7] - o que diretamente reduz a complexidade do algoritmo de Bonsma para $O(|V|^3 + 8.12^k)$.

MLST - Resultados Conhecidos

- MLST $\in FPT$ (exercício 6.15 da lista de exercícios 3).
- $O((17k^4)!|G|)$ usando treewidth por Boadlaender [1].
- $O((2k)^{4k} poly(G))$ por Downey e Fellows [6].
- $O(|G| + 14.23^k k)$ por Fellows, McCartin, Rosamond e Stege usando resultados de teoria de grafos muito sofisticados [11].
- $O(|V|^3 + 9.4815^k k^3)$ por Bonsma, Brueggermann e Woeginger [2].
- Kernel de tamanho $3.75k$ por Estivill-Castro [7] - o que diretamente reduz a complexidade do algoritmo de Bonsma para $O(|V|^3 + 8.12^k)$.
- $O(poly(|V|) + 6.75^k poly(k))$ por Bonsma e Zickfield [4].

MLST - Resultados Conhecidos

- MLST $\in FPT$ (exercício 6.15 da lista de exercícios 3).
- $O((17k^4)!|G|)$ usando treewidth por Boadlaender [1].
- $O((2k)^{4k} poly(G))$ por Downey e Fellows [6].
- $O(|G| + 14.23^k k)$ por Fellows, McCartin, Rosamond e Stege usando resultados de teoria de grafos muito sofisticados [11].
- $O(|V|^3 + 9.4815^k k^3)$ por Bonsma, Brueggermann e Woeginger [2].
- Kernel de tamanho $3.75k$ por Estivill-Castro [7] - o que diretamente reduz a complexidade do algoritmo de Bonsma para $O(|V|^3 + 8.12^k)$.
- $O(poly(|V|) + 6.75^k poly(k))$ por Bonsma e Zickfield [4].
- $O(poly(|V|) + 4^k k^2)$ por Kneis, Langer e Rossmanith [9].

MLST - Resultados Conhecidos

- MLST $\in FPT$ (exercício 6.15 da lista de exercícios 3).
- $O((17k^4)!|G|)$ usando treewidth por Boadlaender [1].
- $O((2k)^{4k} poly(G))$ por Downey e Fellows [6].
- $O(|G| + 14.23^k k)$ por Fellows, McCartin, Rosamond e Stege usando resultados de teoria de grafos muito sofisticados [11].
- $O(|V|^3 + 9.4815^k k^3)$ por Bonsma, Brueggermann e Woeginger [2].
- Kernel de tamanho $3.75k$ por Estivill-Castro [7] - o que diretamente reduz a complexidade do algoritmo de Bonsma para $O(|V|^3 + 8.12^k)$.
- $O(poly(|V|) + 6.75^k poly(k))$ por Bonsma e Zickfield [4].
- $O(poly(|V|) + 4^k k^2)$ por Kneis, Langer e Rossmanith [9].
- $O(poly(|V|) + 3.72^k k^O(1))$ por Daligult, Gutin, Kim e Yeo [5].

DMLOT e DMLST - Resultados Conhecidos

- DMLOT e DMLST $\in$ *FPT*???

DMLOT e DMLST - Resultados Conhecidos

- DMLOT e DMLST $\in FPT$???
- $O(2^{O(klog(k))}poly(|V|))$ por Bonsma e Dorn utilizando técnicas de pathwidth [2].

DMLOT e DMLST - Resultados Conhecidos

- DMLOT e DMLST $\in FPT$???
- $O(2^{O(klog(k))}poly(|V|))$ por Bonsma e Dorn utilizando técnicas de pathwidth [2].
- $O(4^k|V||E|)$ por Kneis, Langer e Rossmanith [9].

DMLOT e DMLST - Resultados Conhecidos

- DMLOT e DMLST $\in FPT$???
- $O(2^{O(klog(k))}poly(|V|))$ por Bonsma e Dorn utilizando técnicas de pathwidth [2].
- $O(4^k|V||E|)$ por Kneis, Langer e Rossmanith [9].
- $O(poly(|V|) + 3.72^k|V|^O(1))$ por Daligult, Gutin, Kim e Yeo [5].

Definições

Definição

Denotamos também por $N_{\overline{T}}(v) := N(v) - V(T)$ *os vizinhos de* v *que não estão em* T *e por* $T_v := (N[v], \cup_{u \in N(v)}\{(v,u)\})$ *a estrela de centro* v *e seus vizinhos.*

Definição

Se T *é uma árvore com pelo menos* k *folhas, chamamos* T *de árvore de* k*-folhas.*

Definições

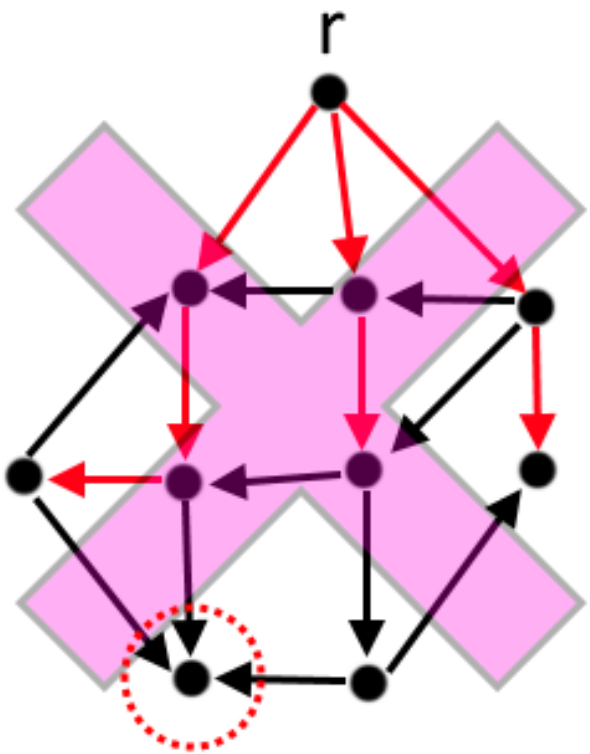
Definição

Se $N(inner(T)) \subseteq V(T)$, então dizemos que T é uma árvore inner-maximal.

Definições

Definição

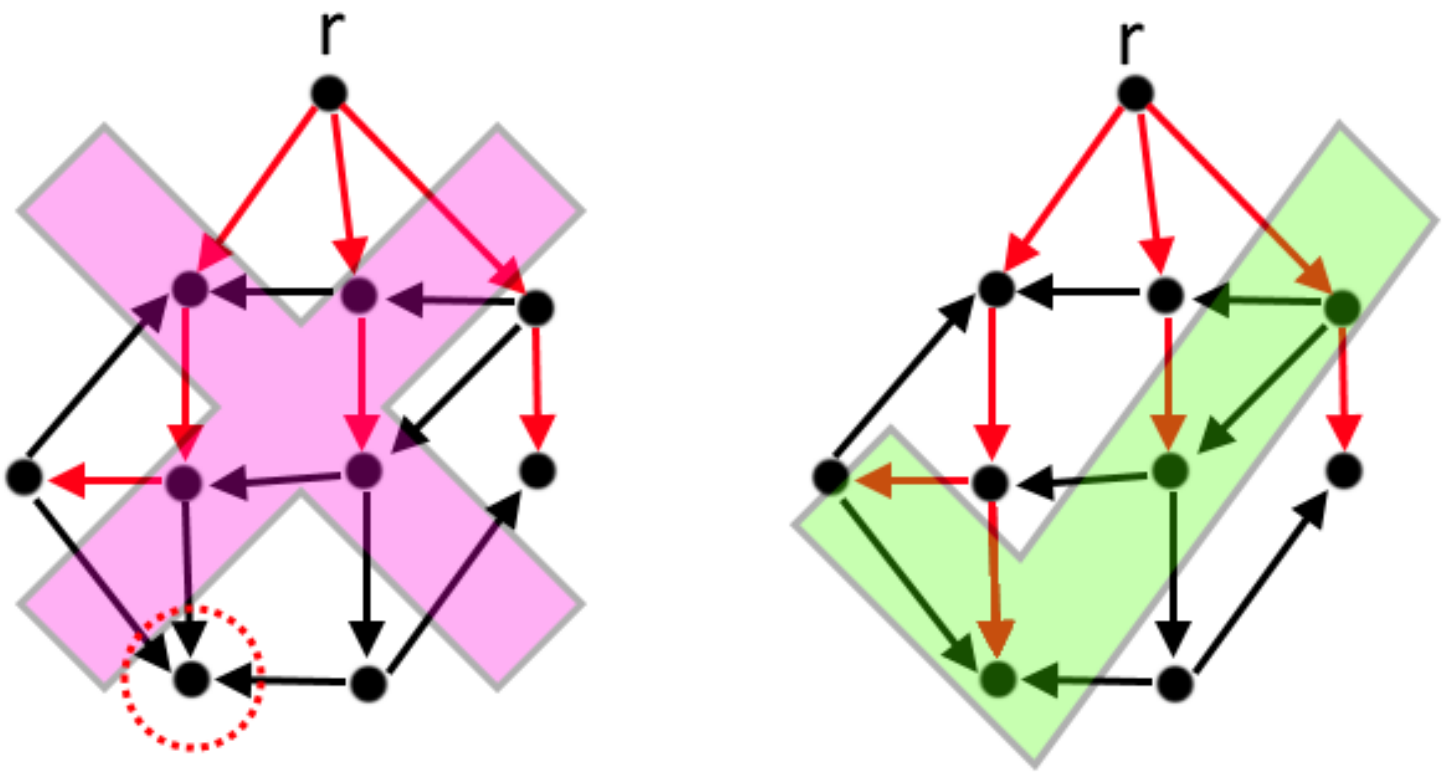
Se $N(inner(T)) \subseteq V(T)$, então dizemos que T é uma árvore inner-maximal.



Definições

Definição

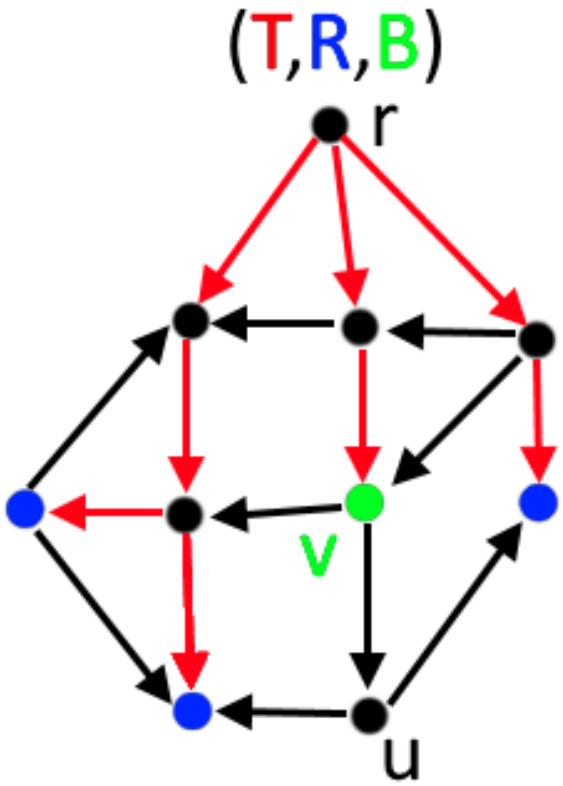
Se $N(inner(T)) \subseteq V(T)$, então dizemos que T é uma árvore inner-maximal.



Definições

Definição

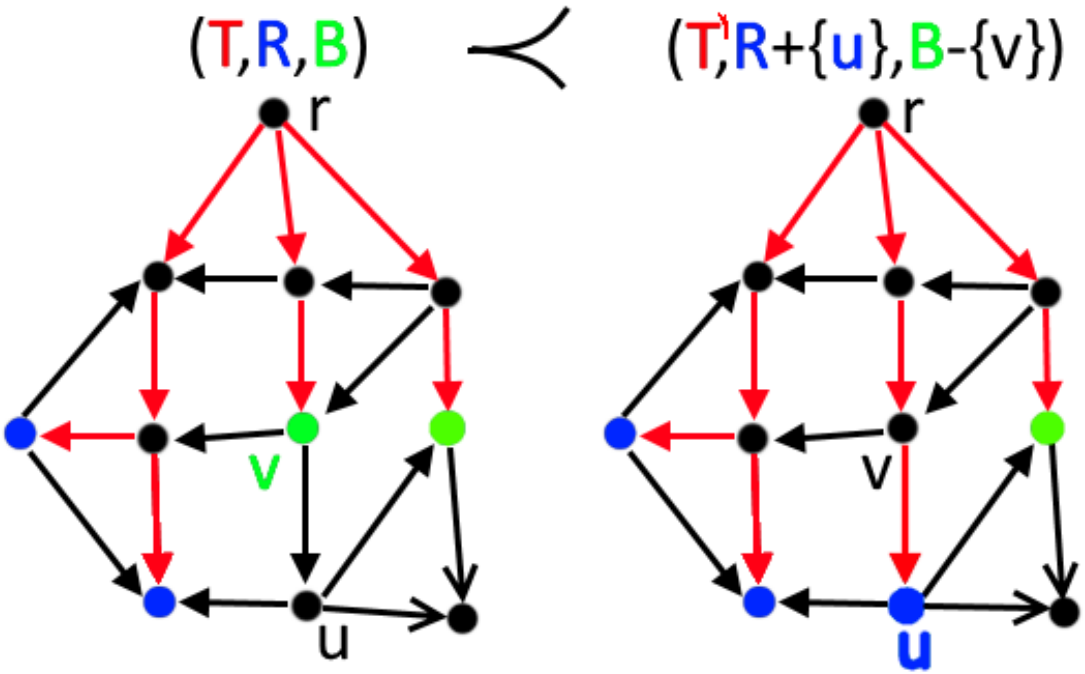
Uma árvore rotulada é um 3-tupla (T, R, B) *tal que* T *é uma árvore, e* R *e* B *particionam leaves*(T).



Definições

Definição

Dizemos que uma árvore $T' \neq T$ estende T, denotando por $T' \succ T$ sss $root(T) = root(T')$ e T for um grafo induzido de T'. Uma árvore rotulada (T, R, B) é (folha-preservativa) estendida por T' se $T' \succ T$, $R \subseteq leaves(T')$ e denotamos por $T' \succ (T, R, B)$. Também dizemos que uma árvore rotulada (T', R', B') estende (T, R, B) sss $T' \succ (T, R, B)$.



Lemas

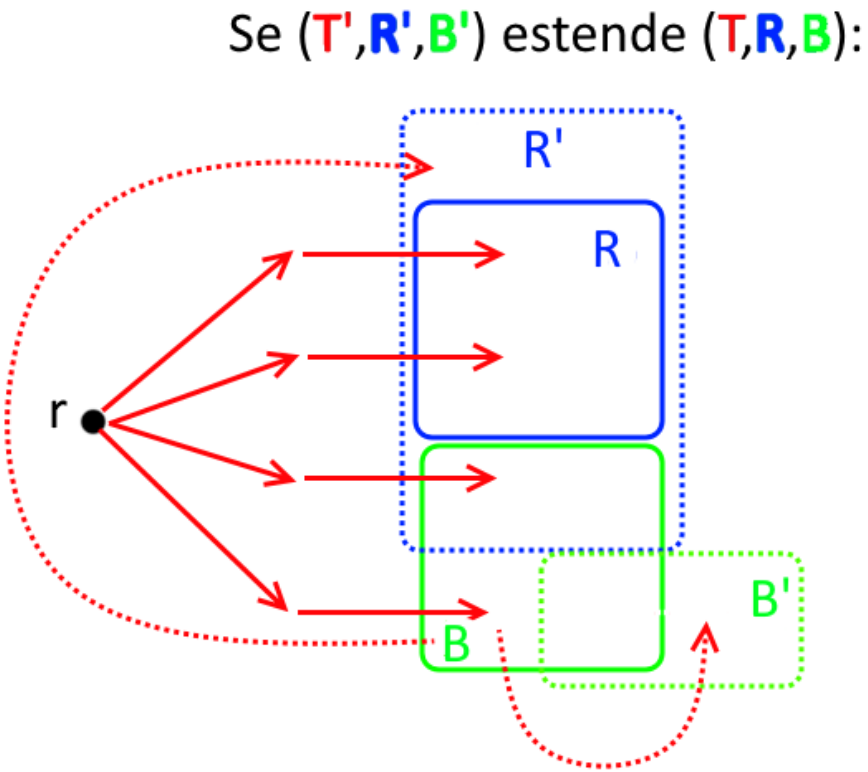
Lema

Lema 1: *Seja* (T, R, B) *uma árvore rotulada inner-maximal e seja* $T' \succ (T, R, B)$, *então* $B \neq \emptyset$.

Lemas

Lema

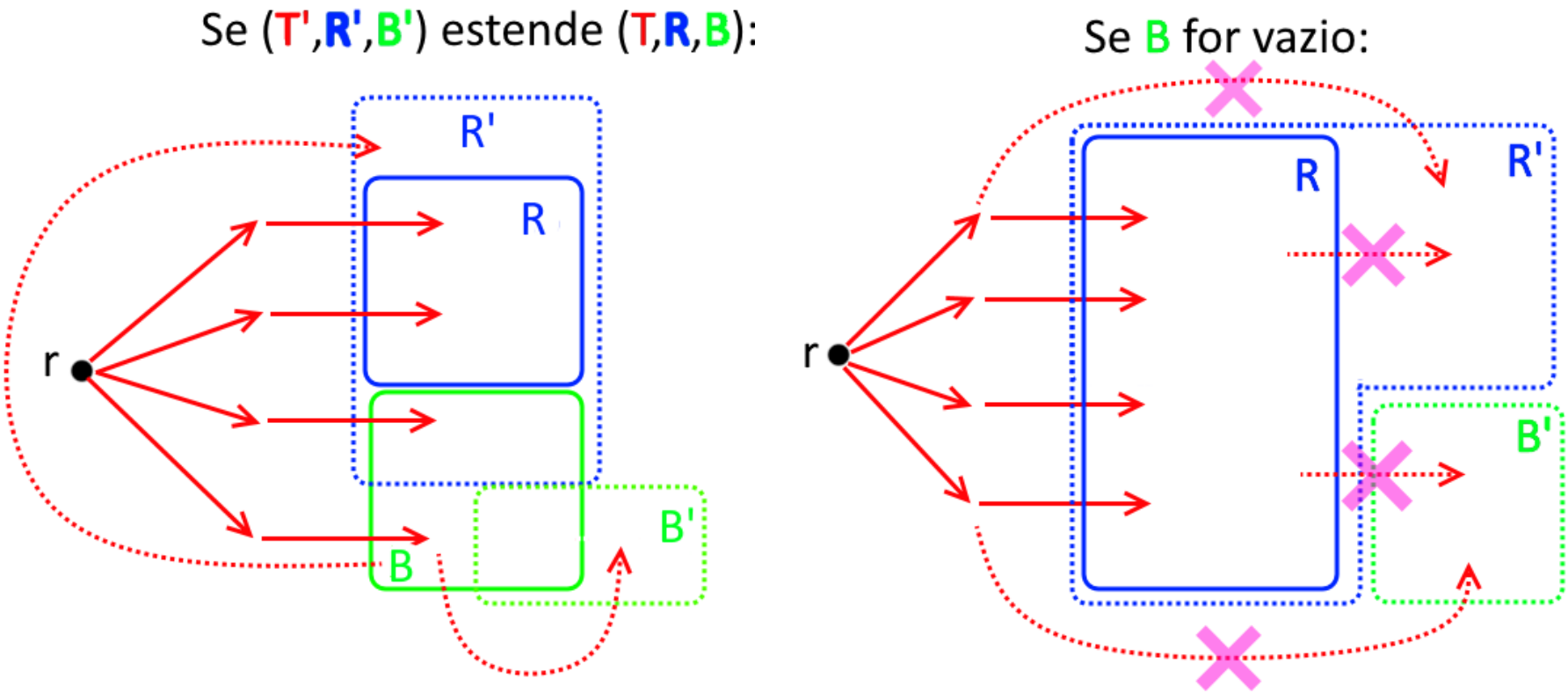
Lema 1: *Seja* (T, R, B) *uma árvore rotulada inner-maximal e seja* $T' \succ (T, R, B)$, *então* $B \neq \emptyset$.



Lemas

Lema

Lema 1: *Seja* (T, R, B) *uma árvore rotulada inner-maximal e seja* $T' \succ (T, R, B)$, *então* $B \neq \emptyset$.



Lemas

Lema

Lema 2: *Seja $G = (V, E)$ um grafo conexo não-direcionado, então G contém uma árvore de k-folhas sss G contém uma árvore de k-folhas geradora com k folhas.*

Lemas

Lema

Lema 2: *Seja $G = (V, E)$ um grafo conexo não-direcionado, então G contém uma árvore de k-folhas sss G contém uma árvore de k-folhas geradora com k folhas.*

Se |V-V(T)|=0:

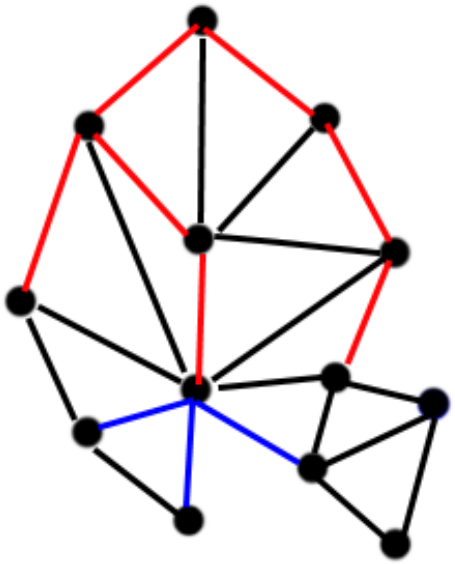
Lemas

Lema

Lema 2: *Seja $G = (V, E)$ um grafo conexo não-direcionado, então G contém uma árvore de k-folhas sss G contém uma árvore de k-folhas geradora com k folhas.*

Se |V-V(T)|=0:

Se |V-V(T)|>0, então BFS:



Lemas

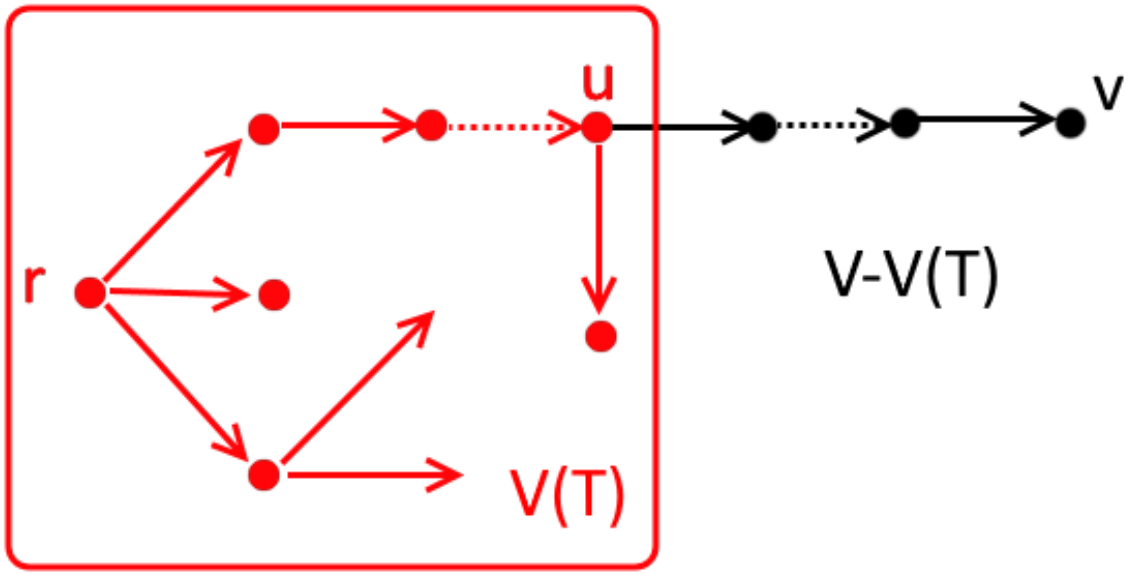
Lema

Lema 3: *Seja $G = (V, E)$ um grafo direcionado. Se G contém uma out-tree de k-folhas geradora enraizada em v, então qualquer out-tree de k-folhas enraizada em v pode ser expandida para uma out-tree de k-folhas geradora de G.*

Lemas

Lema

Lema 3: *Seja $G = (V, E)$ um grafo direcionado. Se G contém uma out-tree de k-folhas geradora enraizada em v, então qualquer out-tree de k-folhas enraizada em v pode ser expandida para uma out-tree de k-folhas geradora de G.*



Lemas

Lema

Lema 4: *Seja* $G = (V, E)$ *um grafo,* (T, R, B) *uma árvore rotulada e* $x \in B$. *Então:*

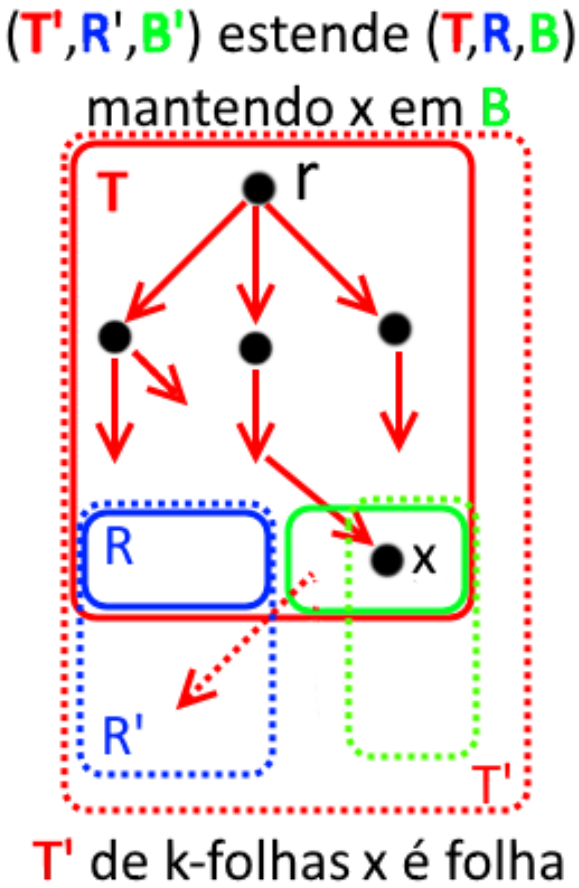
1) *Se não há árvore de k-folhas* T', *tal que,* $T' \succeq (T, R \cup \{x\}, B - \{x\})$, *então toda árvore de k-folhas* T' *que estende* (T, R, B) *tem que* $x \in inner(T')$.
2) *Se existe árvore de k-folhas* T', *tal que* $T' \succeq (T, R, B)$ *e* $x \in inner(T')$, *então existe árvore de k-folhas* $T'' \succeq (T + \{(x, y) : y \in N_{\overline{T}}(x)\}, R, N_{\overline{T}}(x) \cup B - \{x\})$.

Lemas

1) Se não há árvore de k-folhas T', tal que, $T' \succeq (T, R \cup \{x\}, B - \{x\})$, então toda árvore de k-folhas T' que estende (T, R, B) tem que $x \in \mathit{inner}(T')$.

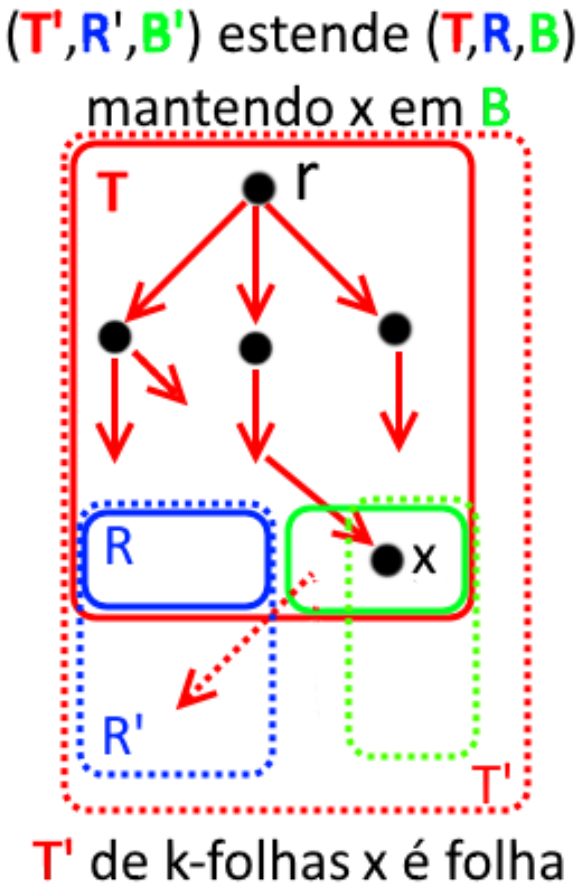
Lemas

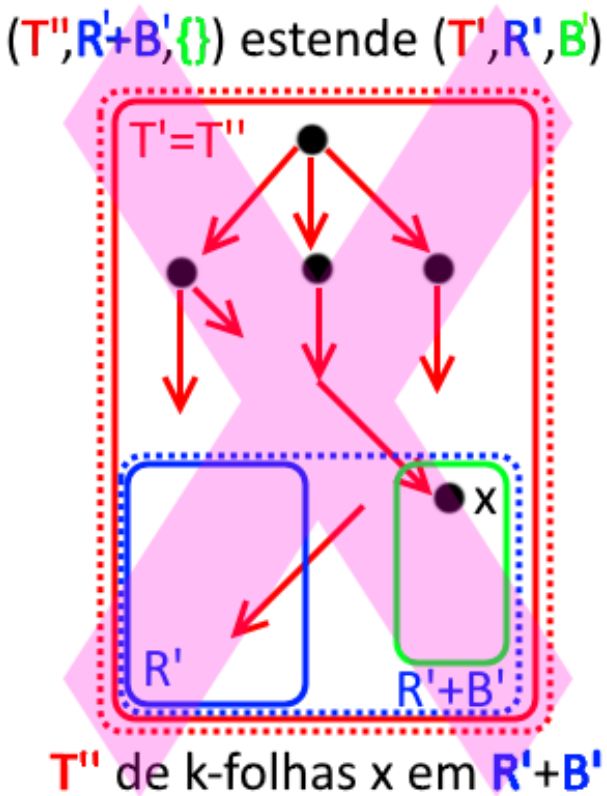
1) Se não há árvore de k-folhas T', tal que, $T' \succeq (T, R \cup \{x\}, B - \{x\})$, então toda árvore de k-folhas T' que estende (T, R, B) tem que $x \in \mathit{inner}(T')$.



Lemas

1) Se não há árvore de k-folhas T', tal que, $T' \succeq (T, R \cup \{x\}, B - \{x\})$, então toda árvore de k-folhas T' que estende (T, R, B) tem que $x \in \mathit{inner}(T')$.





Lemas

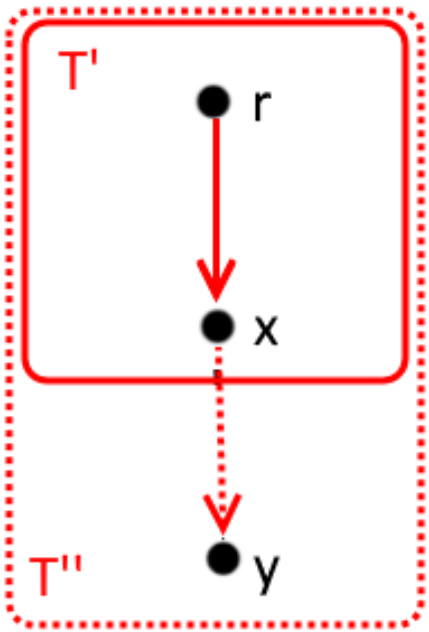
2) Se existe árvore de k-folhas T', tal que $T' \succeq (T, R, B)$ e $x \in \mathit{inner}(T')$, então existe árvore de k-folhas $T'' \succeq (T + \{(x, y) : y \in N_{\overline{T}}(x)\}, R, N_{\overline{T}}(x) \cup B - \{x\})$.

Como $N_{\overline{T}}(x) \neq \emptyset$, existe $y \in N_{\overline{T}}(x)$:

Lemas

2) Se existe árvore de k-folhas T', tal que $T' \succeq (T, R, B)$ e $x \in \mathit{inner}(T')$, então existe árvore de k-folhas $T'' \succeq (T + \{(x, y) : y \in N_{\overline{T}}(x)\}, R, N_{\overline{T}}(x) \cup B - \{x\})$.

Como $N_{\overline{T}}(x) \neq \emptyset$, existe $y \in N_{\overline{T}}(x)$:

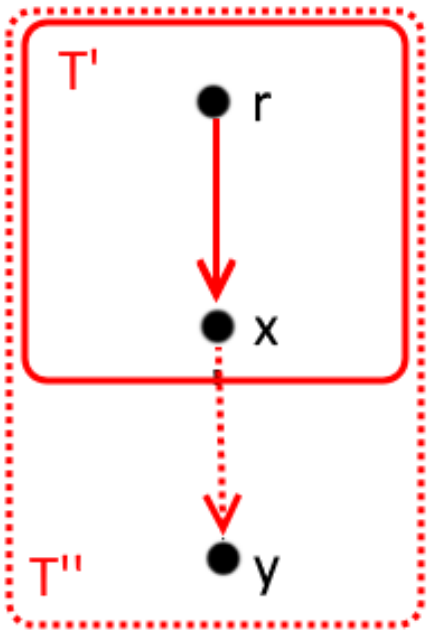


(a) $y \notin V(T')$

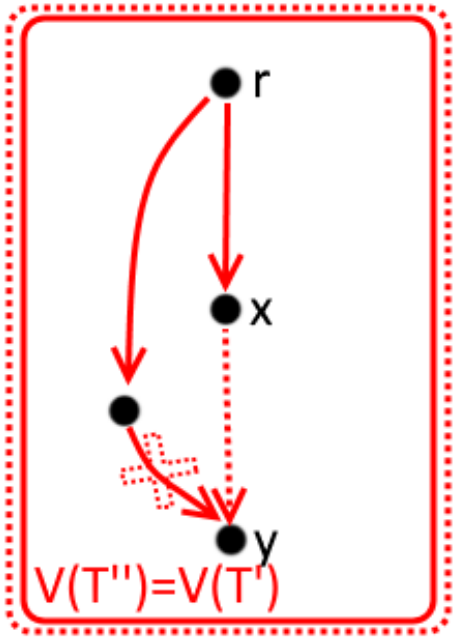
Lemas

2) Se existe árvore de k-folhas T', tal que $T' \succeq (T, R, B)$ e $x \in \mathit{inner}(T')$, então existe árvore de k-folhas $T'' \succeq (T + \{(x, y) : y \in N_{\overline{T}}(x)\}, R, N_{\overline{T}}(x) \cup B - \{x\})$.

Como $N_{\overline{T}}(x) \neq \emptyset$, existe $y \in N_{\overline{T}}(x)$:



(a) $y \notin V(T')$



(b) $y \in V(T')$ mas $(x, y) \notin E(T')$

Lemas

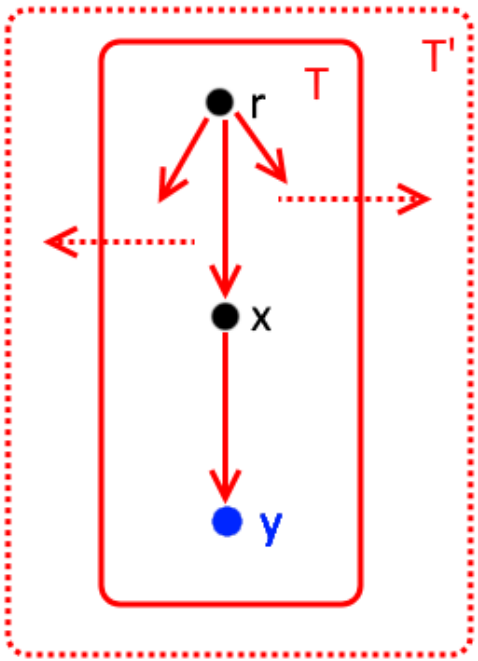
Lema

Lema 5: *Seja $G = (V, E)$ um grafo, (T, R, B) uma árvore de k-folhas e $x \in B$ tal que $N_{\overline{T}}(x) = \{y\}$. Se não existe árvore de k-folhas que estenda $(T, R \cup \{x\}, B - \{x\})$, então não existe árvore de k-folhas que estenda $(T + (x, y), R \cup \{y\}, B - \{x\})$.*

Lemas

Lema

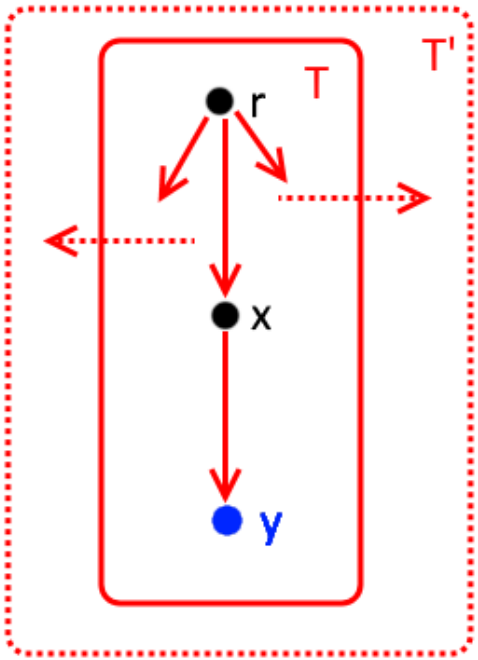
Lema 5: *Seja $G = (V, E)$ um grafo, (T, R, B) uma árvore de k-folhas e $x \in B$ tal que $N_{\overline{T}}(x) = \{y\}$. Se não existe árvore de k-folhas que estenda $(T, R \cup \{x\}, B - \{x\})$, então não existe árvore de k-folhas que estenda $(T + (x, y), R \cup \{y\}, B - \{x\})$.*

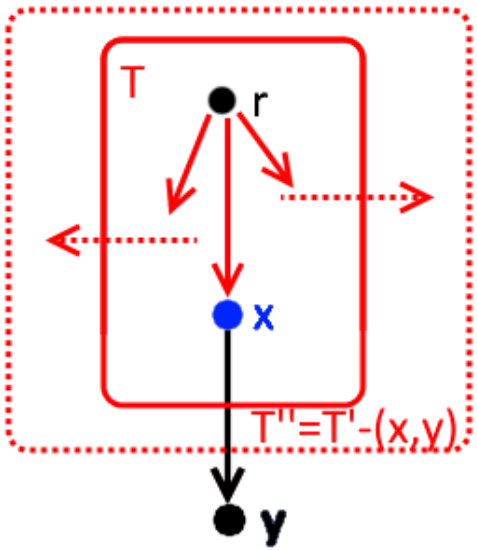


Lemas

Lema

Lema 5: *Seja $G = (V, E)$ um grafo, (T, R, B) uma árvore de k-folhas e $x \in B$ tal que $N_{\overline{T}}(x) = \{y\}$. Se não existe árvore de k-folhas que estenda $(T, R \cup \{x\}, B - \{x\})$, então não existe árvore de k-folhas que estenda $(T + (x, y), R \cup \{y\}, B - \{x\})$.*





T'e T'' são de k-folhas

Lemas

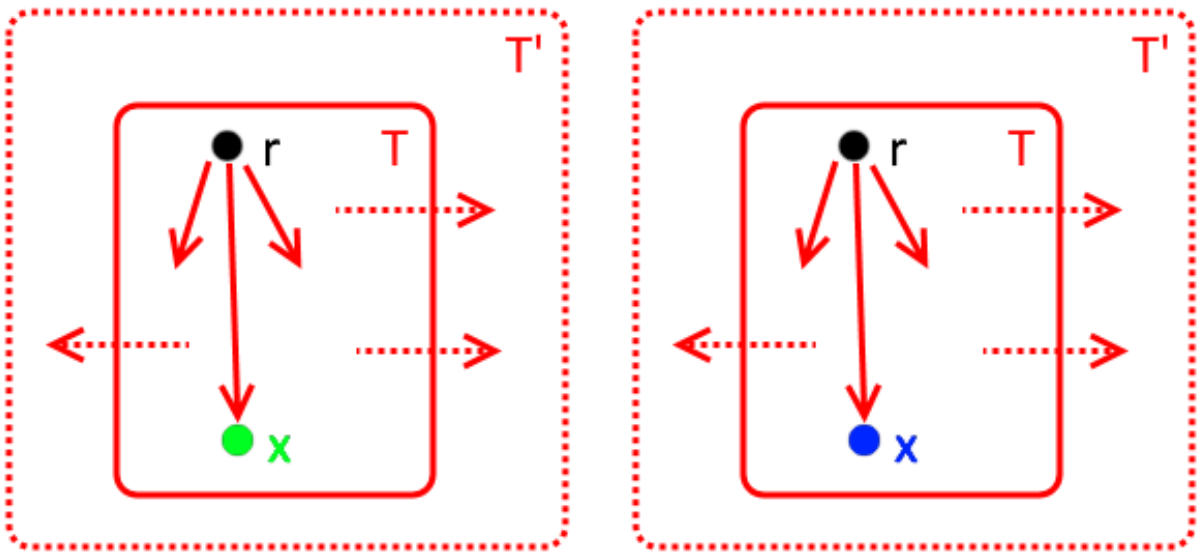
Corolário

Corolário 1: *Seja* $G = (V, E)$ *um grafo,* (T, R, B) *uma árvore de k-folhas e* $x \in B$. *Se* $N_{\overline{T}}(x) = \emptyset$ *e se existe uma árvore de k-folhas que estenda* (T, R, B), *então existe árvore de k-folhas que estende* $(T, R \cup \{x\}, B - \{x\})$.

Lemas

Corolário

Corolário 1: *Seja $G = (V, E)$ um grafo, (T, R, B) uma árvore de k-folhas e $x \in B$. Se $N_{\overline{T}}(x) = \emptyset$ e se existe uma árvore de k-folhas que estenda (T, R, B), então existe árvore de k-folhas que estende $(T, R \cup \{x\}, B - \{x\})$.*



T' de k-folhas estende tanto (T,R,B)
quanto (T,R+{x},B-{x})

Algoritmo MaxLeaf

Algorithm 1 MaxLeaf(G, T, R, B, k)

Entrada: Um grafo G, uma árvore inner-maximal (T, R, B), $k \in \mathbb{N}$
Saída: Existe árvore de k-folhas $T' \succ (T, R, B)$?

1: **if** $|R| + |B| \geq k$ **then return** "SIM"
2: **if** $B = \emptyset$ **then return** "NAO"
3: Escolha um $u \in B$
//Tentar branch em que u é folha
4: **if** MaxLeaf$(G, T, R \cup \{u\}, B - \{u\}, k)$ retorna "SIM" **then return** "SIM"
//Se u não é folha, u deve ser interno em todas as soluções
5: $B \leftarrow B - \{u\}$
6: $N \leftarrow N_{\overline{T}}(u)$
7: $T \leftarrow T \cup \{(u, u') : u' \in N\}$
//Crescer caminho único: Lema 3.5
8: **while** $|N| = 1$ **do**
9: Seja $N = \{v\}$
10: $N \leftarrow N_{\overline{T}}(v)$
11: $T \leftarrow T \cup \{(v, v') : v' \in N\}$
//Não crie branch se não há mais vizinhos: Corolário 3.6
12: **if** $N = \emptyset$ **then return** "NAO"
13: **return** MaxLeaf$(G, T, R, B \cup N, k)$

Lema

Lema 6: *Seja $G = (V, E)$ um grafo e $k > 2$. Se G não contém uma árvore de k-folhas,* $\textsc{MaxLeaf}(G, T, R, B, k)$ *retorna "NAO"para cada $v \in V$. Se G contém tal árvore enraizada em r, então* $\textsc{MaxLeaf}(G, T_r, \emptyset, N(r), k)$ *retorna "SIM".*

MaxLeaf - Corretude

1) Seja (T, R, B) inner maximal e $u \in B$. Então todas as chamadas de MaxLeaf têm como argumento uma árvore inner-maximal:

MaxLeaf - Corretude

1) Seja (T, R, B) inner maximal e $u \in B$. Então todas as chamadas de MaxLeaf têm como argumento uma árvore inner-maximal:

A: $(T_r, \emptyset, N(r))$ é inner-maximal.

MaxLeaf - Corretude

1) Seja (T, R, B) inner maximal e $u \in B$. Então todas as chamadas de MaxLeaf têm como argumento uma árvore inner-maximal:

A: $(T_r, \emptyset, N(r))$ é inner-maximal.

B: $(T, R \cup \{u\}, B - \{u\})$ da Linha 4 é inner-maximal.

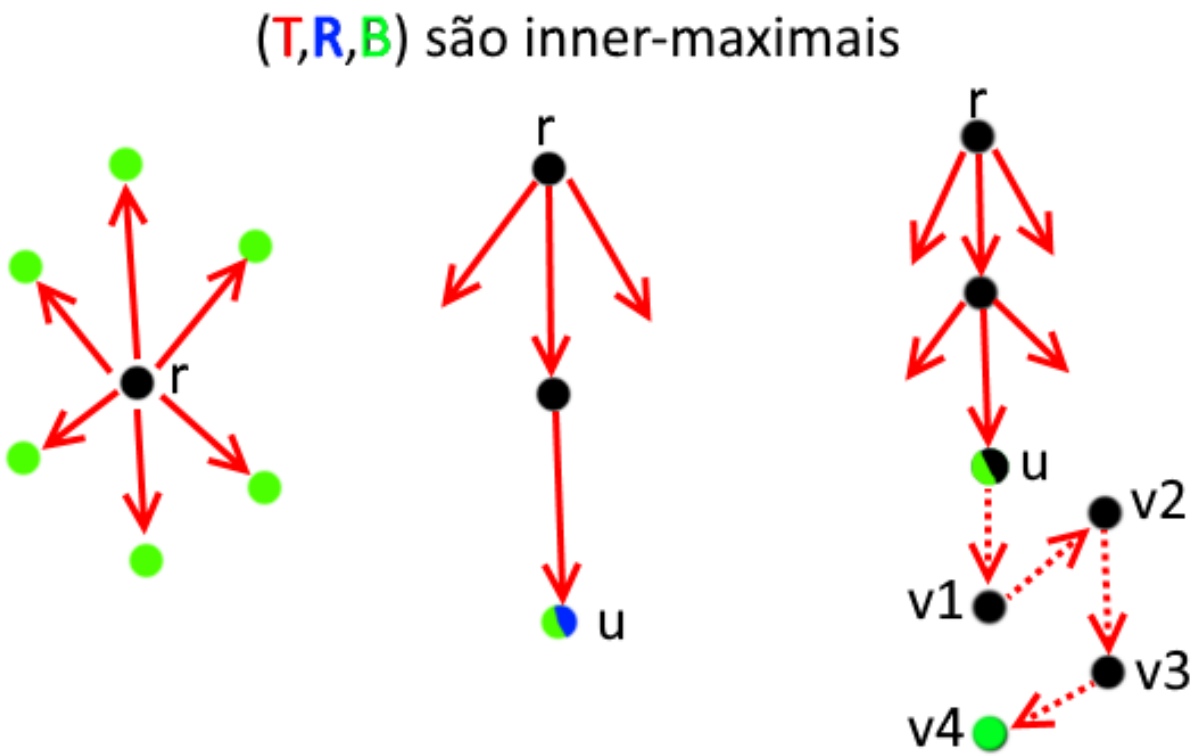
MaxLeaf - Corretude

1) Seja (T, R, B) inner maximal e $u \in B$. Então todas as chamadas de MaxLeaf têm como argumento uma árvore inner-maximal:

A: $(T_r, \emptyset, N(r))$ é inner-maximal.

B: $(T, R \cup \{u\}, B - \{u\})$ da Linha 4 é inner-maximal.

C: $(T, R, B \cup N)$ da Linha 13 é inner-maximal.



MaxLeaf - Corretude

2) MaxLeaf(G, T, R, B, k) só retorna "SIM" se $|R| + |B| \geq k$, i.e., T é uma árvore de k-folhas. Logo MaxLeaf nunca responde "SIM" para isntâncias-não. C.C., se G contém uma árvore de k-folhas enraizada em r, provamos por indução que eventualmente MaxLeaf é chamado com uma árvore de k-folhas.

2) MaxLeaf(G, T, R, B, k) só retorna "SIM" se $|R| + |B| \geq k$, i.e., T é uma árvore de k-folhas. Logo MaxLeaf nunca responde "SIM" para isntâncias-não. C.C., se G contém uma árvore de k-folhas enraizada em r, provamos por indução que eventualmente MaxLeaf é chamado com uma árvore de k-folhas.

- **Hipótese**: Provaremos que se (T, R, B) é inner-maximal tal que existe árvore de k-folhas T' que estende T, então: ou $T = T'$; ou existe árvore de k-folhas (T''', R''', B'''), uma árvore rotulada (T'', R'', B'') tal que $(T''', R''', B''') \succeq (T'', R'', B'') \succ (T, R, B)$ e MaxLeaf é chamada com (T'', R'', B'').

2) MaxLeaf(G, T, R, B, k) só retorna "SIM" se $|R| + |B| \geq k$, i.e., T é uma árvore de k-folhas. Logo MaxLeaf nunca responde "SIM" para isntâncias-não. C.C., se G contém uma árvore de k-folhas enraizada em r, provamos por indução que eventualmente MaxLeaf é chamado com uma árvore de k-folhas.

- **Hipótese**: Provaremos que se (T, R, B) é inner-maximal tal que existe árvore de k-folhas T' que estende T, então: ou $T = T'$; ou existe árvore de k-folhas (T''', R''', B'''), uma árvore rotulada (T'', R'', B'') tal que $(T''', R''', B''') \succeq (T'', R'', B'') \succ (T, R, B)$ e MaxLeaf é chamada com (T'', R'', B'').
- Como o grafo G é finito, MaxLeaf é eventualmente chamado com uma árvore de k-folhas.

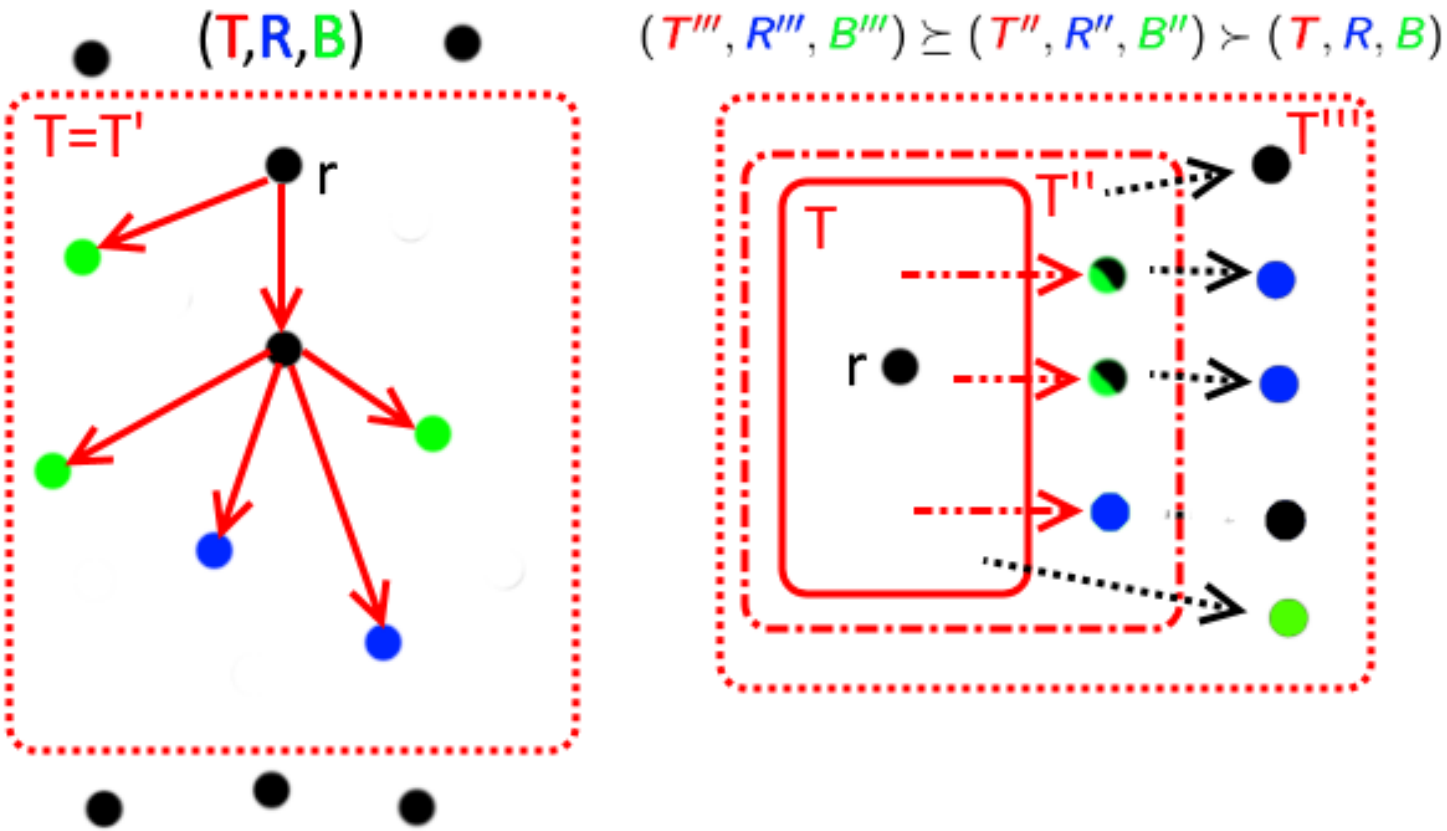


Figura: Ou nossa árvore T já é de k-folhas ou estendemos T para T'' garantido a existência de uma árvore T''' de k-folhas que estende T''.

MaxLeaf - Corretude

- Seja T' uma árvore de k-folhas e a r sua raiz. Então é claro que $T' \succ (T = (\{r\}, \emptyset), \emptyset, \{r\})$. Pelo Lema 4 existe uma árvore de k-folhas T'' que estende $(T_r, \emptyset, N(r))$.

MaxLeaf - Corretude

- Seja T' uma árvore de k-folhas e a r sua raiz. Então é claro que $T' \succ (T = (\{r\}, \emptyset), \emptyset, \{r\})$. Pelo Lema 4 existe uma árvore de k-folhas T'' que estende $(T_r, \emptyset, N(r))$.
- Seja (T, R, B) inner-maximal tal que existe $T' \succeq (T, R, B)$. Se $|leaves(T)| = |R| + |B| \geq k$, então o alg. responde "SIM"corretamente.
- C.C. pelo Lema 1, $B \neq \emptyset$. Se tomarmos então $u \in B$, segundo o Lema 4, temos duas possibilidades:

MaxLeaf - Corretude

- Seja T' uma árvore de k-folhas e a r sua raiz. Então é claro que $T' \succ (T = (\{r\}, \emptyset), \emptyset, \{r\})$. Pelo Lema 4 existe uma árvore de k-folhas T'' que estende $(T_r, \emptyset, N(r))$.
- Seja (T, R, B) inner-maximal tal que existe $T' \succeq (T, R, B)$. Se $|leaves(T)| = |R| + |B| \geq k$, então o alg. responde "SIM"corretamente.
- C.C. pelo Lema 1, $B \neq \emptyset$. Se tomarmos então $u \in B$, segundo o Lema 4, temos duas possibilidades:
  (1) Ou $(T, R \cup \{u\}, B - \{u\})$ é estendível a uma árvore de k-folhas;
  (2) Ou $(T + \{(u, y) : y \in N_{\overline{T}}(u)\}, R, N_{\overline{T}}(u) \cup B - \{u\})$ é estendível a uma árvore de k-folhas.

MaxLeaf - Corretude

- Seja T' uma árvore de k-folhas e a r sua raiz. Então é claro que $T' \succ (T = (\{r\}, \emptyset), \emptyset, \{r\})$. Pelo Lema 4 existe uma árvore de k-folhas T'' que estende $(T_r, \emptyset, N(r))$.
- Seja (T, R, B) inner-maximal tal que existe $T' \succeq (T, R, B)$. Se $|leaves(T)| = |R| + |B| \geq k$, então o alg. responde "SIM"corretamente.
- C.C. pelo Lema 1, $B \neq \emptyset$. Se tomarmos então $u \in B$, segundo o Lema 4, temos duas possibilidades:
  (1) Ou $(T, R \cup \{u\}, B - \{u\})$ é estendível a uma árvore de k-folhas;
  (2) Ou $(T + \{(u, y) : y \in N_{\overline{T}}(u)\}, R, N_{\overline{T}}(u) \cup B - \{u\})$ é estendível a uma árvore de k-folhas.

  Como MaxLeaf é chamado quando ambos os casos (1) e (2) satisfazem a hipótese (Lemas 4 e 5 e Corolário 1), concluímos que o algoritmo funciona. □

Algoritmo MaxLeaf

Algorithm 1 MaxLeaf(G, T, R, B, k)

Entrada: Um grafo G, uma árvore inner-maximal (T, R, B), $k \in \mathbb{N}$
Saída: Existe árvore de k-folhas $T' \succ (T, R, B)$?

1: **if** $|R| + |B| \geq k$ **then return** "SIM"
2: **if** $B = \emptyset$ **then return** "NAO"
3: Escolha um $u \in B$
//Tentar branch em que u é folha
4: **if** MaxLeaf$(G, T, R \cup \{u\}, B - \{u\}, k)$ retorna "SIM" **then return** "SIM"
//Se u não é folha, u deve ser interno em todas as soluções
5: $B \leftarrow B - \{u\}$
6: $N \leftarrow N_{\overline{T}}(u)$
7: $T \leftarrow T \cup \{(u, u') : u' \in N\}$
//Crescer caminho único: Lema 3.5
8: **while** $|N| = 1$ **do**
9: Seja $N = \{v\}$
10: $N \leftarrow N_{\overline{T}}(v)$
11: $T \leftarrow T \cup \{(v, v') : v' \in N\}$
//Não crie branch se não há mais vizinhos: Corolário 3.6
12: **if** $N = \emptyset$ **then return** "NAO"
13: **return** MaxLeaf$(G, T, R, B \cup N, k)$

MaxLeaf - Bound

Lema

Lema 7: *Seja $G = (V, E)$ um grafo e $v \in V$. O número de chamadas recursivas do algoritmo* MaxLeaf *com a entrada $(G, T_v, \emptyset, N(v), k)$ é da ordem de $O(2^{2k-|N(v)|})$.*

Lema

Lema 7: *Seja $G = (V, E)$ um grafo e $v \in V$. O número de chamadas recursivas do algoritmo* MaxLeaf *com a entrada $(G, T_v, \emptyset, N(v), k)$ é da ordem de $O(2^{2k-|N(v)|})$.*

Defina função potencial $\phi(k, R, B) := 2k - 2|R| - |B|$.

Lema

Lema 7: *Seja $G = (V, E)$ um grafo e $v \in V$. O número de chamadas recursivas do algoritmo* MaxLeaf *com a entrada $(G, T_v, \emptyset, N(v), k)$ é da ordem de $O(2^{2k-|N(v)|})$.*

Defina função potencial $\phi(k, R, B) := 2k - 2|R| - |B|$.

- A chamada da Linha 4 reduz potencial em $\phi(k, R, B) - \phi(k, R \cup \{u\}, B - \{u\}) = 1$.

MaxLeaf - Bound

Lema

Lema 7: *Seja $G = (V, E)$ um grafo e $v \in V$. O número de chamadas recursivas do algoritmo* MaxLeaf *com a entrada $(G, T_v, \emptyset, N(v), k)$ é da ordem de $O(2^{2k-|N(v)|})$.*

Defina função potencial $\phi(k, R, B) := 2k - 2|R| - |B|$.

- A chamada da Linha 4 reduz potencial em $\phi(k, R, B) - \phi(k, R \cup \{u\}, B - \{u\}) = 1$.
- A chamada da linha 13 (se caso alcançada temos $|N| \geq 2$) reduz o potencial em pelo menos $\phi(k, R, B) - \phi(k, R, N \cup B - \{u\}) \geq 1$.

MaxLeaf - Bound

- Como $\phi(k, R, B) \leq 0$ implica que $|R| + |B| \geq k$ e o potencial diminui em pelo menos 1 a em cada chamada,

MaxLeaf - Bound

- Como $\phi(k, R, B) \leq 0$ implica que $|R| + |B| \geq k$ e o potencial diminui em pelo menos 1 a em cada chamada, a altura da árvore de busca é de no máximo $\phi(k, R, B) \leq 2k$.

MaxLeaf - Bound

- Como $\phi(k, R, B) \leq 0$ implica que $|R| + |B| \geq k$ e o potencial diminui em pelo menos 1 a em cada chamada, a altura da árvore de busca é de no máximo $\phi(k, R, B) \leq 2k$.
- Então para uma árvore inner-maximal (T, R, B), o número de chamadas recursivas é na ordem de $O(2^{\phi(k,R,B)})$.

MaxLeaf - Bound

- Como $\phi(k, R, B) \leq 0$ implica que $|R| + |B| \geq k$ e o potencial diminui em pelo menos 1 a em cada chamada, a altura da árvore de busca é de no máximo $\phi(k, R, B) \leq 2k$.
- Então para uma árvore inner-maximal (T, R, B), o número de chamadas recursivas é na ordem de $O(2^{\phi(k,R,B)})$.
- Como na primeira chamada do MaxLeaf $R = \emptyset$ e $B = N(v)$,

MaxLeaf - Bound

- Como $\phi(k, R, B) \leq 0$ implica que $|R| + |B| \geq k$ e o potencial diminui em pelo menos 1 a em cada chamada, a altura da árvore de busca é de no máximo $\phi(k, R, B) \leq 2k$.
- Então para uma árvore inner-maximal (T, R, B), o número de chamadas recursivas é na ordem de $O(2^{\phi(k,R,B)})$.
- Como na primeira chamada do MaxLeaf $R = \emptyset$ e $B = N(v)$, a complexidade do algoritmo é de

$$O(2^{\phi(k,\emptyset,N(v))}) = O(2^{2k-|N(v)|}) \subseteq O(4^k) \tag{1}$$

□

Algoritmo MaxLeaf

Algorithm 1 MaxLeaf(G, T, R, B, k)

Entrada: Um grafo G, uma árvore inner-maximal (T, R, B), $k \in \mathbb{N}$
Saída: Existe árvore de k-folhas $T' \succ (T, R, B)$?

1: **if** $|R| + |B| \geq k$ **then return** "SIM"
2: **if** $B = \emptyset$ **then return** "NAO"
3: Escolha um $u \in B$
//Tentar branch em que u é folha
4: **if** MaxLeaf$(G, T, R \cup \{u\}, B - \{u\}, k)$ retorna "SIM" **then return** "SIM"
//Se u não é folha, u deve ser interno em todas as soluções
5: $B \leftarrow B - \{u\}$
6: $N \leftarrow N_{\overline{T}}(u)$
7: $T \leftarrow T \cup \{(u, u') : u' \in N\}$
//Crescer caminho único: Lema 3.5
8: **while** $|N| = 1$ **do**
9: Seja $N = \{v\}$
10: $N \leftarrow N_{\overline{T}}(v)$
11: $T \leftarrow T \cup \{(v, v') : v' \in N\}$
//Não crie branch se não há mais vizinhos: Corolário 3.6
12: **if** $N = \emptyset$ **then return** "NAO"
13: **return** MaxLeaf$(G, T, R, B \cup N, k)$

MaxLeaf - MLST

Teorema

MLST pode ser resolvido em $O(poly(n) + 4^k k^2)$.

MaxLeaf - MLST

Teorema

MLST pode ser resolvido em $O(poly(n) + 4^k k^2)$.

- Usamos o kernel $O(poly(n))$ de Estivill-Castro, logo $O(n) = O(k)$.

MaxLeaf - MLST

Teorema

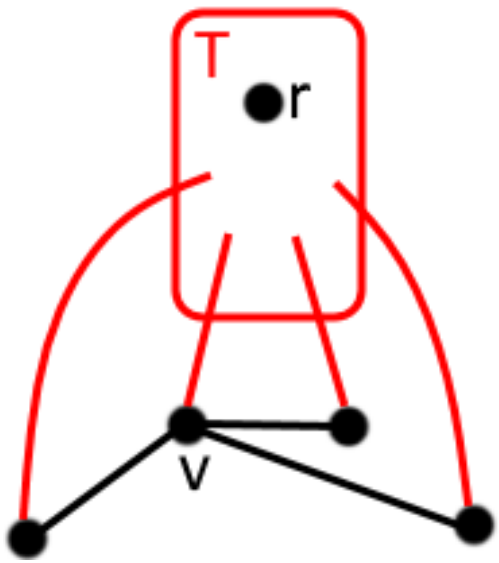
MLST pode ser resolvido em $O(poly(n) + 4^k k^2)$.

- Usamos o kernel $O(poly(n))$ de Estivill-Castro, logo $O(n) = O(k)$.
- Se existe uma árvore T' de k-folhas, então v ou $u \in N(v)$ pode ser raiz dela.

Teorema

MLST pode ser resolvido em $O(poly(n) + 4^k k^2)$.

- Usamos o kernel $O(poly(n))$ de Estivill-Castro, logo $O(n) = O(k)$.
- Se existe uma árvore T' de k-folhas, então v ou $u \in N(v)$ pode ser raiz dela.

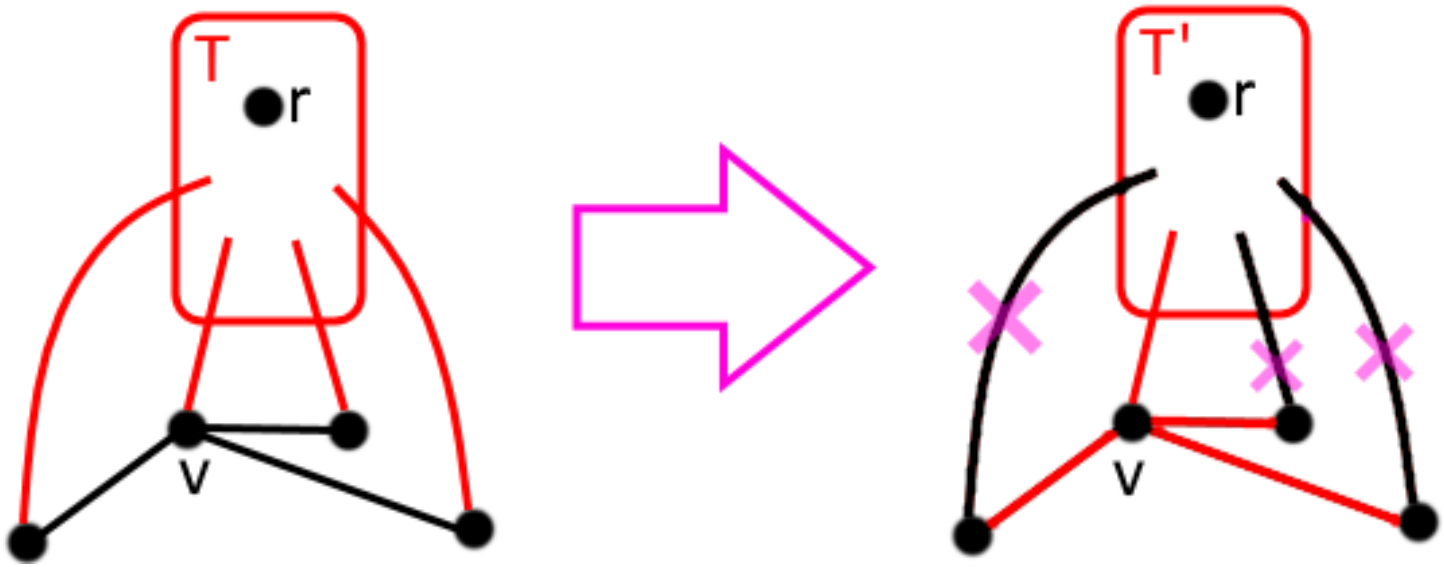


(a) $N[v]$ é folha de T.

Teorema

MLST pode ser resolvido em $O(poly(n) + 4^k k^2)$.

- Usamos o kernel $O(poly(n))$ de Estivill-Castro, logo $O(n) = O(k)$.
- Se existe uma árvore T' de k-folhas, então v ou $u \in N(v)$ pode ser raiz dela.



(a) $N[v]$ é folha de T.

(b) v pode se se tornar raiz.

- Executando MaxLeaf com cada $u \in N[v]$ como raiz, se $grau(v) = d$ for mínimo, segundo o Lema 7, o número de chamadas recursivas é de:

$$\sum_{u \in N[v]} O(2^{\phi(k,\emptyset,N(u))}) \subseteq O((d+1)2^{2k-d}) = O\left(4^k \frac{d+1}{2^d}\right) \subseteq O(4^k)$$

MaxLeaf - MLST

- Executando MaxLeaf com cada $u \in N[v]$ como raiz, se $grau(v) = d$ for mínimo, segundo o Lema 7, o número de chamadas recursivas é de:

$$\sum_{u \in N[v]} O(2^{\phi(k,\emptyset,N(u))}) \subseteq O((d+1)2^{2k-d}) = O\left(4^k \frac{d+1}{2^d}\right) \subseteq O(4^k)$$

- Gasta se $O(k^2)$ por chamada do MaxLeaf.

MaxLeaf - MLST

- Executando MaxLeaf com cada $u \in N[v]$ como raiz, se $grau(v) = d$ for mínimo, segundo o Lema 7, o número de chamadas recursivas é de:

$$\sum_{u \in N[v]} O(2^{\phi(k,\emptyset,N(u))}) \subseteq O((d+1)2^{2k-d}) = O\left(4^k \frac{d+1}{2^d}\right) \subseteq O(4^k)$$

- Gasta se $O(k^2)$ por chamada do MaxLeaf.
- Encontrado uma árvore, executar o BFS gasta mais $O(k^2)$.

MaxLeaf - MLST

- Executando MaxLeaf com cada $u \in N[v]$ como raiz, se $grau(v) = d$ for mínimo, segundo o Lema 7, o número de chamadas recursivas é de:

$$\sum_{u \in N[v]} O(2^{\phi(k,\emptyset,N(u))}) \subseteq O((d+1)2^{2k-d}) = O\left(4^k \frac{d+1}{2^d}\right) \subseteq O(4^k)$$

- Gasta se $O(k^2)$ por chamada do MaxLeaf.
- Encontrado uma árvore, executar o BFS gasta mais $O(k^2)$.
- O tempo total é de $O(poly(n) + 4^k k^2 + k^2) = O(poly(n) + 4^k k^2)$. □

Algoritmo MaxLeaf

Algorithm 1 MaxLeaf(G, T, R, B, k)

Entrada: Um grafo G, uma árvore inner-maximal (T, R, B), $k \in \mathbb{N}$
Saída: Existe árvore de k-folhas $T' \succ (T, R, B)$?

1: **if** $|R| + |B| \geq k$ **then return** "SIM"
2: **if** $B = \emptyset$ **then return** "NAO"
3: Escolha um $u \in B$
//Tentar branch em que u é folha
4: **if** MaxLeaf$(G, T, R \cup \{u\}, B - \{u\}, k)$ retorna "SIM" **then return** "SIM"
//Se u não é folha, u deve ser interno em todas as soluções
5: $B \leftarrow B - \{u\}$
6: $N \leftarrow N_{\overline{T}}(u)$
7: $T \leftarrow T \cup \{(u, u') : u' \in N\}$
//Crescer caminho único: Lema 3.5
8: **while** $|N| = 1$ **do**
9: Seja $N = \{v\}$
10: $N \leftarrow N_{\overline{T}}(v)$
11: $T \leftarrow T \cup \{(v, v') : v' \in N\}$
//Não crie branch se não há mais vizinhos: Corolário 3.6
12: **if** $N = \emptyset$ **then return** "NAO"
13: **return** MaxLeaf$(G, T, R, B \cup N, k)$

MaxLeaf - DMLOT e DMLST

Teorema

DMLOT e DMLST podem ser resolvidos em $O(4^k nm)$.

MaxLeaf - DMLOT e DMLST

Teorema

DMLOT e DMLST podem ser resolvidos em $O(4^k nm)$.

- Executamos MaxLeaf com raiz v para cada possível $v \in V$.

MaxLeaf - DMLOT e DMLST

Teorema

DMLOT e DMLST podem ser resolvidos em $O(4^k nm)$.

- Executamos MaxLeaf com raiz v para cada possível $v \in V$.
- Segundo o Lema 7, o número de chamadas recursivas é limitado por:

$$\sum_{v \in V} O(2^{\phi(k,\emptyset,N(v))}) \subseteq O(n2^{2k}) = O(4^k n)$$

MaxLeaf - DMLOT e DMLST

Teorema

DMLOT e DMLST podem ser resolvidos em $O(4^k nm)$.

- Executamos MaxLeaf com raiz v para cada possível $v \in V$.
- Segundo o Lema 7, o número de chamadas recursivas é limitado por:

$$\sum_{v \in V} O(2^{\phi(k,\emptyset,N(v))}) \subseteq O(n2^{2k}) = O(4^k n)$$

- Note que a cada execução visitamos cada uma das m arestas no máximo uma vez. Logo a complexidade é de $O(4^k nm)$ para o DMLOT.

MaxLeaf - DMLOT e DMLST

Teorema

DMLOT e DMLST podem ser resolvidos em $O(4^k nm)$.

- Executamos MaxLeaf com raiz v para cada possível $v \in V$.
- Segundo o Lema 7, o número de chamadas recursivas é limitado por:

$$\sum_{v \in V} O(2^{\phi(k,\emptyset,N(v))}) \subseteq O(n2^{2k}) = O(4^k n)$$

- Note que a cada execução visitamos cada uma das m arestas no máximo uma vez. Logo a complexidade é de $O(4^k nm)$ para o DMLOT.
- Segundo o Lema 3, se existir árvore geradora T' de k-folhas, pelo menos uma das até n árvores de k-folhas encontradas pode ser estendida para T'. Então usando BFS o tempo total é de $O(4^k nm + n(n+m)) \subseteq O(4^k nm + nm) = O(4^k nm)$. □

Recapitulação e Conclusão

- Vimos que o problema de "Encontrar Árvores de Muitas Folhas"é um problema famoso e proeminente na área de algoritmos parametrizados

Recapitulação e Conclusão

- Vimos que o problema de "Encontrar Árvores de Muitas Folhas"é um problema famoso e proeminente na área de algoritmos parametrizados
- O algoritmo desenvolvido por Kneis, Langer e Rossmanith [9] é recursivo, utiliza o método de limitar a árvore de busca e sua análise é simples.

Recapitulação e Conclusão

- Vimos que o problema de "Encontrar Árvores de Muitas Folhas"é um problema famoso e proeminente na área de algoritmos parametrizados
- O algoritmo desenvolvido por Kneis, Langer e Rossmanith [9] é recursivo, utiliza o método de limitar a árvore de busca e sua análise é simples.
- O algoritmo recebe uma árvore (T, R, B) inner-maximal, seleciona uma folha $u \in B$ e faz um branching em testando se u deve ser fixado em R ou se u deve se tornar vértice interno.

Recapitulação e Conclusão

- Vimos que o problema de "Encontrar Árvores de Muitas Folhas"é um problema famoso e proeminente na área de algoritmos parametrizados
- O algoritmo desenvolvido por Kneis, Langer e Rossmanith [9] é recursivo, utiliza o método de limitar a árvore de busca e sua análise é simples.
- O algoritmo recebe uma árvore (T, R, B) inner-maximal, seleciona uma folha $u \in B$ e faz um branching em testando se u deve ser fixado em R ou se u deve se tornar vértice interno.
- Uma modificação desse algoritmo consegue um tempo de $O(3.72^k n^{O(1)})$ tanto para o MLST quanto para o DMLOT e para o DMLST (a análise é mais complexa).

Fim. Dúvidas?


H.L. Bodlaender.
On linear time minor tests with depth-first search.
Journal of Algorithms, 14(1):1 – 23, 1993.


Paul Bonsma and Frederic Dorn.
Tight Bounds and a Fast FPT Algorithm for Directed Max-Leaf Spanning Tree, pages 222–233.
Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.


Paul Bonsma and Florian Zickfeld.
A 3/2-approximation algorithm for finding spanning trees with many leaves in cubic graphs.
SIAM Journal on Discrete Mathematics, 25(4):1652–1666, 2011.


Paul Bonsma and Florian Zickfeld.
Improved bounds for spanning trees with many leaves.
Discrete Mathematics, 312(6):1178 – 1194, 2012.


Jean Daligault, Gregory Gutin, Eun Jung Kim, and Anders Yeo.
Fpt algorithms and kernels for the directed k-leaf problem.
Journal of Computer and System Sciences, 76(2):144 – 152, 2010.


Rodney G. Downey and Michael R. Fellows.
Parameterized Computational Feasibility, pages 219–244.
Birkhäuser Boston, Boston, MA, 1995.


Vladimir Estivill-Castro, Michael Fellows, Michael Langston, and Frances Rosamond.
Fpt is p-time extremal structure i.
In *Algorithms and Complexity in Durham 2005, Proceedings of the first ACiD Workshop, volume 4 of Texts in Algorithmics*, pages 1–41.
King's College Publications, 2005.


G. Galbiati, F. Maffioli, and A. Morzenti.
A short note on the approximability of the maximum leaves spanning tree problem.
Information Processing Letters, 52(1):45 – 49, 1994.


Joachim Kneis, Alexander Langer, and Peter Rossmanith.
A New Algorithm for Finding Trees with Many Leaves, pages 270–281.

Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.


Hsueh-I Lu and R Ravi.
Approximating maximum leaf spanning trees in almost linear time.
Journal of Algorithms, 29(1):132 – 141, 1998.


R. Fellows Michael, Catherine McCartin, A. Rosamond Frances, and Ulrike Stege.
Coordinatized Kernels and Catalytic Reductions: An Improved FPT Algorithm for Max Leaf Spanning Tree and Other Problems, pages 240–251.
Springer Berlin Heidelberg, Berlin, Heidelberg, 2000.


Roberto Solis-Oba, Paul Bonsma, and Stefanie Lowski.
A 2-approximation algorithm for finding a spanning tree with maximum number of leaves.
Algorithmica, pages 1–15, 2015.