Parameterized Approximation Schemes for Steiner Trees with Small Number of Steiner Vertices

Pavel Dvořák, Andreas Emil Feldmann, Dušan Knop,
Tomáš Masařík, Tomáš Toufar, Pavel Veselý

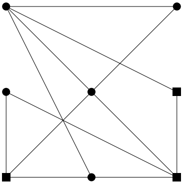
Faculty of Mathematics and Physics,
Charles University,
Prague, Czech Republic.

STACS 2018,
Caen, France

Steiner Tree

Steiner Tree
Input: A graph $G = (V, E)$, a set of terminals $R \subseteq V$.
Task: Find a tree $T \subseteq E$ of smallest size such that (R, T) is connected.

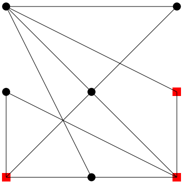
A vertex in $V \setminus R$ is called a Steiner vertex.
The number of Steiner vertices in the optimal solution is denoted by p, (i.e. $|V(T^*) \setminus R|$ for the optimum T^*).



Steiner Tree

Steiner Tree
Input: A graph $G = (V, E)$, a set of terminals $R \subseteq V$.
Task: Find a tree $T \subseteq E$ of smallest size such that (R, T) is connected.

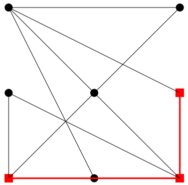
A vertex in $V \setminus R$ is called a Steiner vertex.
The number of Steiner vertices in the optimal solution is denoted by p, (i.e. $|V(T^*) \setminus R|$ for the optimum T^*).



Steiner Tree

Steiner Tree
Input: A graph $G = (V, E)$, a set of terminals $R \subseteq V$.
Task: Find a tree $T \subseteq E$ of smallest size such that (R, T) is connected.

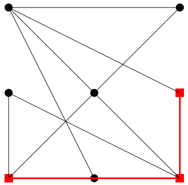
A vertex in $V \setminus R$ is called a Steiner vertex.
The number of Steiner vertices in the optimal solution is denoted by p, (i.e. $|V(T^*) \setminus R|$ for the optimum T^*).



Steiner Tree

Steiner Tree
Input: A graph $G = (V, E)$, a set of terminals $R \subseteq V$.
Task: Find a tree $T \subseteq E$ of smallest size such that (R, T) is connected.

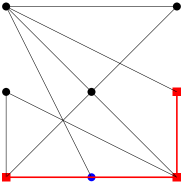
A vertex in $V \setminus R$ is called a Steiner vertex.
The number of Steiner vertices in the optimal solution is denoted by p, (i.e. $|V(T^*) \setminus R|$ for the optimum T^*).



Steiner Tree

Steiner Tree
Input: A graph $G = (V, E)$, a set of terminals $R \subseteq V$.
Task: Find a tree $T \subseteq E$ of smallest size such that (R, T) is connected.

A vertex in $V \setminus R$ is called a Steiner vertex.
The number of Steiner vertices in the optimal solution is denoted by p, (i.e. $|V(T^*) \setminus R|$ for the optimum T^*).



Steiner Tree

Weighted Steiner Tree
Input: A graph $G = (V, E)$, a set of terminals $R \subseteq V$, and a weight function $w\colon E \to \mathbb{R}^+$.
Task: Find a tree $T \subseteq E$ of smallest weight such that (R, T) is connected.

Steiner Tree

Weighted Steiner Tree
Input: A graph $G = (V, E)$, a set of terminals $R \subseteq V$, and a weight function $w \colon E \to \mathbb{R}^+$.
Task: Find a tree $T \subseteq E$ of smallest weight such that (R, T) is connected.

Directed Steiner Tree
Input: A directed graph $G = (V, A)$, a set of terminals $R \subseteq V$, a root $r \in R$.
Task: Find an arborescence $T \subseteq A$ of smallest size such that every terminal is reachable from r by edges in T.

Steiner Tree and Parameters

Two natural parameters of the problem.

- the number of terminals (denoted by $|R|$),
- the number of Steiner vertices in the optimal solution (denoted by p) i.e. $|V(T^*) \setminus R|$.

Steiner Tree and Parameters

Two natural parameters of the problem.

- the number of terminals (denoted by $|R|$),
- the number of Steiner vertices in the optimal solution (denoted by p) i.e. $|V(T^*) \setminus R|$.

Note, that if the number of Steiner vertices is considered as a parameter instead there is a trivial FPT algorithm.

Steiner Tree – Known Results

Parameterized complexity:

- FPT w.r.t. number of terminals
  - $3^k n^3$ algorithm in the 70s [Dreyfus & Wagner 71],
  - the best known now is $2^k n^2$ [Björklund et al. 07] for the unweighted case.
- W[1]-hard w.r.t. p

Steiner Tree – Known Results

Parameterized complexity:

- FPT w.r.t. number of terminals
  - $3^k n^3$ algorithm in the 70s [Dreyfus & Wagner 71],
  - the best known now is $2^k n^2$ [Björklund et al. 07] for the unweighted case.
- W[1]-hard w.r.t. p

Approximation:

- 2-approximation is simple
- $(\ln 4 + \varepsilon)$-approximation algorithm [Byrka et al. 13]
- (96/95)-approximation is NP-hard (which means no approximation scheme exists unless $\mathsf{P} = \mathsf{NP}$) [Chlebík and Chlebíková 02]

Steiner Tree – Known Results

Parameterized complexity:

- FPT w.r.t. number of terminals
  - $3^k n^3$ algorithm in the 70s [Dreyfus & Wagner 71],
  - the best known now is $2^k n^2$ [Björklund et al. 07] for the unweighted case.
- W[1]-hard w.r.t. p

Approximation:

- 2-approximation is simple
- $(\ln 4 + \varepsilon)$-approximation algorithm [Byrka et al. 13]
- (96/95)-approximation is NP-hard (which means no approximation scheme exists unless P = NP) [Chlebík and Chlebíková 02]

Since Steiner Tree is both hard to approximate and W[1]-hard w.r.t. p, it is an excellent target for parameterized approximation.

Steiner Tree – Our Results

Theorem

For any computable functions g, f, it is impossible to compute an $f(p)$-approximation for the Weighted Directed Steiner Tree problem parameterized by p in time $g(p) \cdot n^{O(1)}$, unless $\mathsf{W}[1] = \mathsf{FPT}$.

Results	Undirected	Directed
Unweighted		
Weighted		$f(p)$-approx. hard

Steiner Tree – Our Results

Definition (EPAS)

Efficient Parameterized Approximation Scheme for minimization problem is defined as follows: For all $\varepsilon > 0$ there exists an algorithm that runs in time $f(\varepsilon, p) \cdot n^{O(1)}$ and returns $(1 + \varepsilon)$-approximation.

Steiner Tree – Our Results

Theorem

There is an EPAS for the Unweighted Directed Steiner Tree problem

in time $2^{p^2/\varepsilon} \cdot n^{O(1)}$.

Results	Undirected	Directed
Unweighted		EPAS
Weighted		$f(p)$-approx. hard

Steiner Tree – Our Results

Theorem

There is an EPAS for the Weighted Undirected Steiner Tree problem

in time $2^{O(p^2/\varepsilon^4)} \cdot n^{O(1)}$.

Results	Undirected	Directed
Unweighted		EPAS
Weighted	EPAS	$f(p)$-approx. hard

Steiner Tree – Our Results

Theorem

There is an EPAS for the Weighted Undirected Steiner Tree problem

$$\text{in time } 2^{O(p^2/\varepsilon^4)} \cdot n^{O(1)}.$$

Results	Undirected	Directed
Unweighted		EPAS
Weighted	EPAS	$f(p)$-approx. hard

EPAS for Undirected Unweighted Steiner Tree – Sketch

- We can contract every edge between terminals.

EPAS for Undirected Unweighted Steiner Tree – Sketch

- We can contract every edge between terminals.
- If there is a Steiner vertex with 'many' terminals as neighbors then we contract them. (Denote its neighbors that are terminals by Q.)

EPAS for Undirected Unweighted Steiner Tree – Sketch

- We can contract every edge between terminals.
- If there is a Steiner vertex with 'many' terminals as neighbors then we contract them. (Denote its neighbors that are terminals by Q.)
- The contraction cost compared to a lower bound for the optimum cost $\frac{|Q|}{|Q|-1}$.

EPAS for Undirected Unweighted Steiner Tree – Sketch

- We can contract every edge between terminals.
- If there is a Steiner vertex with 'many' terminals as neighbors then we contract them. (Denote its neighbors that are terminals by Q.)
- The contraction cost compared to a lower bound for the optimum cost $\frac{|Q|}{|Q|-1}$.
- If this fraction is less than $1+\varepsilon$ then we can perform contractions recursively.

EPAS for Undirected Unweighted Steiner Tree – Sketch

- We can contract every edge between terminals.
- If there is a Steiner vertex with 'many' terminals as neighbors then we contract them. (Denote its neighbors that are terminals by Q.)
- The contraction cost compared to a lower bound for the optimum cost $\frac{|Q|}{|Q|-1}$.
- If this fraction is less than $1 + \varepsilon$ then we can perform contractions recursively.
- If not then we have a bounded number of terminals in terms of ε and p: $|R| \leq \frac{2p}{\varepsilon}$.

EPAS for Undirected Unweighted Steiner Tree – Sketch

- We can contract every edge between terminals.
- If there is a Steiner vertex with 'many' terminals as neighbors then we contract them. (Denote its neighbors that are terminals by Q.)
- The contraction cost compared to a lower bound for the optimum cost $\frac{|Q|}{|Q|-1}$.
- If this fraction is less than $1 + \varepsilon$ then we can perform contractions recursively.
- If not then we have a bounded number of terminals in terms of ε and p: $|R| \leq \frac{2p}{\varepsilon}$.
- So, we can use a known algorithm parameterized by $|R|$.

Steiner Tree – Our Results

Theorem

There is an EPAS for the Weighted Undirected Steiner Tree problem in time $2^{O(p^2/\varepsilon^4)} \cdot n^{O(1)}$.

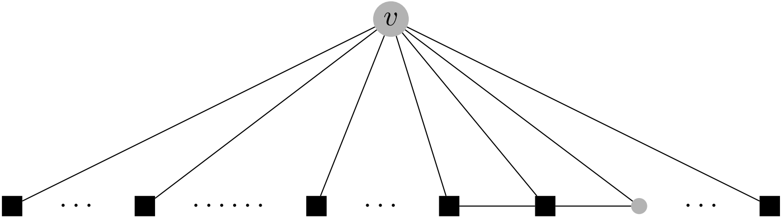
Results	Undirected	Directed
Unweighted		EPAS
Weighted	EPAS	$f(p)$-approx. hard

Overview of algorithm

We define a star C as the central vertex v and the subset Q of its adjacent terminals, where $|Q| \geq 2$.

We define a ratio of a star C, as

$$\frac{w(C)}{(|Q|-1)}.$$

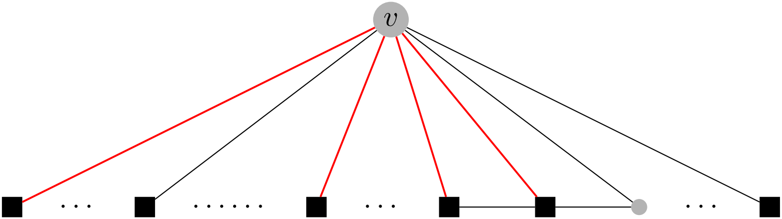


Overview of algorithm

We define a star C as the central vertex v and the subset Q of its adjacent terminals, where $|Q| \geq 2$.

We define a ratio of a star C, as

$$\frac{w(C)}{(|Q|-1)}.$$

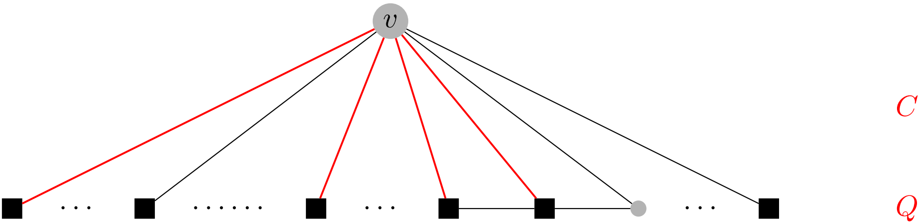


Overview of algorithm

We define a star C as the central vertex v and the subset Q of its adjacent terminals, where $|Q| \geq 2$.

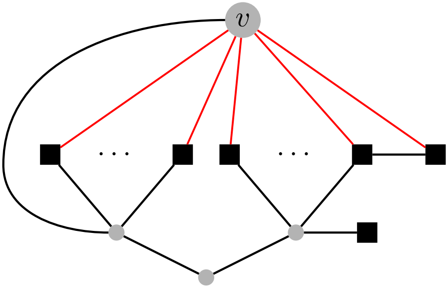
We define a ratio of a star C, as

$$\frac{w(C)}{(|Q|-1)}.$$



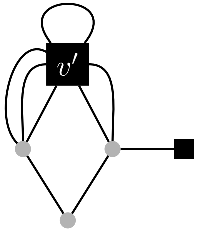
Overview of algorithm

Phase 1: We contract the star with the best ratio until the number of terminals (|R|) is small ($\mathcal{O}(p^2/\varepsilon^4)$)
Phase 2: Run an FPT algorithm for parameterization by $|R|$.



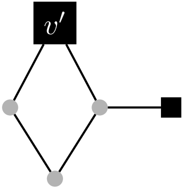
Overview of algorithm

Phase 1: We contract the star with the best ratio until the number of terminals ($|\mathrm{R}|$) is small ($\mathcal{O}(p^2/\varepsilon^4)$)
Phase 2: Run an FPT algorithm for parameterization by $|R|$.



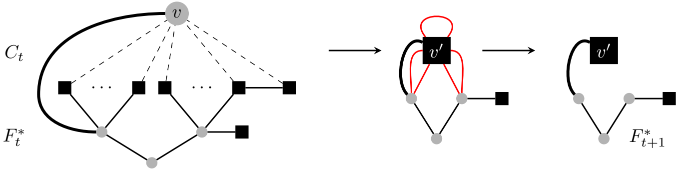
Overview of algorithm

Phase 1: We contract the star with the best ratio until the number of terminals ($|\mathrm{R}|$) is small ($\mathcal{O}(p^2/\varepsilon^4)$)
Phase 2: Run an FPT algorithm for parameterization by $|R|$.



Analysis

- For analysis we start with an optimal solution T^*.
- We maintain a tree originating from T^* in the contracted instances.
- Denote the tree after t steps of the algorithm by T_t^*.

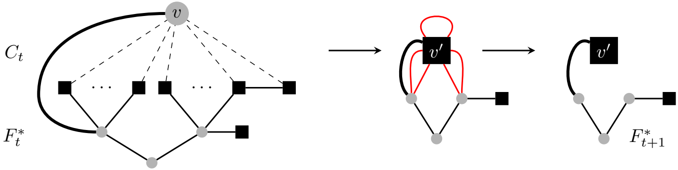


Analysis

- For analysis we start with an optimal solution T^*.
- We maintain a tree originating from T^* in the contracted instances.
- Denote the tree after t steps of the algorithm by T_t^*.

We compare the weight of each contraction with a subset of an optimal solution T^*.

- In each step we compare weight of $w(C_t)$ with $w(D_t)$.
- D_t is a feedback edge set of $T_t^* \,/\, C_t$

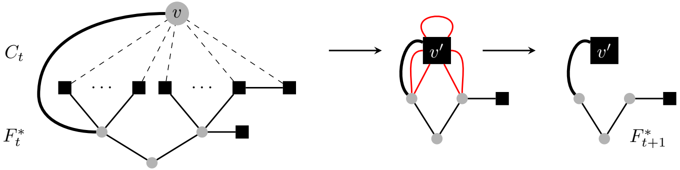


Analysis

- For analysis we start with an optimal solution T^*.
- We maintain a tree originating from T^* in the contracted instances.
- Denote the tree after t steps of the algorithm by T_t^*.

We compare the weight of each contraction with a subset of an optimal solution T^*.

- In each step we compare weight of $w(C_t)$ with $w(D_t)$.
- D_t is a feedback edge set of T_t^* / C_t



Analysis

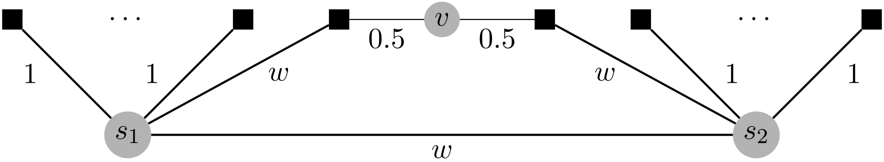
- If $w(C_t) \leq (1 + \varepsilon)w(D_t)$ for every t our algorithm is $(1 + \varepsilon)$-approximating.

Analysis

- If $w(C_t) \leq (1+\varepsilon)w(D_t)$ for every t our algorithm is $(1+\varepsilon)$-approximating.
- Unfortunately, this is not always the case. We call such contractions bad and other contraction good.

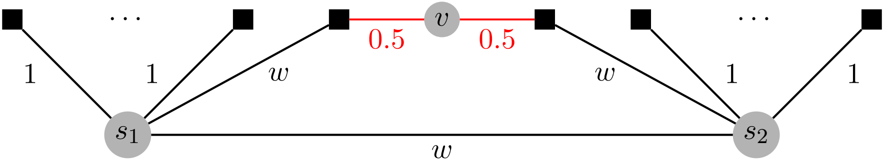
Analysis

- If $w(C_t) \leq (1+\varepsilon)w(D_t)$ for every t our algorithm is $(1+\varepsilon)$-approximating.
- Unfortunately, this is not always the case. We call such contractions bad and other contraction good.



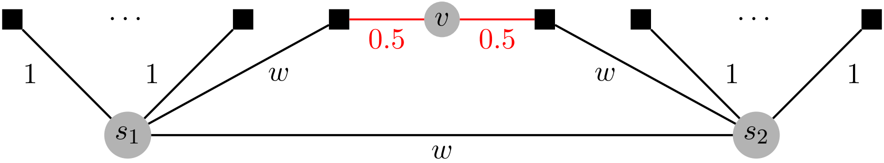
Analysis

- If $w(C_t) \leq (1+\varepsilon)w(D_t)$ for every t our algorithm is $(1+\varepsilon)$-approximating.
- Unfortunately, this is not always the case. We call such contractions bad and other contraction good.



Analysis

- If $w(C_t) \leq (1+\varepsilon)w(D_t)$ for every t our algorithm is $(1+\varepsilon)$-approximating.
- Unfortunately, this is not always the case. We call such contractions bad and other contraction good.



Proposition

If a star C contains at least $(1+\varepsilon)p/\varepsilon$ terminals then contraction of C is good.

Bounding Bad Contractions – Key Technical Idea – Sketch

- We rescale weights to be ≥ 1.

Bounding Bad Contractions – Key Technical Idea – Sketch

- We rescale weights to be ≥ 1.
- We bound the number of bad contractions in a single interval $\big((1+\delta)^j, (1+\delta)^{j+1}\big)$ based on the ratio of the contracted star.

Bounding Bad Contractions – Key Technical Idea – Sketch

- We rescale weights to be ≥ 1.
- We bound the number of bad contractions in a single interval $\left((1+\delta)^j, (1+\delta)^{j+1}\right)$ based on the ratio of the contracted star.
- Now, crucial is to obtain a lower bound on the optimum based on the largest j s.t. there was a contraction in j-th interval, or

Bounding Bad Contractions – Key Technical Idea – Sketch

- We rescale weights to be ≥ 1.
- We bound the number of bad contractions in a single interval $\big((1+\delta)^j, (1+\delta)^{j+1}\big)$ based on the ratio of the contracted star.
- Now, crucial is to obtain a lower bound on the optimum based on the largest j s.t. there was a contraction in j-th interval, or
- otherwise to prove that the number of terminals is bounded in therms of p and ε.

Kernelization – Our Results

Definition (PSAKS)

Polynomial Size Approximate Kernelization Scheme for minimization problem is defined as follows: For all $\varepsilon > 0$ there exists an algorithm that runs in polynomial time ($n^{\mathrm{O}(1)}$) s.t. it returns an $(1+\varepsilon)$-approximate kernel of size $\mathrm{O}(p^{f(\varepsilon)})$.

Kernelization – Our Results

Theorem

There is a PSAKS for the Weighted Undirected Steiner Tree problem. It computes an $(1+\varepsilon)$-approximate kernel

$$\text{of size } (p/\varepsilon)^{2^{O(1/\varepsilon)}}.$$

Kernelization – Our Results

Theorem

There is a PSAKS for the Weighted Undirected Steiner Tree problem. It computes an $(1+\varepsilon)$-approximate kernel

$$\text{of size } (p/\varepsilon)^{2^{O(1/\varepsilon)}}.$$

PSAKS	Undirected	Directed
Unweighted		noPSAKS [DFKMTV]
Weighted	PSAKS	$f(p)$-approx. hard

Lossy Kernelization

- The phase 1 leaves us with an instance with $\mathcal{O}(p^2/\varepsilon^4)$ terminals, but the instance size may still be large (unbounded in terms of p and ε).

Lossy Kernelization

- The phase 1 leaves us with an instance with $\mathcal{O}(p^2/\varepsilon^4)$ terminals, but the instance size may still be large (unbounded in terms of p and ε).
- There exists a PSAKS for parameterization by $|R|$, returning kernel of size $|R|^{2^{\mathcal{O}(1/\varepsilon)}}$ [Lokshtanov et al. 17].

Lossy Kernelization

- The phase 1 leaves us with an instance with $\mathcal{O}(p^2/\varepsilon^4)$ terminals, but the instance size may still be large (unbounded in terms of p and ε).
- There exists a PSAKS for parameterization by $|R|$, returning kernel of size $|R|^{2^{\mathcal{O}(1/\varepsilon)}}$ [Lokshtanov et al. 17].
- Combining these two we get a polynomial size $(1+\varepsilon)$-approximate kernel

  of size$(p/\varepsilon)^{2^{\mathcal{O}(1/\varepsilon)}}$.

Generalizations of Steiner Tree

Weighted Steiner Forest

Input: A graph $G = (V, E)$, a list of terminal pairs $\{s_1, t_1\}, \ldots, \{s_k, t_k\}$, and a weight function $w\colon E \to \mathbb{R}^+$.

Task: Find a forest $F \subseteq E$ of smallest weight such that s_i and t_i is in the same component of F for every i.

We extend our results to Steiner Forest while parameterized by the number of components of the optimal solution in addition to p.

Generalizations of Steiner Tree

Weighted Steiner Forest

Input: A graph $G = (V, E)$, a list of terminal pairs $\{s_1, t_1\}, \ldots, \{s_k, t_k\}$, and a weight function $w\colon E \to \mathbb{R}^+$.

Task: Find a forest $F \subseteq E$ of smallest weight such that s_i and t_i is in the same component of F for every i.

In contrast there is no FPT algorithm w.r.t. to number of Terminals in the optimal solution and it is APX-hard even without Steiner vertices.

Generalizations of Steiner Tree

Weighted Steiner Forest

Input: A graph $G = (V, E)$, a list of terminal pairs $\{s_1, t_1\}, \ldots, \{s_k, t_k\}$, and a weight function $w \colon E \to \mathbb{R}^+$.

Task: Find a forest $F \subseteq E$ of smallest weight such that s_i and t_i is in the same component of F for every i.

In contrast there is no FPT algorithm w.r.t. to number of Terminals in the optimal solution and it is APX-hard even without Steiner vertices.

We extend our results to Steiner Forest while parameterized by the number of components of the optimal solution in addition to p.

Open problems

PSAKS	Undirected	Directed
Unweighted		noPSAKS [DFKMTV]
Weighted	PSAKS	$f(p)$-approx. hard

Open problems

Definition (EPSAKS)

Efficient Polynomial Size Approximate Kernelization Scheme for minimization problem stands for: For all $\varepsilon > 0$ there exists an algorithm that runs in polynomial time ($n^{\mathrm{O}(1)}$) such that it returns an $(1+\varepsilon)$-approximate kernel of size $f(\varepsilon) \cdot p^{\mathrm{O}(1)}$.

Open problems

EPSAKS	Undirected	Directed
Unweighted	unknown	noPSAKS [DFKMTV]
Weighted	unknown	$f(p)$-approx. hard

Open problems

EPSAKS	Undirected	Directed
Unweighted	unknown	noPSAKS [DFKMTV]
Weighted	unknown	$f(p)$-approx. hard

Runtime lower-bound

What is the best runtime dependence on p and ε under Exponential Time Hypothesis?

Open problems

EPSAKS	Undirected	Directed
Unweighted	unknown	noPSAKS [DFKMTV]
Weighted	unknown	$f(p)$-approx. hard

Runtime lower-bound

What is the best runtime dependence on p and ε under Exponential Time Hypothesis?

Thank you for your attention!