A simple quadratic kernel for Token Jumping

Joint work with: Moritz Mühlenthaler and Daniel W. Cranston

Benjamin Peyrille

Université Grenoble Alpes, G-SCOP

January 6th 2025

Independent set reconfiguration

Let: $G = (V, E)$ be a simple graph,
I, J be two independent sets of V of identical sizes.
We represent vertices of I as tokens ○ and vertices of J with targets ⊕.

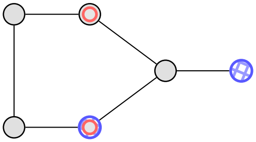
We want to **move** I to J iteratively, preserving the independent set property.

Independent set reconfiguration

Let: $G = (V, E)$ be a simple graph,
I, J be two independent sets of V of identical sizes.
We represent vertices of I as tokens and vertices of J with targets.

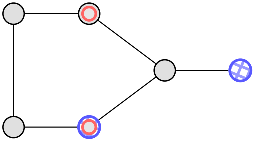
We want to **move** I to J iteratively, preserving the independent set property.

Token Sliding



Slide along edges

Token Jumping



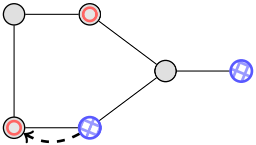
Jump anywhere

Independent set reconfiguration

Let: $G = (V, E)$ be a simple graph,
I, J be two independent sets of V of identical sizes.
We represent vertices of I as tokens and vertices of J with targets.

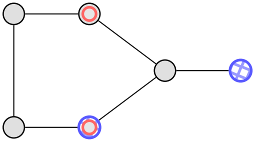
We want to **move** I to J iteratively, preserving the independent set property.

Token Sliding



Slide along edges

Token Jumping



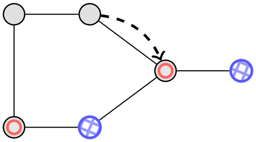
Jump anywhere

Independent set reconfiguration

Let: $G = (V, E)$ be a simple graph,
I, J be two independent sets of V of identical sizes.
We represent vertices of I as tokens and vertices of J with targets.

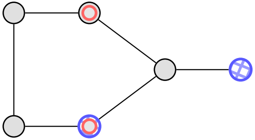
We want to **move** I to J iteratively, preserving the independent set property.

<u>Token Sliding</u>



Slide along edges

<u>Token Jumping</u>



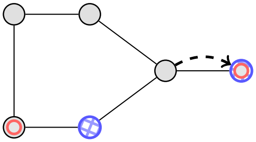
Jump anywhere

Independent set reconfiguration

Let: $G = (V, E)$ be a simple graph,
I, J be two independent sets of V of identical sizes.
We represent vertices of I as tokens and vertices of J with targets.

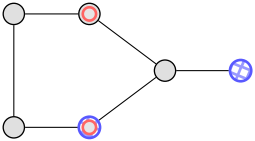
We want to **move** I to J iteratively, preserving the independent set property.

Token Sliding



Slide along edges

Token Jumping



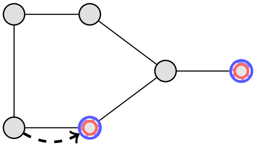
Jump anywhere

Independent set reconfiguration

Let: $G = (V, E)$ be a simple graph,
I, J be two independent sets of V of identical sizes.
We represent vertices of I as tokens and vertices of J with targets .

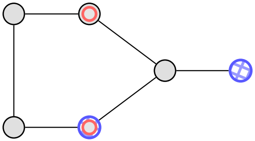
We want to **move** I to J iteratively, preserving the independent set property.

Token Sliding



Slide along edges

Token Jumping



Jump anywhere

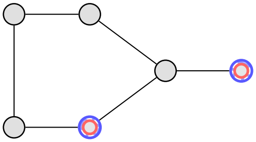
Independent set reconfiguration

Let: $G = (V, E)$ be a simple graph,
I, J be two independent sets of V of identical sizes.

We represent vertices of I as tokens and vertices of J with targets .

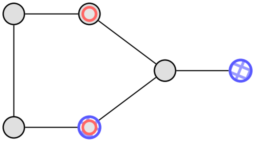
We want to **move** I to J iteratively, preserving the independent set property.

Token Sliding



Slide along edges

Token Jumping



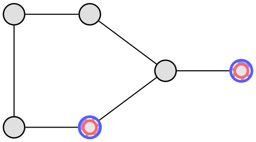
Jump anywhere

Independent set reconfiguration

Let: $G = (V, E)$ be a simple graph,
I, J be two independent sets of V of identical sizes.
We represent vertices of I as tokens and vertices of J with targets.

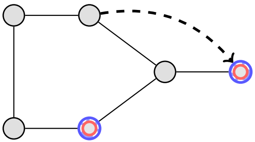
We want to **move** I to J iteratively, preserving the independent set property.

Token Sliding



Slide along edges

Token Jumping



Jump anywhere

Independent set reconfiguration

Let: $G = (V, E)$ be a simple graph,
I, J be two independent sets of V of identical sizes.

We represent vertices of I as tokens and vertices of J with targets .

We want to **move** I to J iteratively, preserving the independent set property.

<u>ISR Reachability - Token Jumping</u>

Input: A simple graph $G = (V, E)$, two independent sets I and J of G of same size.

Output: Yes if we can iteratively reach J from I using the Token Jumping rule, No otherwise.

Independent set reconfiguration

Let: $G = (V, E)$ be a simple graph,
I, J be two independent sets of V of identical sizes.

We represent vertices of I as tokens ○ and vertices of J with targets ⊕.

We want to **move** I to J iteratively, preserving the independent set property.

Token Jumping

Input: A simple graph $G = (V, E)$, two independent sets I and J of G of same size.

Output: YES if we can iteratively reach J from I using the Token Jumping rule, NO otherwise.

Hardness

Hardness result (van der Zanden, 2015)

Token Jumping is PSPACE-complete even for subcubic graphs of bounded bandwidth.

Hardness

Hardness result (van der Zanden, 2015)

Token Jumping is PSPACE-complete even for subcubic graphs of bounded bandwidth.

A problem is **fixed-parameter tractable** (FPT) for some input **parameter** k if there exists an algorithm that solves it in time $O(f(k) \cdot \text{poly}(n))$ where f is an arbitrary computable function and n is the size of the instance.

Hardness

Hardness result (van der Zanden, 2015)

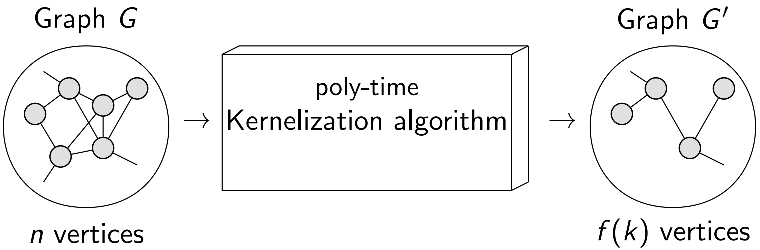
TOKEN JUMPING is PSPACE-complete even for subcubic graphs of bounded bandwidth.

A problem is **fixed-parameter tractable** (FPT) for some input **parameter** k if there exists an algorithm that solves it in time $O(f(k) \cdot \text{poly}(n))$ where f is an arbitrary computable function and n is the size of the instance.

Parameterized hardness result (Mouawad, 2017)

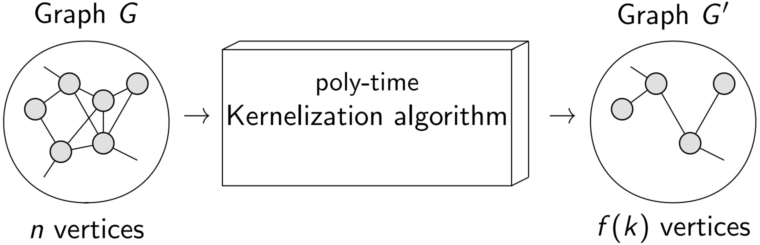
TOKEN JUMPING is $W[1]$-hard (not FPT) when only parameterized by the number of tokens k.

Positive results: known kernels



Kernelization $\implies$ FPT (bruteforce on $f(k)$ vertices)
If the function f is polynomial, we say the problem admits a **polynomial kernel**.

Positive results: known kernels



Kernelization $\implies$ FPT (bruteforce on $f(k)$ vertices)
If the function f is polynomial, we say the problem admits a **polynomial kernel**.

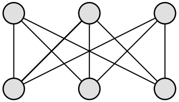
- FPT on planar graphs and $K_{3,t}$-free graphs (Ito et al, 2014).
- Polynomial kernel for $K_{t,t}$-free graphs (Bousquet et al, 2017).
- Polynomial kernel on graphs of bounded degeneracy (Lokshtanov et al. 2018).

Surfaces

Let G be a simple graph and let g be its **genus**, that is, the minimal integer such that G has a crossing-free drawing on an orientable surface of genus g.

Surfaces

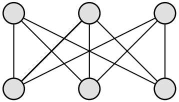
Let G be a simple graph and let g be its **genus**, that is, the minimal integer such that G has a crossing-free drawing on an orientable surface of genus g.



$K_{3,3}$ is not planar ($g \neq 0$)

Surfaces

Let G be a simple graph and let g be its **genus**, that is, the minimal integer such that G has a crossing-free drawing on an orientable surface of genus g.



$K_{3,3}$ is not planar ($g \neq 0$)

$K_{3,3}$ embedded on the torus ($g = 1$)

In a nutshell, the genus g of a graph G is
the minimum number of handles required to draw G on a mug.

Surfaces

Let G be a simple graph and let g be its **genus**, that is, the minimal integer such that G has a crossing-free drawing on an orientable surface of genus g.

Main result (Cranston, Mühlenthaler, P., 2024+)

Token Jumping parameterized by the genus g of the input graph and the number of tokens k admits a kernel of size $O((g+k)^2)$.
Furthermore, this kernel does not require knowledge of the genus.

Surfaces

Let G be a simple graph and let g be its **genus**, that is, the minimal integer such that G has a crossing-free drawing on an orientable surface of genus g.

Main result (Cranston, Mühlenthaler, P., 2024+)

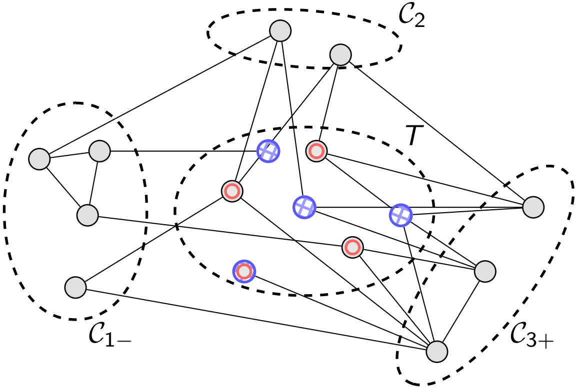
TOKEN JUMPING parameterized by the genus g of the input graph and the number of tokens k admits a kernel of size $O((g+k)^2)$.
Furthermore, this kernel does not require knowledge of the genus.

Positive kernelization results applied on graphs on surfaces:

Classes of graphs	Kernel size	For genus g
$K_{3,t}$-free (Ito et al, 14)	Ramsey$((2t+1)k, t+3)$	Ramsey$((8g+7)k, 4g+6)$
$K_{t,t}$-free (Bousquet et al, 17)	$O(f(t)\cdot k^{t\cdot 3^t})$	$O(h(g)\cdot k^{(4g+3)\cdot 3^{4g+3}})$
d-degenerate (Lokshtanov et al, 18)	$(2d+1)(2d+1)!(2k-1)^{2d+1}$	$(2H(g)-1)(2H(g)-1)!(2k-1)^{2H(g)-1}$
all graphs (This presentation!)	$O((g+k)^2)$	-

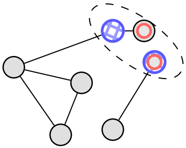
First step: Partition

- T: vertices containing the independent sets
- $\mathcal{C}_{1-}$: vertices neighboring at most one element of T
- $\mathcal{C}_2$: vertices neighboring exactly two elements of T
- $\mathcal{C}_{3+}$: vertices neighboring at least three elements of T



$\mathcal{C}_{1-}$ and $\mathcal{C}_{3+}$: easily bounded

Heawood's number $H(g) = \left\lfloor (7 + \sqrt{1 + 48g})/2 \right\rfloor$ is the maximum number of colors required to properly color a graph of genus g.
If $|\mathcal{C}_{1-}| \geq H(g) \cdot k$, the instance is YES. So we can assume

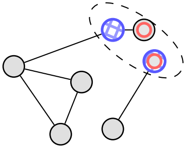


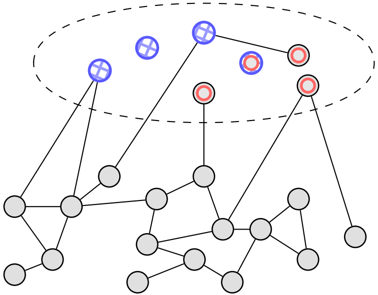
$$|\mathcal{C}_{1-}| < H(g) \cdot k.$$

$\mathcal{C}_{1-}$ and $\mathcal{C}_{3+}$: easily bounded

Heawood's number $H(g) = \left\lfloor (7 + \sqrt{1 + 48g})/2 \right\rfloor$ is the maximum number of colors required to properly color a graph of genus g.
If $|\mathcal{C}_{1-}| \geq H(g) \cdot k$, the instance is YES. So we can assume

$$|\mathcal{C}_{1-}| < H(g) \cdot k.$$





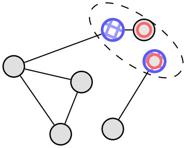
$k = 4$

$|\mathcal{C}_{1-}| = 4 \cdot 4 = 16$

Planar: $H(G) = 4$

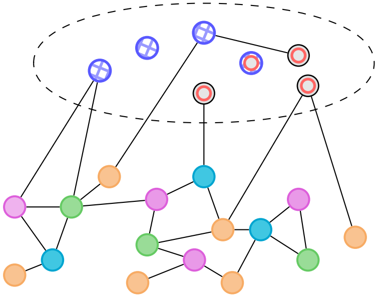
$\mathcal{C}_{1-}$ and $\mathcal{C}_{3+}$: easily bounded

Heawood's number $H(g) = \left\lfloor (7+\sqrt{1+48g})/2 \right\rfloor$ is the maximum number of colors required to properly color a graph of genus g.
If $|\mathcal{C}_{1-}| \geq H(g) \cdot k$, the instance is YES. So we can assume



$$|\mathcal{C}_{1-}| < H(g) \cdot k.$$

$|\mathcal{C}_{1-}| = 4 \cdot 4 = 16$



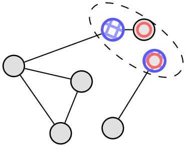
$k = 4$

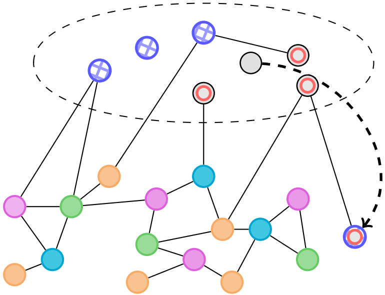
Planar: $H(G) = 4$

$\mathcal{C}_{1-}$ and $\mathcal{C}_{3+}$: easily bounded

Heawood's number $H(g) = \left\lfloor (7 + \sqrt{1 + 48g})/2 \right\rfloor$ is the maximum number of colors required to properly color a graph of genus g.
If $|\mathcal{C}_{1-}| \geq H(g) \cdot k$, the instance is YES. So we can assume

$$|\mathcal{C}_{1-}| < H(g) \cdot k.$$





$k = 4$

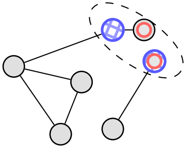
$|\mathcal{C}_{1-}| = 4 \cdot 4 = 16$

Planar: $H(G) = 4$

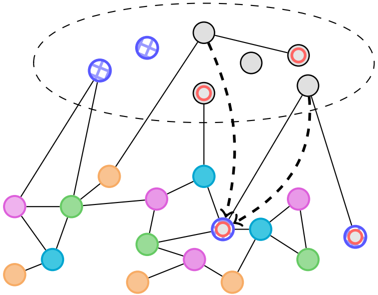
$\mathcal{C}_{1-}$ and $\mathcal{C}_{3+}$: easily bounded

Heawood's number $H(g) = \lfloor (7+\sqrt{1+48g})/2 \rfloor$ is the maximum number of colors required to properly color a graph of genus g.
If $|\mathcal{C}_{1-}| \geq H(g) \cdot k$, the instance is YES. So we can assume

$$|\mathcal{C}_{1-}| < H(g) \cdot k.$$



$|\mathcal{C}_{1-}| = 4 \cdot 4 = 16$

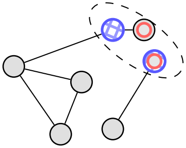


$k = 4$

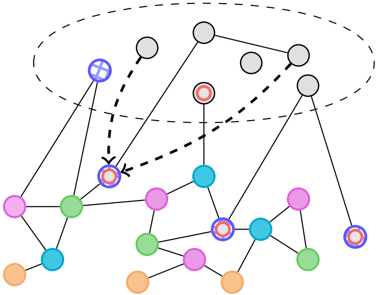
Planar: $H(G) = 4$

$\mathcal{C}_{1-}$ and $\mathcal{C}_{3+}$: easily bounded

Heawood's number $H(g) = \lfloor (7 + \sqrt{1 + 48g})/2 \rfloor$ is the maximum number of colors required to properly color a graph of genus g.
If $|\mathcal{C}_{1-}| \geq H(g) \cdot k$, the instance is YES. So we can assume



$$|\mathcal{C}_{1-}| < H(g) \cdot k.$$



$k = 4$

$|\mathcal{C}_{1-}| = 4 \cdot 4 = 16$

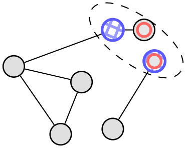
Planar: $H(G) = 4$

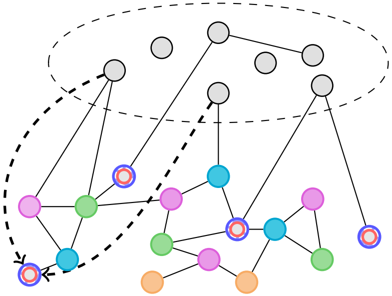
$\mathcal{C}_{1-}$ and $\mathcal{C}_{3+}$: easily bounded

Heawood's number $H(g) = \left\lfloor (7 + \sqrt{1+48g})/2 \right\rfloor$ is the maximum number of colors required to properly color a graph of genus g.

If $|\mathcal{C}_{1-}| \geq H(g) \cdot k$, the instance is YES. So we can assume

$$|\mathcal{C}_{1-}| < H(g) \cdot k.$$





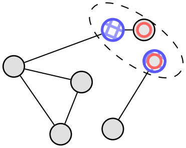
$k = 4$

$|\mathcal{C}_{1-}| = 4 \cdot 4 = 16$

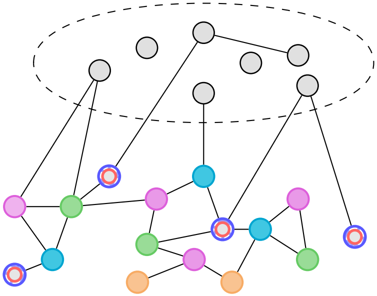
Planar: $H(G) = 4$

$\mathcal{C}_{1-}$ and $\mathcal{C}_{3+}$: easily bounded

Heawood's number $H(g) = \left\lfloor (7 + \sqrt{1 + 48g})/2 \right\rfloor$ is the maximum number of colors required to properly color a graph of genus g.
If $|\mathcal{C}_{1-}| \geq H(g) \cdot k$, the instance is YES. So we can assume



$$|\mathcal{C}_{1-}| < H(g) \cdot k.$$



$k = 4$

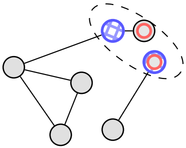
$|\mathcal{C}_{1-}| = 4 \cdot 4 = 16$

Planar: $H(G) = 4$

$\mathcal{C}_{1-}$ and $\mathcal{C}_{3+}$: easily bounded

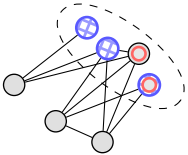
Heawood's number $H(g) = \lfloor (7 + \sqrt{1 + 48g})/2 \rfloor$ is the maximum number of colors required to properly color a graph of genus g.
If $|\mathcal{C}_{1-}| \geq H(g) \cdot k$, the instance is Yes. So we can assume

$$|\mathcal{C}_{1-}| < H(g) \cdot k.$$



Theorem (Bouchet, 1978)

A graph of genus g cannot have any $K_{3,4g+3}$ as a subgraph.



Using an auxillary graph, we can use Euler's formula to get

$$|\mathcal{C}_{3+}| \leq 16g^2 + 16gk + 8k.$$

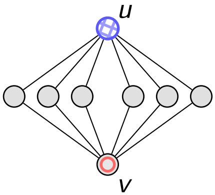
$\mathcal{C}_2$: not clear yet

Let $C_{\{u,v\}}$ be the **projection class** of $\{u,v\} \subseteq T$, that is $\{w : w \in V - T \text{ s.t } N_T(w) = \{u,v\}\}$.
Let $\{u,v\}$ such that $C_{\{u,v\}} \neq \emptyset$.

$\mathcal{C}_2$: not clear yet

Let $C_{\{u,v\}}$ be the **projection class** of $\{u, v\} \subseteq T$, that is $\{w : w \in V - T \text{ s.t } N_T(w) = \{u, v\}\}$.
Let $\{u, v\}$ such that $C_{\{u,v\}} \neq \emptyset$.
There can be an arbitrary number of vertices in $C_{\{u,v\}}$:

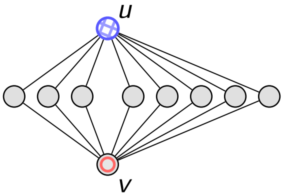
$C_{\{u,v\}}$



$\mathcal{C}_2$: not clear yet

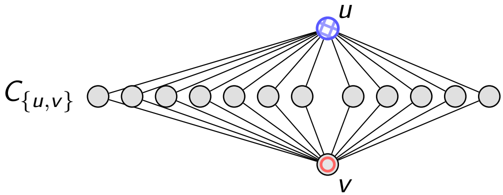
Let $C_{\{u,v\}}$ be the **projection class** of $\{u, v\} \subseteq T$, that is $\{w : w \in V - T \text{ s.t } N_T(w) = \{u, v\}\}$.
Let $\{u, v\}$ such that $C_{\{u,v\}} \neq \emptyset$.
There can be an arbitrary number of vertices in $C_{\{u,v\}}$:

$C_{\{u,v\}}$



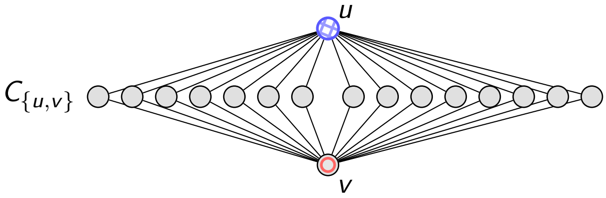
$\mathcal{C}_2$: not clear yet

Let $C_{\{u,v\}}$ be the **projection class** of $\{u, v\} \subseteq T$, that is $\{w : w \in V - T \text{ s.t } N_T(w) = \{u, v\}\}$.
Let $\{u, v\}$ such that $C_{\{u,v\}} \neq \emptyset$.
There can be an arbitrary number of vertices in $C_{\{u,v\}}$:



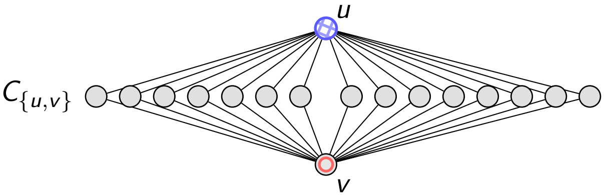
$\mathcal{C}_2$: not clear yet

Let $C_{\{u,v\}}$ be the **projection class** of $\{u, v\} \subseteq T$, that is $\{w : w \in V - T \text{ s.t } N_T(w) = \{u, v\}\}$.
Let $\{u, v\}$ such that $C_{\{u,v\}} \neq \emptyset$.
There can be an arbitrary number of vertices in $C_{\{u,v\}}$:



$\mathcal{C}_2$: not clear yet

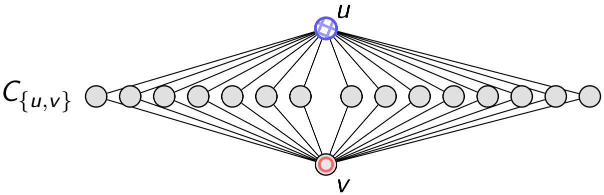
Let $C_{\{u,v\}}$ be the **projection class** of $\{u, v\} \subseteq T$, that is $\{w : w \in V - T \text{ s.t } N_T(w) = \{u, v\}\}$.
Let $\{u, v\}$ such that $C_{\{u,v\}} \neq \emptyset$.
There can be an arbitrary number of vertices in $C_{\{u,v\}}$:



Our goal: show $|\mathcal{C}_2| = O((g + k)^2)$.

$\mathcal{C}_2$: not clear yet

Let $C_{\{u,v\}}$ be the **projection class** of $\{u, v\} \subseteq T$, that is $\{w : w \in V - T \text{ s.t } N_T(w) = \{u, v\}\}$.
Let $\{u, v\}$ such that $C_{\{u,v\}} \neq \emptyset$.
There can be an arbitrary number of vertices in $C_{\{u,v\}}$:



Our goal: show $|\mathcal{C}_2| = O((g + k)^2)$.

By Euler's formula, the number of non-empty projection classes is at most $6k + 6g$.

We will show that if any $C_{\{u,v\}}$ is bigger than $8g + 4k$, the problem is solved.

Planar zones

Theorem (Malnič and Mohar, 1992)

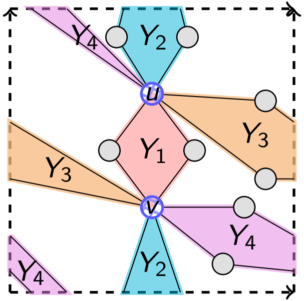
The maximum number of non-homotopic internally disjoint u, v-paths on any graph of genus g is $\max(1, 4g)$.

Planar zones

Theorem (Malnič and Mohar, 1992)

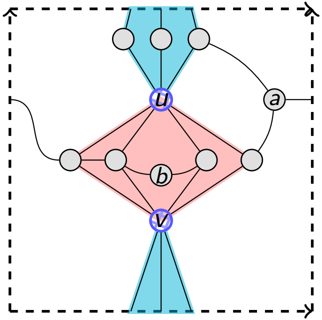
The maximum number of non-homotopic internally disjoint u, v-paths on any graph of genus g is $\max(1, 4g)$.

Hence, paths between u and v in $C_{\{u,v\}}$ divide the surface in at most $4g$ **planar zones**.



Four zones for $C_{\{u,v\}}$ on a torus.

Anatomy of the zone



Each zone has two **outer** vertices and some **inner** vertices.

Inner vertices form induced linear forests in $C_{\{u,v\}}$ whose independent sets are large and easy to find.

- Vertices outside a zone <u>cannot</u> be adjacent to inner vertices of $C_{\{u,v\}}$.
- Vertices inside a zone can only be adjacent to <u>two</u> vertices of $C_{\{u,v\}}$.

Problem solved

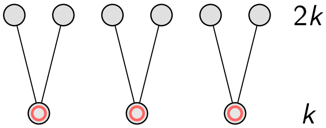
$$\begin{aligned} C_{\{u,v\}} \text{ is large } (8g + 4k) \quad &\implies \quad \geq 4k \quad \text{inner vertices} \\ &\implies \quad \geq 4k \quad \text{size linear forest} \\ &\implies \quad 2k \quad \text{size independent set } T_{\{u,v\}} \text{ in } C_{\{u,v\}} \end{aligned}$$

Recall each token of I is adjacent to at most two inner vertices of $C_{\{u,v\}}$.
We can move all tokens from I to $T_{\{u,v\}}$ if I is not frozen. We then do the same for J.

Problem solved

$$
\begin{aligned}
C_{\{u,v\}} \text{ is large } (8g+4k) \quad &\implies \quad \geq 4k \quad \text{inner vertices} \\
&\implies \quad \geq 4k \quad \text{size linear forest} \\
&\implies \quad 2k \quad \text{size independent set } T_{\{u,v\}} \text{ in } C_{\{u,v\}}
\end{aligned}
$$

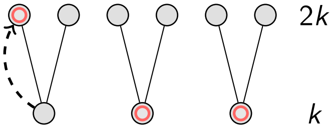
Recall each token of I is adjacent to at most two inner vertices of $C_{\{u,v\}}$.
We can move all tokens from I to $T_{\{u,v\}}$ if I is not frozen. We then do the same for J.



Problem solved

$$\begin{aligned} C_{\{u,v\}} \text{ is large } (8g+4k) \quad &\Longrightarrow \quad \geq 4k \quad \text{inner vertices} \\ &\Longrightarrow \quad \geq 4k \quad \text{size linear forest} \\ &\Longrightarrow \quad 2k \quad \text{size independent set } T_{\{u,v\}} \text{ in } C_{\{u,v\}} \end{aligned}$$

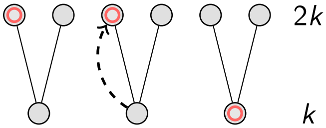
Recall each token of I is adjacent to at most two inner vertices of $C_{\{u,v\}}$.
We can move all tokens from I to $T_{\{u,v\}}$ if I is not frozen. We then do the same for J.



Problem solved

$$
\begin{aligned}
C_{\{u,v\}} \text{ is large } (8g+4k) \quad &\Longrightarrow \quad \geq 4k \quad \text{inner vertices} \\
&\Longrightarrow \quad \geq 4k \quad \text{size linear forest} \\
&\Longrightarrow \quad 2k \quad \text{size independent set } T_{\{u,v\}} \text{ in } C_{\{u,v\}}
\end{aligned}
$$

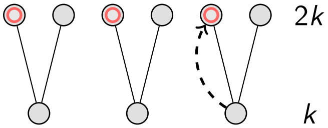
Recall each token of I is adjacent to at most two inner vertices of $C_{\{u,v\}}$.
We can move all tokens from I to $T_{\{u,v\}}$ if I is not frozen. We then do the same for J.



Problem solved

$$
\begin{aligned}
C_{\{u,v\}} \text{ is large } (8g+4k) \quad &\implies \quad \geq 4k \quad \text{inner vertices} \\
&\implies \quad \geq 4k \quad \text{size linear forest} \\
&\implies \quad 2k \quad \text{size independent set } T_{\{u,v\}} \text{ in } C_{\{u,v\}}
\end{aligned}
$$

Recall each token of I is adjacent to at most two inner vertices of $C_{\{u,v\}}$.
We can move all tokens from I to $T_{\{u,v\}}$ if I is not frozen. We then do the same for J.



Problem solved

$$
\begin{aligned}
C_{\{u,v\}} \text{ is large } (8g+4k) \quad &\implies \quad \geq 4k \quad \text{inner vertices}\\
&\implies \quad \geq 4k \quad \text{size linear forest}\\
&\implies \quad 2k \quad \text{size independent set } T_{\{u,v\}} \text{ in } C_{\{u,v\}}
\end{aligned}
$$

Recall each token of I is adjacent to at most two inner vertices of $C_{\{u,v\}}$.
We can move all tokens from I to $T_{\{u,v\}}$ if I is not frozen. We then do the same for J.

So we can assume all $C_{\{u,v\}}$ are of size at most $8g + 4k$.

Problem solved... or is it?

$$\begin{aligned} C_{\{u,v\}} \text{ is large } (8g+4k) \quad &\implies \quad \geq 4k \quad \text{inner vertices} \\ &\implies \quad \geq 4k \quad \text{size linear forest} \\ &\implies \quad 2k \quad \text{size independent set } T_{\{u,v\}} \text{ in } C_{\{u,v\}} \end{aligned}$$

Recall each token of I is adjacent to at most two inner vertices of $C_{\{u,v\}}$.
We can move all tokens from I to $T_{\{u,v\}}$ if I is not frozen. We then do the same for J.

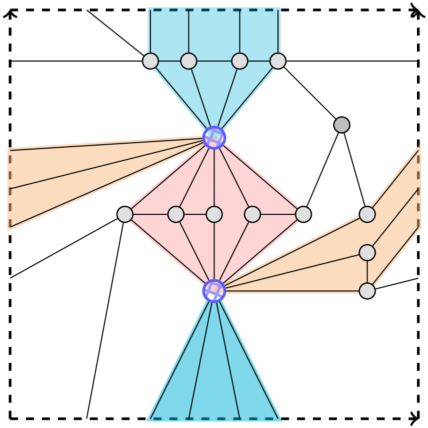
So we can assume all $C_{\{u,v\}}$ are of size at most $8g+4k$.

Problem: knowing the genus of the graph or a crossing-free drawing, is hard.

We will find that large linear forest without any information on the genus.

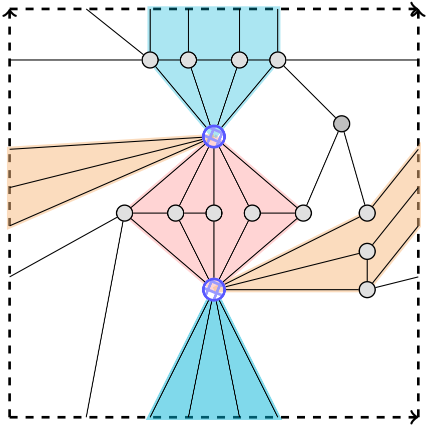
The algorithm

1 $Z := C_{\{u,v\}}$



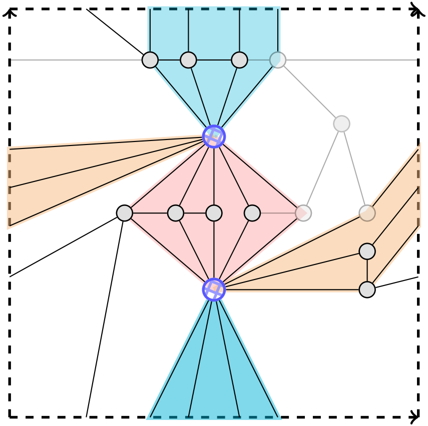
The algorithm

$Z := C_{\{u,v\}}$
for $v \in V - (C_{\{u,v\}} \cup Y)$ **do**
  if *v has at least 3 neighbors in $C_{\{u,v\}}$* **then**
    $Z \leftarrow Z - N(v)$



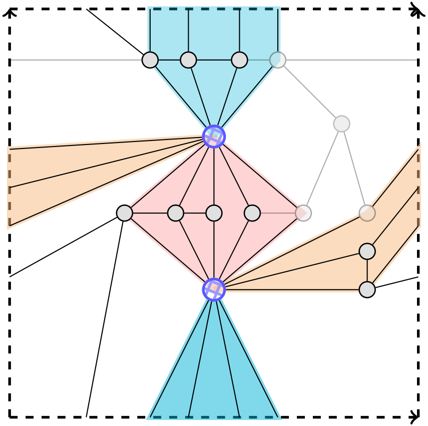
The algorithm

$Z := C_{\{u,v\}}$
for $v \in V - (C_{\{u,v\}} \cup Y)$ **do**
  if v *has at least* 3 *neighbors in* $C_{\{u,v\}}$ **then**
    $Z \leftarrow Z - N(v)$



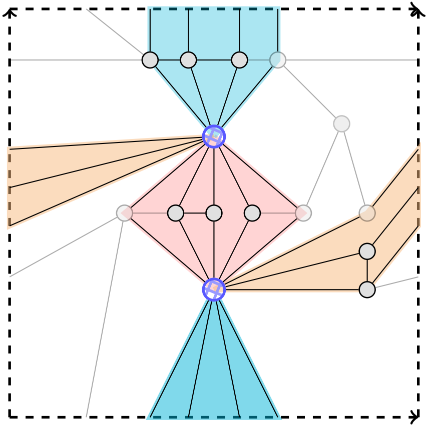
The algorithm

$Z := C_{\{u,v\}}$
for $v \in V - (C_{\{u,v\}} \cup Y)$ **do**
    if *v has at least 3 neighbors in $C_{\{u,v\}}$* **then**
        $Z \leftarrow Z - N(v)$
for $w \in Z$ **do**
    if *w has degree at least 3 in $G[Z]$* **then**
        $Z \leftarrow Z - w$



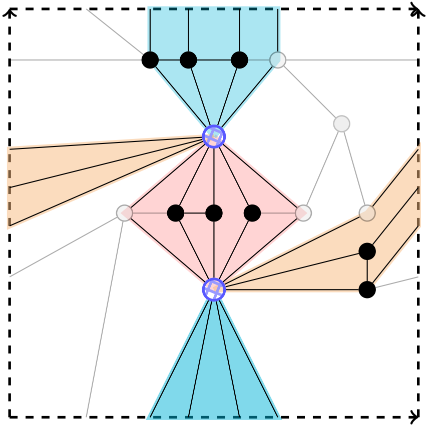
The algorithm

$Z := C_{\{u,v\}}$
for $v \in V - (C_{\{u,v\}} \cup Y)$ **do**
  if *v has at least 3 neighbors in $C_{\{u,v\}}$* **then**
    $Z \leftarrow Z - N(v)$
for $w \in Z$ **do**
  if *w has degree at least 3 in $G[Z]$* **then**
    $Z \leftarrow Z - w$



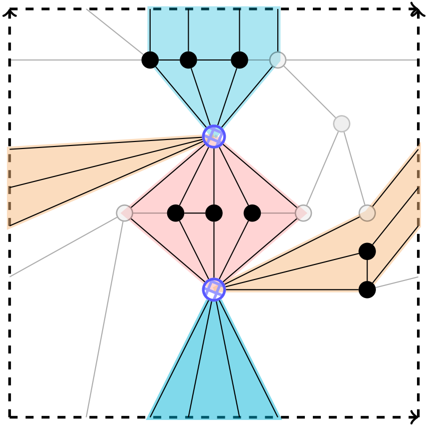
The algorithm

$Z := C_{\{u,v\}}$
for $v \in V - (C_{\{u,v\}} \cup Y)$ **do**
  if *v has at least 3 neighbors in $C_{\{u,v\}}$* **then**
    $Z \leftarrow Z - N(v)$
for $w \in Z$ **do**
  if *w has degree at least 3 in $G[Z]$* **then**
    $Z \leftarrow Z - w$
Remove arbitrarily one vertex from each cycle in $G[Z]$
return Z



The algorithm

$Z := C_{\{u,v\}}$
for $v \in V - (C_{\{u,v\}} \cup Y)$ **do**
  if *v has at least 3 neighbors in $C_{\{u,v\}}$* **then**
    $Z \leftarrow Z - N(v)$
for $w \in Z$ **do**
  if *w has degree at least 3 in $G[Z]$* **then**
    $Z \leftarrow Z - w$
Remove arbitrarily one vertex from each cycle in $G[Z]$
return Z



This procedure outputs a linear forest of size at least equal to the number of inner vertices, without any information on the genus.

Conclusion

We give a kernelization algorithm with quadratic size $O((g+k)^2)$ for Token Jumping.

Conclusion

We give a kernelization algorithm with quadratic size $O((g+k)^2)$ for Token Jumping.

Our algorithm uses very simple rules and requires no information on the genus of the input graph.

Conclusion

We give a kernelization algorithm with quadratic size $O((g+k)^2)$ for Token Jumping.

Our algorithm uses very simple rules and requires no information on the genus of the input graph.

Open question:

- Can there be a kernel of size $O(g^2 + gk + k)$ for planar graphs and for graphs in general?
- What other problems can be parameterized in such a way?

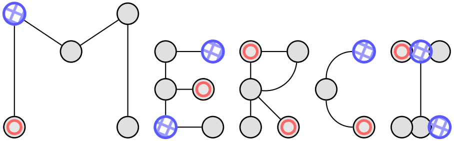
Conclusion

We give a kernelization algorithm with quadratic size $O((g+k)^2)$ for Token Jumping.

Our algorithm uses very simple rules and requires no information on the genus of the input graph.

Open question:

- Can there be a kernel of size $O(g^2 + gk + k)$ for planar graphs and for graphs in general?
- What other problems can be parameterized in such a way?



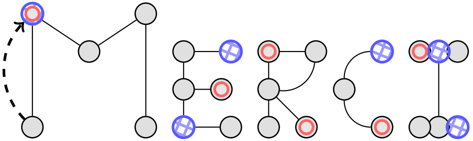
Conclusion

We give a kernelization algorithm with quadratic size $O((g+k)^2)$ for Token Jumping.

Our algorithm uses very simple rules and requires no information on the genus of the input graph.

Open question:

- Can there be a kernel of size $O(g^2 + gk + k)$ for planar graphs and for graphs in general?
- What other problems can be parameterized in such a way?



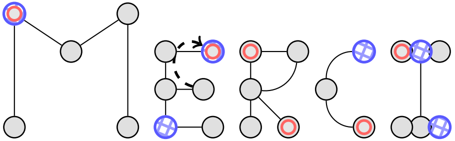
Conclusion

We give a kernelization algorithm with quadratic size $O((g+k)^2)$ for Token Jumping.

Our algorithm uses very simple rules and requires no information on the genus of the input graph.

Open question:

- Can there be a kernel of size $O(g^2 + gk + k)$ for planar graphs and for graphs in general?
- What other problems can be parameterized in such a way?



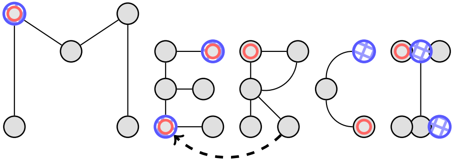
Conclusion

We give a kernelization algorithm with quadratic size $O((g+k)^2)$ for Token Jumping.

Our algorithm uses very simple rules and requires no information on the genus of the input graph.

Open question:

- Can there be a kernel of size $O(g^2 + gk + k)$ for planar graphs and for graphs in general?
- What other problems can be parameterized in such a way?



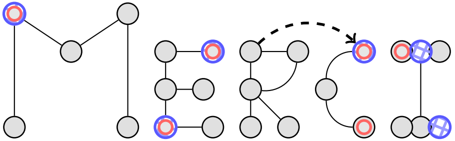
Conclusion

We give a kernelization algorithm with quadratic size $O((g+k)^2)$ for Token Jumping.

Our algorithm uses very simple rules and requires no information on the genus of the input graph.

Open question:

- Can there be a kernel of size $O(g^2 + gk + k)$ for planar graphs and for graphs in general?
- What other problems can be parameterized in such a way?



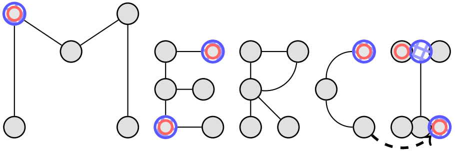
Conclusion

We give a kernelization algorithm with quadratic size $O((g+k)^2)$ for Token Jumping.

Our algorithm uses very simple rules and requires no information on the genus of the input graph.

Open question:

- Can there be a kernel of size $O(g^2 + gk + k)$ for planar graphs and for graphs in general?
- What other problems can be parameterized in such a way?



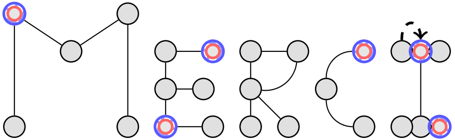
Conclusion

We give a kernelization algorithm with quadratic size $O((g+k)^2)$ for Token Jumping.

Our algorithm uses very simple rules and requires no information on the genus of the input graph.

Open question:

- Can there be a kernel of size $O(g^2 + gk + k)$ for planar graphs and for graphs in general?
- What other problems can be parameterized in such a way?



Conclusion

We give a kernelization algorithm with quadratic size $O((g+k)^2)$ for Token Jumping.

Our algorithm uses very simple rules and requires no information on the genus of the input graph.

Open question:

- Can there be a kernel of size $O(g^2 + gk + k)$ for planar graphs and for graphs in general?
- What other problems can be parameterized in such a way?