A 3-approximation algorithm for the maximum leaf k-forest problem

Orlando Lee

Institute of Computing - UNICAMP

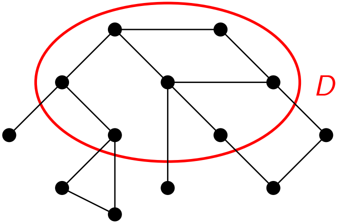
LAGOS 2017 - Marseille

Joint work with M.F. Reis and M.C. San Felice and F. Usberti

Supported by FAPESP

Minimum connected dominating set

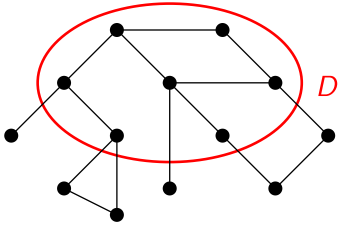
Let $G = (V, E)$ be a connected graph.



$D \subseteq V$ is dominating: every vertex of $V - D$ is adjacent to some vertex in D.

Minimum connected dominating set

Let $G = (V, E)$ be a connected graph.

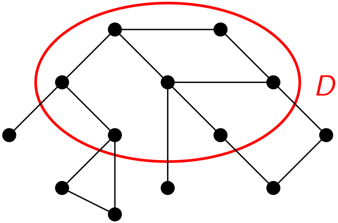


$D \subseteq V$ is dominating: every vertex of $V - D$ is adjacent to some vertex in D.

D is **connected**: $G[D]$ is connected.

Minimum connected dominating set

Let $G = (V, E)$ be a connected graph.



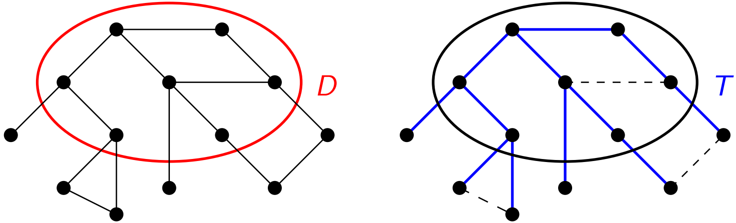
$D \subseteq V$ is dominating: every vertex of $V - D$ is adjacent to some vertex in D.

D is **connected**: $G[D]$ is connected.

Minimum connected dominating set problem: find a smallest connected dominating set in G.

Maximum leaf spanning tree

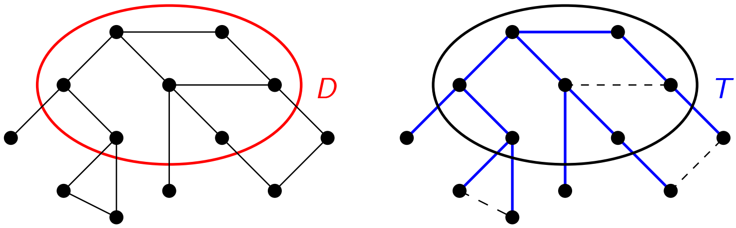
Let $G = (V, E)$ be a connected graph.



Maximum leaf spanning tree: find a spanning tree T with maximum number of **leaves**.

Maximum leaf spanning tree

Let $G = (V, E)$ be a connected graph.

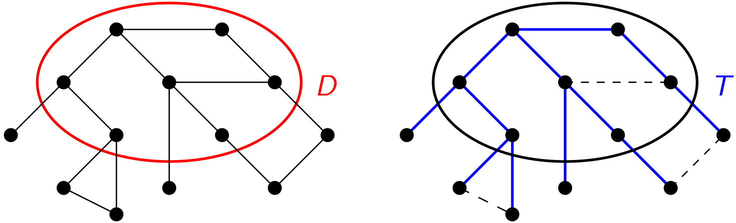


Maximum leaf spanning tree: find a spanning tree T with maximum number of **leaves**.

Equivalent to minimum connected dominating set problem.

Maximum leaf spanning tree

Let $G = (V, E)$ be a connected graph.



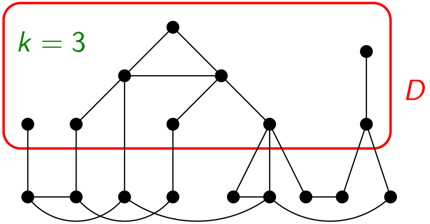
Maximum leaf spanning tree: find a spanning tree T with maximum number of **leaves**.

Equivalent to minimum connected dominating set problem.

D corresponds to the internal vertices of T.

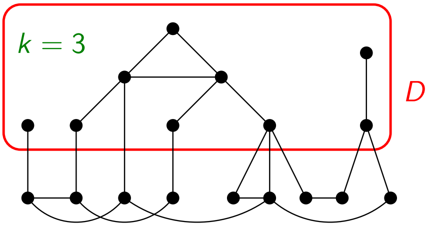
Minimum dominating set with at most k components

Let $G = (V, E)$ be a connected graph and let k be a positive integer.



Minimum dominating set with at most k components

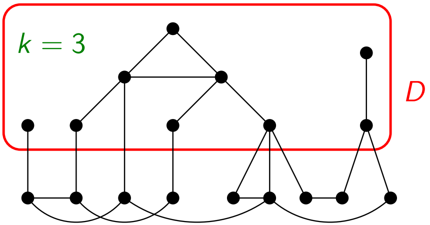
Let $G = (V, E)$ be a connected graph and let k be a positive integer.



Problem: find a smallest dominating set D such that $G[D]$ has at most k components.

Minimum dominating set with at most k components

Let $G = (V, E)$ be a connected graph and let k be a positive integer.

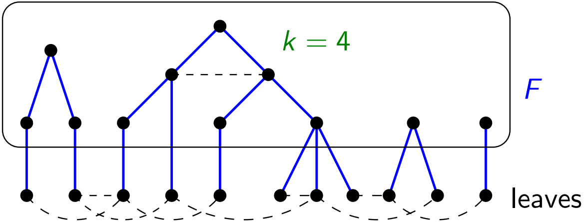


Problem: find a smallest dominating set D such that $G[D]$ has at most k components.

$k = 1$ corresponds to minimum connected dominating set problem.

Maximum leaf (spanning) k-forest

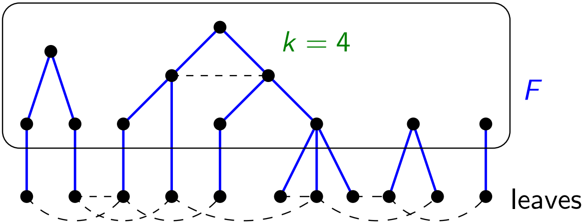
Let $G = (V, E)$ be a connected graph and let k be a positive integer.



k-forest: forest with at most k components (trees).

Maximum leaf (spanning) k-forest

Let $G = (V, E)$ be a connected graph and let k be a positive integer.

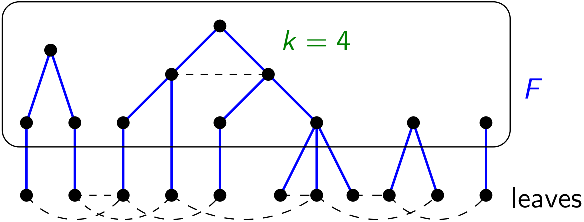


k-forest: forest with at most k components (trees).

Maximum leaf k-forest: find a spanning k-forest with maximum number of **leaves**.

Maximum leaf (spanning) k-forest

Let $G = (V, E)$ be a connected graph and let k be a positive integer.



k-forest: forest with at most k components (trees).

Maximum leaf k-forest: find a spanning k-forest with maximum number of **leaves**.

Remark: a tree in F with exactly two vertices has only one leaf.

Known results

Minimum connected dominating set

Known results

Minimum connected dominating set

- NP-hard (Garey and Johnson, 1979).

Known results

Minimum connected dominating set

- NP-hard (Garey and Johnson, 1979).
- $O(\Delta)$-approximation algorithm (Guha and Khuller, 1988).

Known results

Minimum connected dominating set

- NP-hard (Garey and Johnson, 1979).
- $O(\Delta)$-approximation algorithm (Guha and Khuller, 1988).

Maximum leaf spanning tree

Known results

Minimum connected dominating set

- NP-hard (Garey and Johnson, 1979).
- $O(\Delta)$-approximation algorithm (Guha and Khuller, 1988).

Maximum leaf spanning tree

- MAX SNP-hard (Galbiati, Maffioli and Morzenti, 1994).

Known results

Minimum connected dominating set

- NP-hard (Garey and Johnson, 1979).
- $O(\Delta)$-approximation algorithm (Guha and Khuller, 1988).

Maximum leaf spanning tree

- MAX SNP-hard (Galbiati, Maffioli and Morzenti, 1994).
- 3-approximation algorithm (Lu and Ravi, 1996).

Known results

Minimum connected dominating set

- NP-hard (Garey and Johnson, 1979).
- $O(\Delta)$-approximation algorithm (Guha and Khuller, 1988).

Maximum leaf spanning tree

- MAX SNP-hard (Galbiati, Maffioli and Morzenti, 1994).
- 3-approximation algorithm (Lu and Ravi, 1996).
- 2-approximation algorithm (Solis-Oba, 1998; Solis-Oba, Bonsma and Lowski, 2017).

Our contribution

- We present a 3-approximation algorithm for maximum leaf k-forest.

Our contribution

- We present a 3-approximation algorithm for maximum leaf k-forest.
- The algorithm is inspired on Lu and Ravi's 3-approximation algorithm for maximum leaf spanning tree.

Our contribution

- We present a 3-approximation algorithm for maximum leaf k-forest.
- The algorithm is inspired on Lu and Ravi's 3-approximation algorithm for maximum leaf spanning tree.
- A key ingredient in Lu and Ravi's algorithm (and ours) is the concept of leafy forest.

Leafy forest

A leafy tree is a tree T such that:

- T contains at least a vertex of degree 3, and
- each degree 2 vertex in T is adjacent to two degree 3 vertices of T.

Leafy forest

A leafy tree is a tree T such that:

- T contains at least a vertex of degree 3, and
- each degree 2 vertex in T is adjacent to two degree 3 vertices of T.

A leafy forest is a forest in which every component is a **leafy tree**.

Leafy forest

A leafy tree is a tree T such that:

- T contains at least a vertex of degree 3, and
- each degree 2 vertex in T is adjacent to two degree 3 vertices of T.

A leafy forest is a forest in which every component is a **leafy tree**.

A leafy forest is **maximal** if it cannot be extended to a larger (more edges or vertices) leafy forest.

Building a maximal leafy tree (Lu and Ravi)

Start a new tree T by adding a vertex with degree ≥ 3 and all its neighbors.

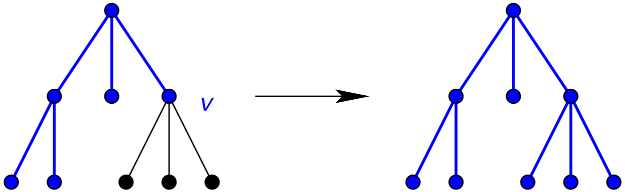
Repeat one of the following expansions until it is no longer possible:

Building a maximal leafy tree (Lu and Ravi)

Start a new tree T by adding a vertex with degree ≥ 3 and all its neighbors.

Repeat one of the following expansions until it is no longer possible:

Expansion Type 1: a leaf v of T has ≥ 2 neighbors in $G - V(T)$



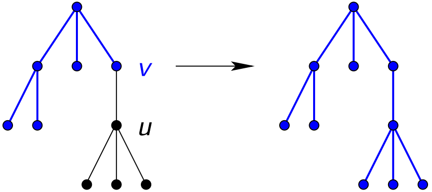
Add all those neighbors to T.

Building a maximal leafy tree (Lu and Ravi)

Start a new tree T by adding a vertex with degree ≥ 3 and all its neighbors.

Repeat one of the following expansions until it is no longer possible:

Expansion Type 2: a leaf v of T is adjacent to a vertex u in $G - V(T)$ with degree ≥ 3



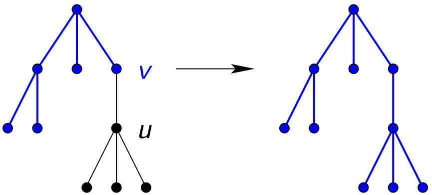
Add u and all its neighbors in $G - V(T)$ to T.

Building a maximal leafy tree (Lu and Ravi)

Start a new tree T by adding a vertex with degree ≥ 3 and all its neighbors.

Repeat one of the following expansions until it is no longer possible:

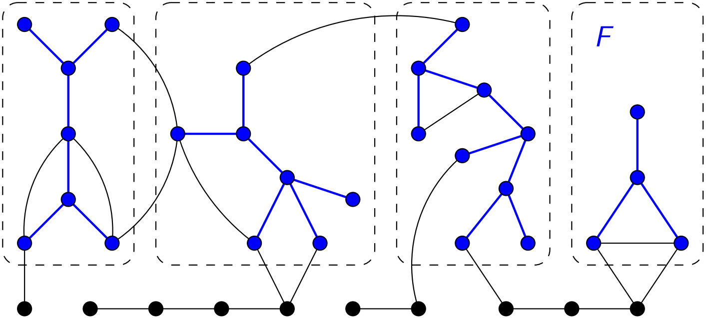
Expansion Type 2: a leaf v of T is adjacent to a vertex u in $G - V(T)$ with degree ≥ 3



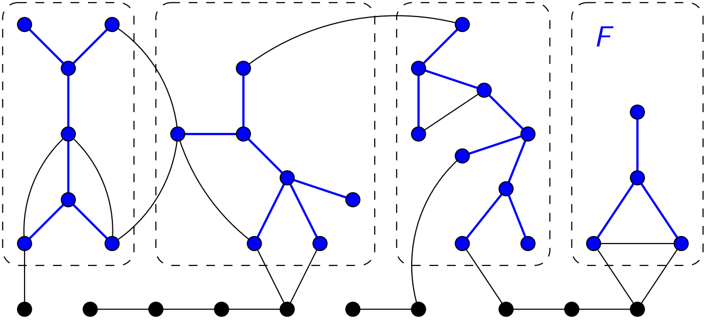
Add u and all its neighbors in $G - V(T)$ to T.

Algorithm: repeatedly build a new leafy tree until no vertex of degree ≥ 3 remains.

Maximal leafy forest

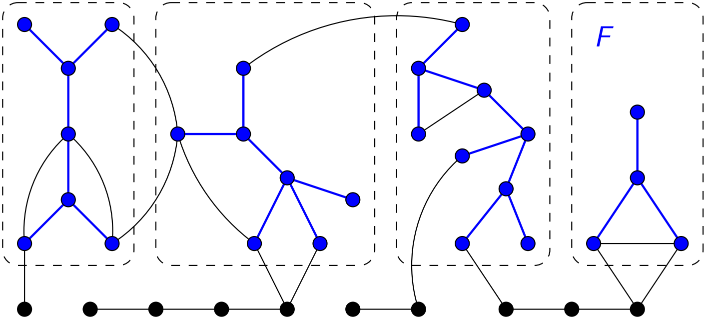


Maximal leafy forest



- Each component of $G - V(F)$ is a **path**.

Maximal leafy forest



- Each component of $G - V(F)$ is a **path**.
- Only **leaves** of F can have neighbors in $G - V(F)$.

Lu and Ravi's 3-approximation algorithm

Type	Name	#trees	#leaves
maximal leafy forest	F	q	$\ell(F)$
optimum spanning tree	T^*	1	$\ell(T^*)$
Lu and Ravi's tree	T'	1	$\ell(T')$

Lu and Ravi's 3-approximation algorithm

Type	Name	#trees	#leaves
maximal leafy forest	F	q	$\ell(F)$
optimum spanning tree	T^*	1	$\ell(T^*)$
Lu and Ravi's tree	T'	1	$\ell(T')$

Goal: find T' such that $3\ell(T') \geq \ell(T^*)$.

Lu and Ravi's 3-approximation algorithm

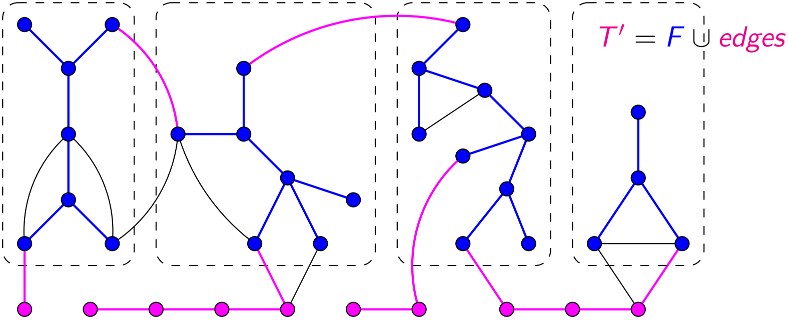
Type	Name	#trees	#leaves
maximal leafy forest	F	q	$\ell(F)$
optimum spanning tree	T^*	1	$\ell(T^*)$
Lu and Ravi's tree	T'	1	$\ell(T')$

Goal: find T' such that $3\ell(T') \geq \ell(T^*)$.

Key Lemma. $3\ell(F) \geq \ell(T^*) + 6q - 1$. (Lu and Ravi)

Lu and Ravi's 3-approximation algorithm

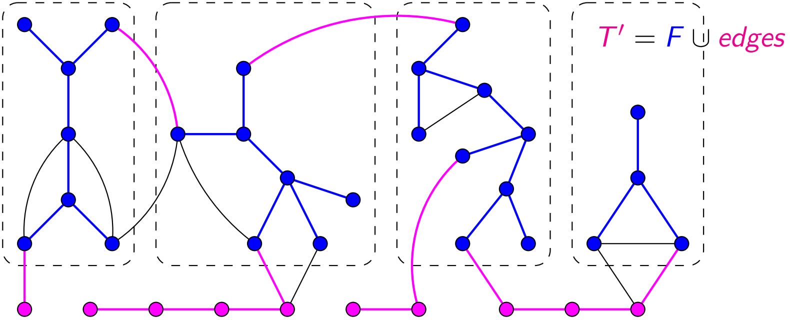
Key Lemma. $3\ell(F) \geq \ell(T^*) + 6q - 1$. (Lu and Ravi)



Build T' by connecting trees in F through edges/paths and then connnect the forest to the remaining outer paths.

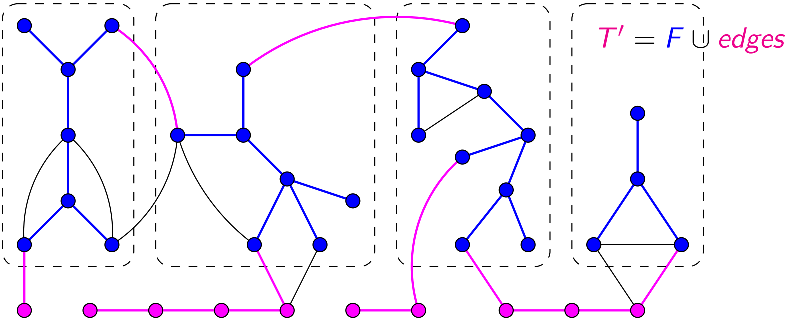
Lu and Ravi's 3-approximation algorithm

Key Lemma. $3\ell(F) \geq \ell(T^*) + 6q - 1$. (Lu and Ravi)



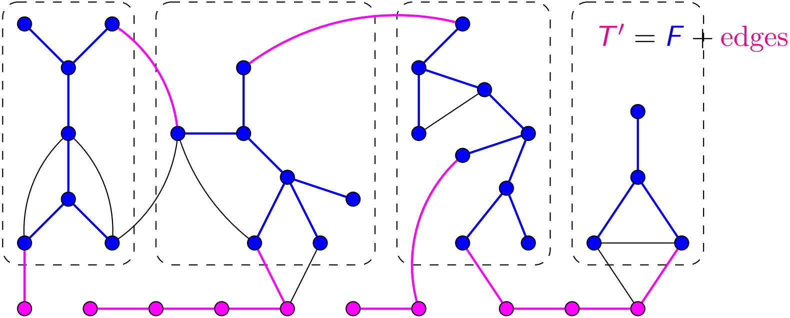
We might lose leaves by connecting trees through edges/paths.
We do not lose leaves by connecting F to an outer path.

Key Lemma. $3\ell(F) \geq \ell(T^*) + 6q - 1$. (Lu and Ravi)



$\ell(T') \geq \ell(F) - 2(q-1)$ (lost at most $2(q-1)$ leaves of F)

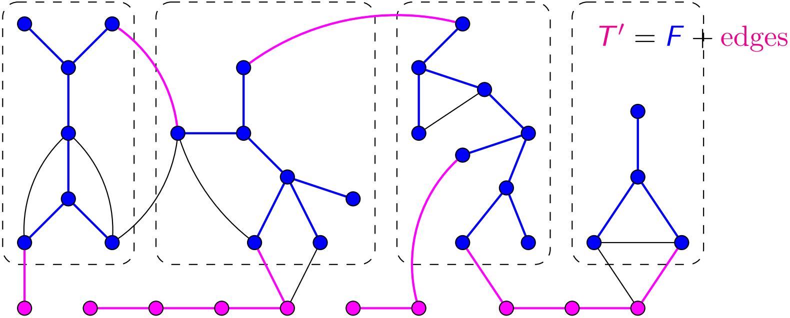
Key Lemma. $3\ell(F) \geq \ell(T^*) + 6q - 1$. (Lu and Ravi)



$3\ell(T') \geq 3[\ell(F) - 2(q-1)]$

Lu and Ravi's 3-approximation algorithm

Key Lemma. $3\ell(F) \geq \ell(T^*) + 6q - 1$. (Lu and Ravi)



$3\ell(T') \geq 3[\ell(F) - 2(q-1)] = 3\ell(F) - 6q + 6 > \ell(T^*)$
(3-approximation)

Our algorithm for maximum leaf k-forest

Type	Name	#trees	#leaves
maximal leafy forest	F	q	$\ell(F)$
optimum k-forest	F^*	q^*	$\ell(F^*)$
algorithm's k-forest	F'	q'	$\ell(F')$

Our algorithm for maximum leaf k-forest

Type	Name	#trees	#leaves
maximal leafy forest	F	q	$\ell(F)$
optimum k-forest	F^*	q^*	$\ell(F^*)$
algorithm's k-forest	F'	q'	$\ell(F')$

Goal: find F' such that $3\ell(F') \geq \ell(F^*)$.

Our algorithm for maximum leaf k-forest

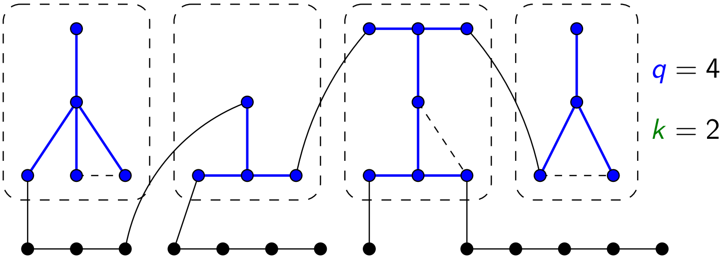
Type	Name	#trees	#leaves
maximal leafy forest	F	q	$\ell(F)$
optimum k-forest	F^*	q^*	$\ell(F^*)$
algorithm's k-forest	F'	q'	$\ell(F')$

Goal: find F' such that $3\ell(F') \geq \ell(F^*)$.

Key Lemma. $3\ell(F) \geq \ell(F^*) + 6q - 2q^* + 1$.

Key Lemma. $3\ell(F) \geq \ell(F^*) + 6q - 2q^* + 1.$

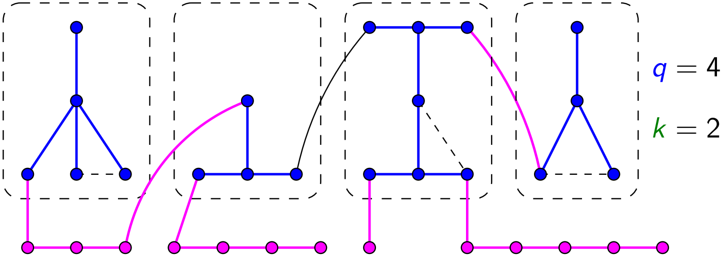
Case 1: $q \geq k$



Our algorithm for maximum leaf k-forest

Key Lemma. $3\ell(F) \geq \ell(F^*) + 6q - 2q^* + 1.$

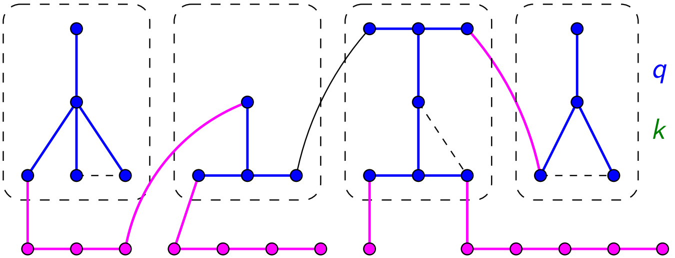
Case 1: $q \geq k$



Build F' by connecting trees in F through edges/paths and then connnect the forest to the remaining outer paths.

Key Lemma. $3\ell(F) \geq \ell(F^*) + 6q - 2q^* + 1.$

Case 1: $q \geq k$

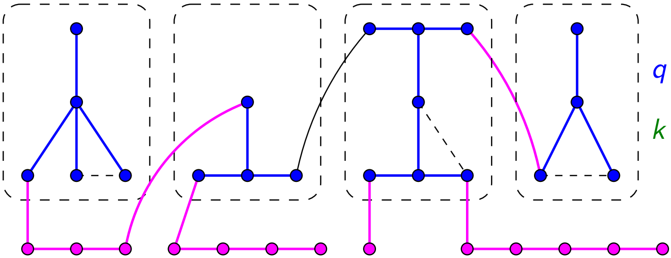


$\ell(F') \geq \ell(F) - 2(q-k)$ (lost at most $2(q-k)$ leaves of F)

Our algorithm for maximum leaf k-forest

Key Lemma. $3\ell(F) \geq \ell(F^*) + 6q - 2q^* + 1.$

Case 1: $q \geq k$



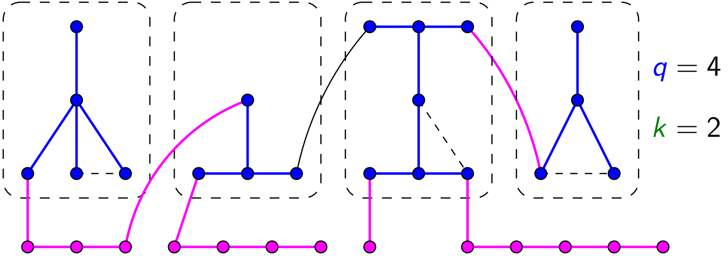
$q = 4$

$k = 2$

$3\ell(F') \geq 3\ell(F) - 6(q - k)$

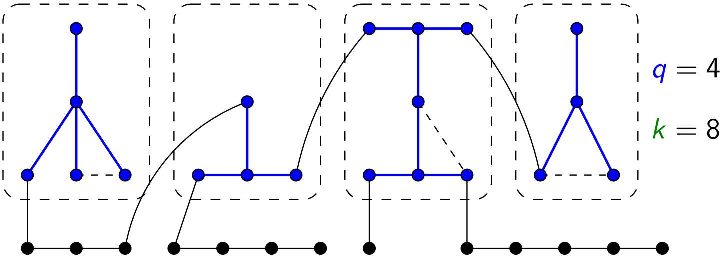
Key Lemma. $3\ell(F) \geq \ell(F^*) + 6q - 2q^* + 1.$

Case 1: $q \geq k$



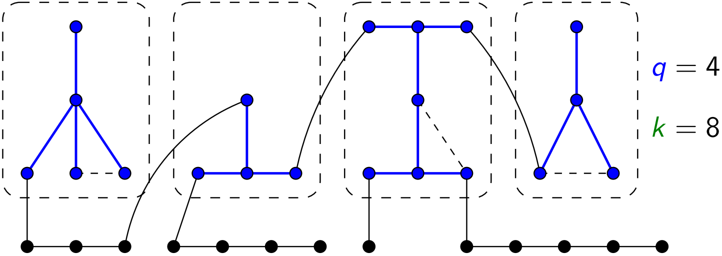
$$3\ell(F') \geq 3\ell(F) - 6(q-k) \geq \ell(F^*) + 6q - 2q^* + 1 - 6q + 6k \geq \ell(F^*)$$

Case 2: $q < k$



- **Natural try:** connect F to the outer paths and return the k-forest.

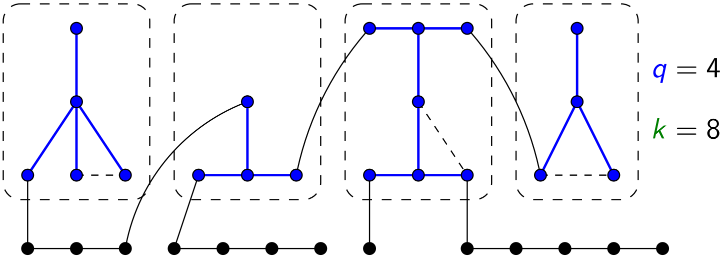
Case 2: $q < k$



- **Natural try:** connect F to the outer paths and return the k-forest.
- This may not work (few leaves compared to the optimum).

Our algorithm for maximum leaf k-forest

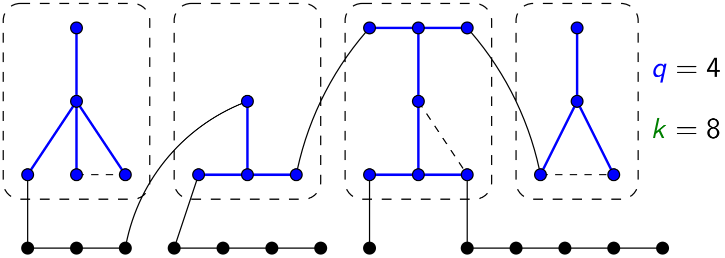
Case 2: $q < k$



- **Natural try:** connect F to the outer paths and return the k-forest.
- This may not work (few leaves compared to the optimum).
- Instead we try to increase the number of trees to k as follows.

Our algorithm for maximum leaf k-forest

Case 2: $q < k$



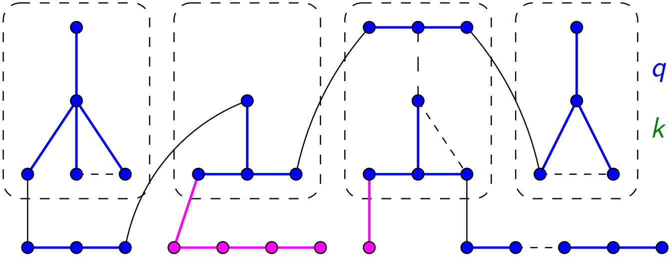
While there are less than k trees in F try to

- delete an edge e from F such that $\ell(F - e) > \ell(F)$, or
- add a nontrivial component of $G - V(F)$ to F.

Then connect the remaining outer paths to F.

Our algorithm for maximum leaf k-forest

Case 2a: $q < k$ and **we added** $(k - q)$ **new trees**



While there are less than k trees in F try to

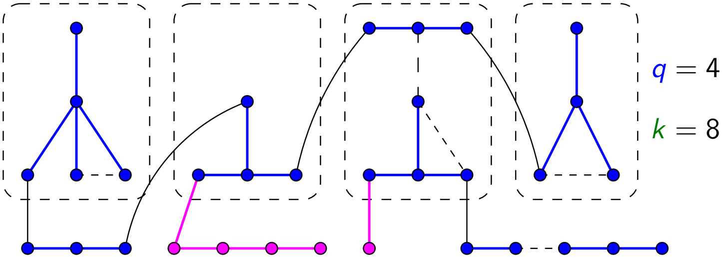
- delete an edge e from F such that $\ell(F - e) > \ell(F)$, or
- add a nontrivial component of $G - V(F)$ to F.

Then connect the remaining outer paths to F.

Our algorithm for maximum leaf k-forest

Key Lemma. $3\ell(F) \geq \ell(F^*) + 6q - 2q^* + 1.$

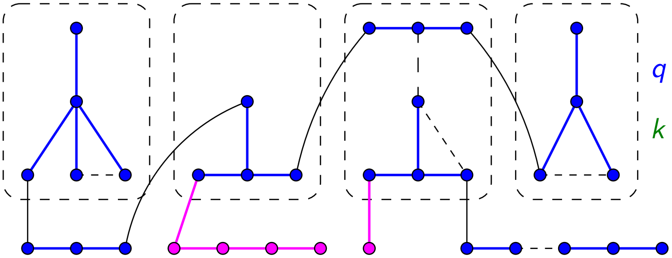
Case 2a: $q < k$ and **we added $(k - q)$ new trees**



$\ell(F') \geq \ell(F) + (k - q)$ (gained $(k - q)$ leaves)

Key Lemma. $3\ell(F) \geq \ell(F^*) + 6q - 2q^* + 1.$

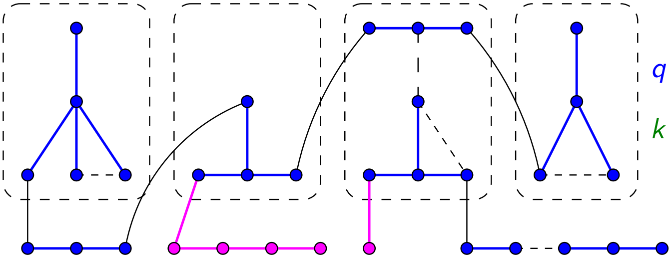
Case 2a: $q < k$ and **we added** $(k - q)$ **new trees**



$$3\ell(F') \geq 3\ell(F) + 3(k - q)$$

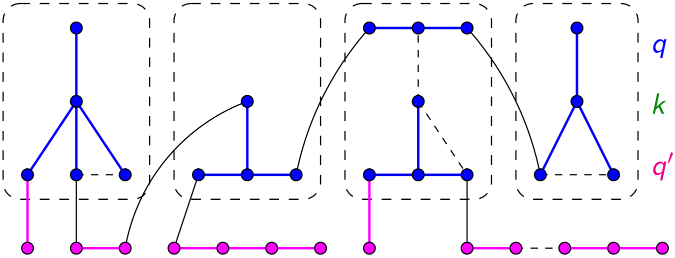
Key Lemma. $3\ell(F) \geq \ell(F^*) + 6q - 2q^* + 1.$

Case 2a: $q < k$ and **we added $(k - q)$ new trees**



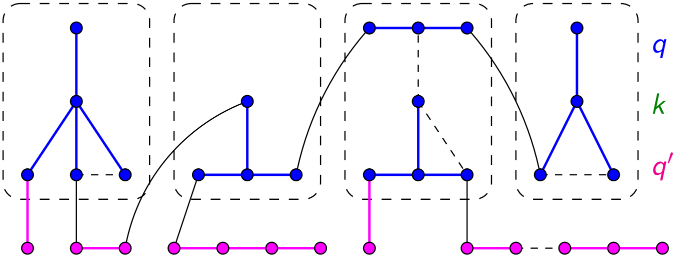
$$3\ell(F') \geq 3\ell(F) + 3(k - q) \geq \ell(F^*) + 6q - 2q^* + 1 + 3k - 3q \geq \ell(F^*)$$

Case 2b: $q < k$ and **we did not add** $(k - q)$ **new trees**



Every degree 2 vertex of F' is adjacent to a leaf.

Case 2b: $q < k$ and **we did not add $(k - q)$ new trees**



$q = 4$

$k = 20$

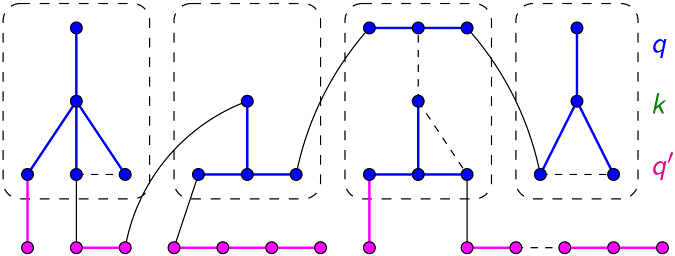
$q' = 9$

Every degree 2 vertex of F' is adjacent to a leaf.

So $3\ell(T') \geq |V(T')|$ for each tree T' in F'.

Our algorithm for maximum leaf k-forest

Case 2b: $q < k$ and **we did not add $(k-q)$ new trees**



Every degree 2 vertex of F' is adjacent to a leaf.

So $3\ell(T') \geq |V(T')|$ for each tree T' in F'.

Thus, $3\ell(F') \geq |V(G)| \geq \ell(F^*)$.

Conclusion and future work

- Improve the analysis of the performance guarantee.

Conclusion and future work

- Improve the analysis of the performance guarantee.
- Try to extend Solis-Oba's algorithm to maximum leaf k-forest.

Conclusion and future work

- Improve the analysis of the performance guarantee.
- Try to extend Solis-Oba's algorithm to maximum leaf k-forest.
- Design $O(\Delta)$ or $O(\log n)$-approximation algorithm for minimum dominating set with at most k components.

Conclusion and future work

- Improve the analysis of the performance guarantee.
- Try to extend Solis-Oba's algorithm to maximum leaf k-forest.
- Design $O(\Delta)$ or $O(\log n)$-approximation algorithm for minimum dominating set with at most k components.

Thank you!