A Linear Kernel for Independent Set Reconfiguration in Planar Graphs

Nicolas Bousquet

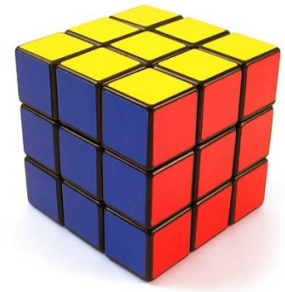
(Joint work with Daniel W. Cranston)

https://arxiv.org/abs/2506.03319

STACS'26

Reconfiguration

A one-player game is a puzzle : one player makes a series of moves, trying to accomplish some goal.

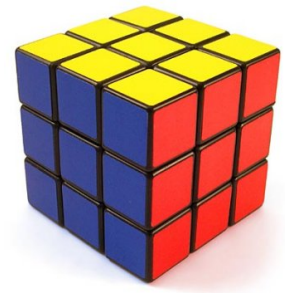


Question :

Giving my current position, can I reach a fixed target position ?

Reconfiguration

A one-player game is a puzzle : one player makes a series of moves, trying to accomplish some goal.



Question :
Giving my current position, can I reach a fixed target position ?

- Important problems in random sampling, bioinformatics, discrete geometry, games...etc... for decades.

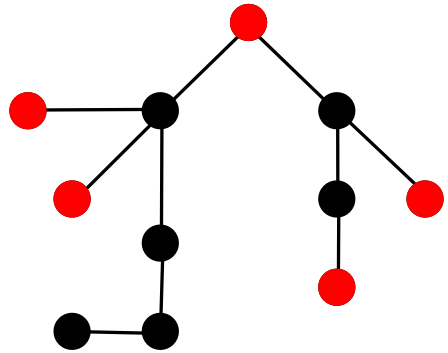
Main questions

- **Reachability problem.** Given two configurations, is it possible to transform the one into the other ?
- **Connectivity problem.** Given any pair of configurations, is it possible to transform the one into the other ?
- **Minimization.** Given two configurations, what is the length of a shortest sequence ?

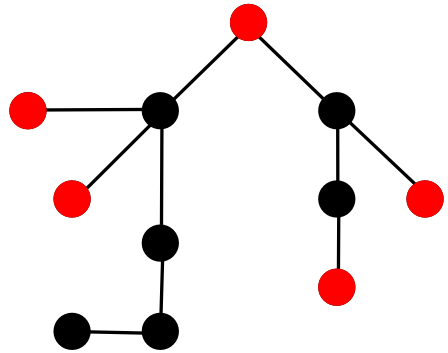
Main questions

- **Reachability problem.** Given two configurations, is it possible to transform the one into the other ?
- **Connectivity problem.** Given any pair of configurations, is it possible to transform the one into the other ?
- **Minimization.** Given two configurations, what is the length of a shortest sequence ?
- **Algorithmics.** Can we efficiently solve these questions ? (In polynomial time, FPT-time...).

Reconfiguration of independent sets

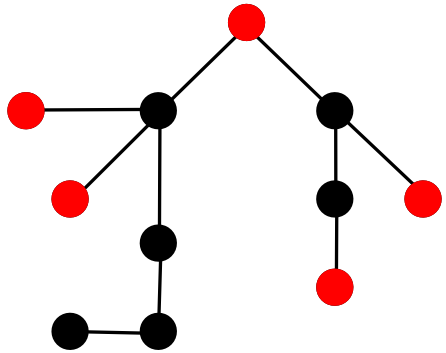


Reconfiguration of independent sets

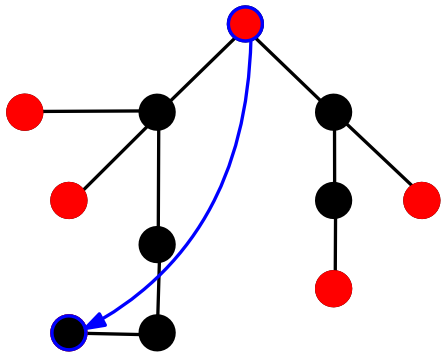


Token Jumping
Select one vertex of I and move it anywhere else.
(keeping an IS)

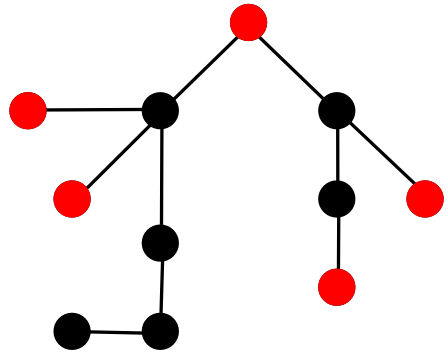
Reconfiguration of independent sets



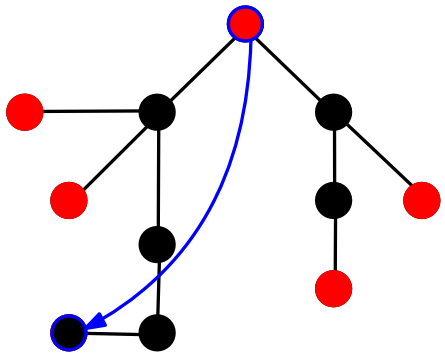
Token Jumping
Select one vertex of I and move it anywhere else.
(keeping an IS)



Reconfiguration of independent sets

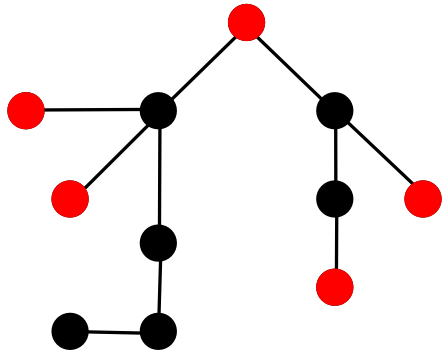


Token Jumping
Select one vertex of I and move it anywhere else.
(keeping an IS)

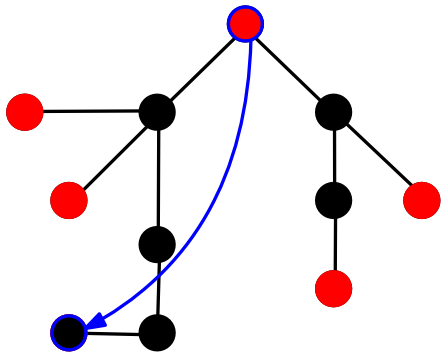


Token Sliding
Select one vertex of I and move it to an adjacent vertex.
(keeping an IS).

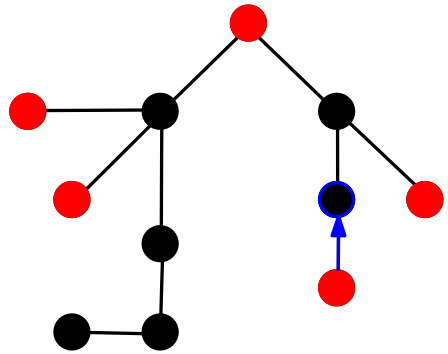
Reconfiguration of independent sets



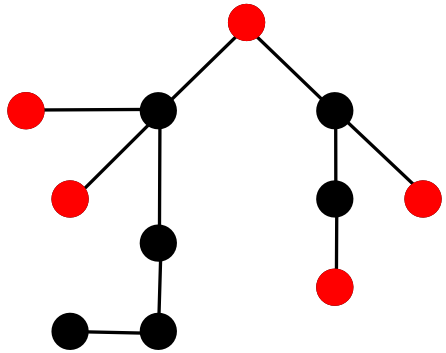
Token Jumping
Select one vertex of I and move it anywhere else.
(keeping an IS)



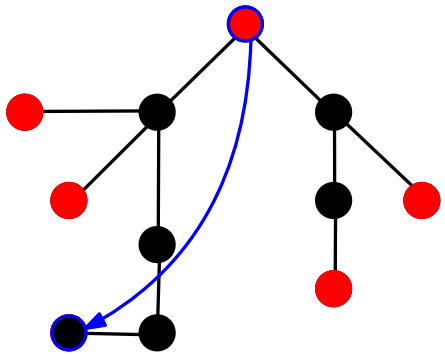
Token Sliding
Select one vertex of I and move it to an adjacent vertex.
(keeping an IS).



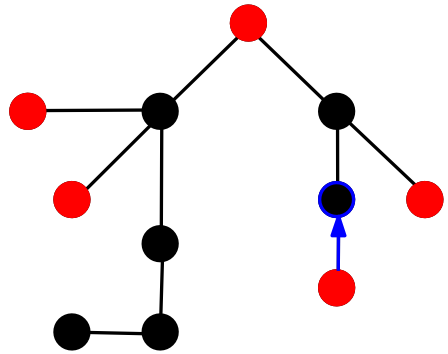
Reconfiguration of independent sets



Token Jumping
Select one vertex of I and move it anywhere else.
(keeping an IS)

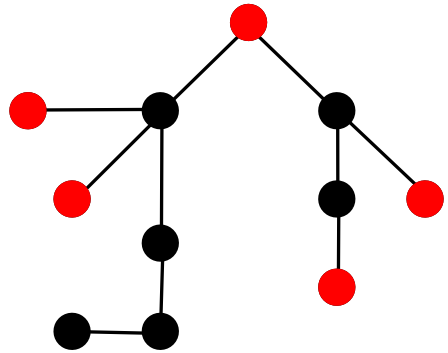


Token Sliding
Select one vertex of I and move it to an adjacent vertex.
(keeping an IS).

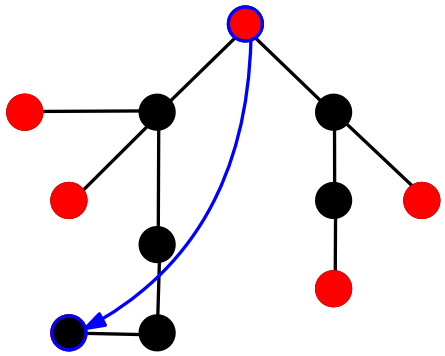


Question : What is the complexity of TS / TJ-Reachability ?

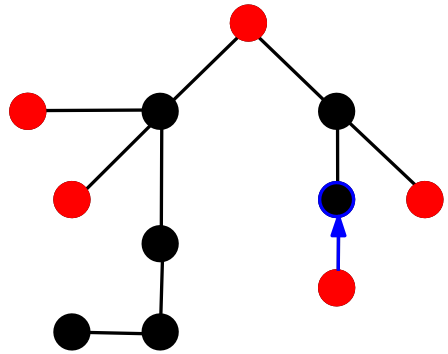
Reconfiguration of independent sets



Token Jumping
Select one vertex of I and move it anywhere else.
(keeping an IS)



Token Sliding
Select one vertex of I and move it to an adjacent vertex.
(keeping an IS).



Question : What is the complexity of TS / TJ-Reachability ?

Today : Focus on TJ.

Genesis

[Hopcroft, Schwartz, Sharir '83] Warehouseman's problem - Motion of rectangular robots in a grid.

$\Rightarrow$ PSPACE-complete (but large robots are needed !).

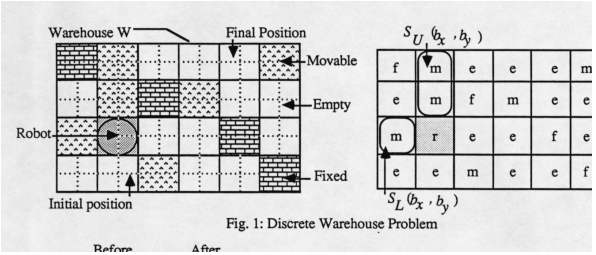


Fig. 1: Discrete Warehouse Problem

Genesis

[Hopcroft, Schwartz, Sharir '83] Warehouseman's problem - Motion of rectangular robots in a grid.

$\Rightarrow$ PSPACE-complete (but large robots are needed !).

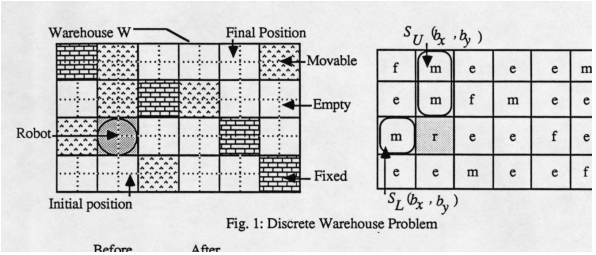


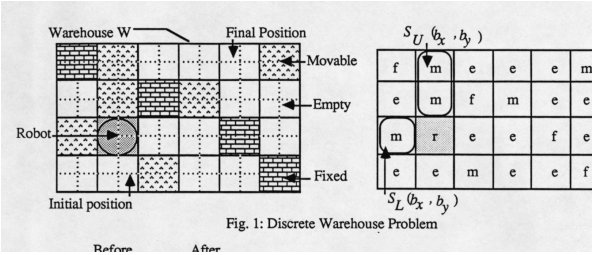
Fig. 1: Discrete Warehouse Problem

Question : What is the complexity of the Warehouseman problem for "dominos shaped" robots ?

Genesis

[Hopcroft, Schwartz, Sharir '83] Warehouseman's problem - Motion of rectangular robots in a grid.

$\Rightarrow$ PSPACE-complete (but large robots are needed !).



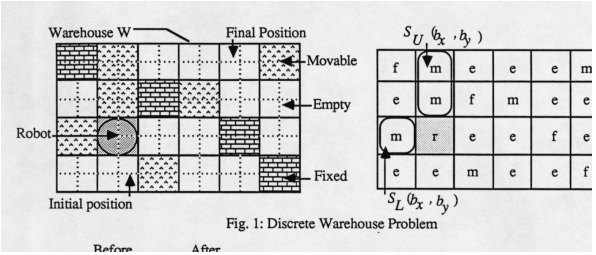
Question : What is the complexity of the Warehouseman problem for "dominos shaped" robots ?

[Flake, Baum '03] Rush hour is PSPACE-complete.

Genesis

[Hopcroft, Schwartz, Sharir '83] Warehouseman's problem - Motion of rectangular robots in a grid.

$\Rightarrow$ PSPACE-complete (but large robots are needed !).



Question : What is the complexity of the Warehouseman problem for "dominos shaped" robots ?

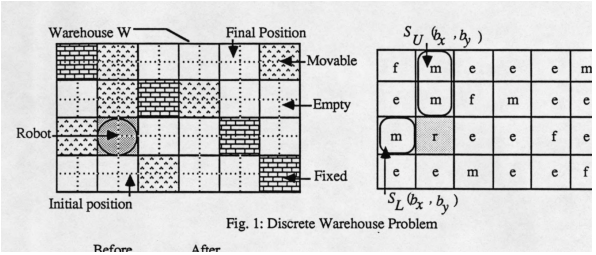
[Flake, Baum '03] Rush hour is PSPACE-complete.

[Hearn, Demaine '05] The Warehouseman's problem with dominos shaped robots is PSPACE-complete

Genesis

[Hopcroft, Schwartz, Sharir '83] Warehouseman's problem - Motion of rectangular robots in a grid.

$\Rightarrow$ PSPACE-complete (but large robots are needed !).



Question : What is the complexity of the Warehouseman problem for "dominos shaped" robots ?

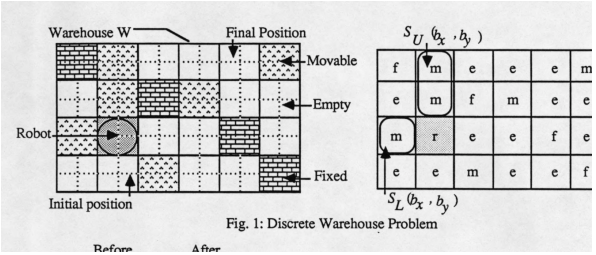
[Flake, Baum '03] Rush hour is PSPACE-complete.

[Hearn, Demaine '05] The Warehouseman's problem with dominos shaped robots is PSPACE-complete

[Hearn, Demaine '05] TJ-Reachability for Independent Set Reconfiguration is PSPACE-complete even restricted to planar graphs of maximum degree at most 3...

Genesis

[Hopcroft, Schwartz, Sharir '83] Warehouseman's problem - Motion of rectangular robots in a grid.
$\Rightarrow$ PSPACE-complete (but large robots are needed !).



Question : What is the complexity of the Warehouseman problem for "dominos shaped" robots ?

[Flake, Baum '03] Rush hour is PSPACE-complete.

[Hearn, Demaine '05] The Warehouseman's problem with dominos shaped robots is PSPACE-complete

[Hearn, Demaine '05] TJ-Reachability for Independent Set Reconfiguration is PSPACE-complete even restricted to planar graphs of maximum degree at most 3...
[Wrochna '17] ... and on bounded bandwidth graphs !

Parameterized complexity

- [Bodlaender et al. '21] XL-complete param. by k (= size of the IS)

  ($\Rightarrow$ No FPT algorithms under complexity asumptions)

Parameterized complexity

- [Bodlaender et al. '21] XL-complete param. by k (= size of the IS)
  (⇒ No FPT algorithms under complexity asumptions)
- [Ito et al. '14] FPT on planar graphs parameterized by k (and even $K_{3,\ell}$-free graphs).
  ⇔ TJ-ISR can be decided in $O(f(k) \cdot Poly(n))$.

Parameterized complexity

- [Bodlaender et al. '21] XL-complete param. by k (= size of the IS)
  (⇒ No FPT algorithms under complexity asumptions)
- [Ito et al. '14] FPT on planar graphs parameterized by k (and even $K_{3,\ell}$-free graphs).
  ⇔ TJ-ISR can be decided in $O(f(k) \cdot Poly(n))$.
- [B., Mary, Parrreau '17] FPT on $K_{\ell,\ell}$-free graphs.

Parameterized complexity

- [Bodlaender et al. '21] XL-complete param. by k (= size of the IS)
  (⇒ No FPT algorithms under complexity asumptions)
- [Ito et al. '14] FPT on planar graphs parameterized by k (and even $K_{3,\ell}$-free graphs).
  ⇔ TJ-ISR can be decided in $O(f(k) \cdot Poly(n))$.
- [B., Mary, Parrreau '17] FPT on $K_{\ell,\ell}$-free graphs.
- [Cranston, Muelhenthaler, Peyrille '24+] Quadratic kernel on planar graphs.
  ⇔ Equivalent instance of size $O(k^2)$.

Parameterized complexity

- [Bodlaender et al. '21] XL-complete param. by k (= size of the IS)
  (⇒ No FPT algorithms under complexity asumptions)
- [Ito et al. '14] FPT on planar graphs parameterized by k (and even $K_{3,\ell}$-free graphs).
  ⇔ TJ-ISR can be decided in $O(f(k) \cdot Poly(n))$.
- [B., Mary, Parrreau '17] FPT on $K_{\ell,\ell}$-free graphs.
- [Cranston, Muelhenthaler, Peyrille '24+] Quadratic kernel on planar graphs.
  ⇔ Equivalent instance of size $O(k^2)$.

Question : Linear kernel ?

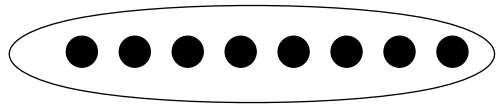
Parameterized complexity

- [Bodlaender et al. '21] XL-complete param. by k (= size of the IS)
  (⇒ No FPT algorithms under complexity asumptions)
- [Ito et al. '14] FPT on planar graphs parameterized by k (and even $K_{3,\ell}$-free graphs).
  ⇔ TJ-ISR can be decided in $O(f(k) \cdot Poly(n))$.
- [B., Mary, Parrreau '17] FPT on $K_{\ell,\ell}$-free graphs.
- [Cranston, Muelhenthaler, Peyrille '24+] Quadratic kernel on planar graphs.
  ⇔ Equivalent instance of size $O(k^2)$.

Question : Linear kernel ?

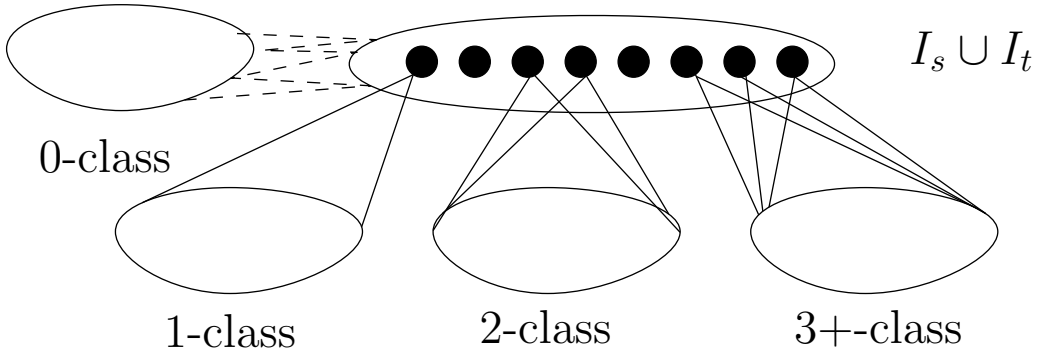
→ **This paper :** Positive answer !

Warm up : A quadratic kernel



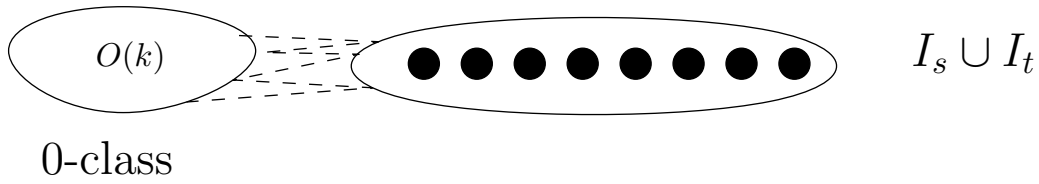
- Consider the neighborhood types in $I_s \cup I_t$ (as in [Ito et al. '14])
- Bounded expansion $\Rightarrow$ $O(k)$ types
  $\Rightarrow$ Partition of $V \setminus (I_s \cup I_t)$ into $O(k)$ classes

Warm up : A quadratic kernel



- Consider the neighborhood types in $I_s \cup I_t$ (as in [Ito et al. '14])
- Bounded expansion $\Rightarrow$ $O(k)$ types
  $\Rightarrow$ Partition of $V \setminus (I_s \cup I_t)$ into $O(k)$ classes

Warm up : A quadratic kernel

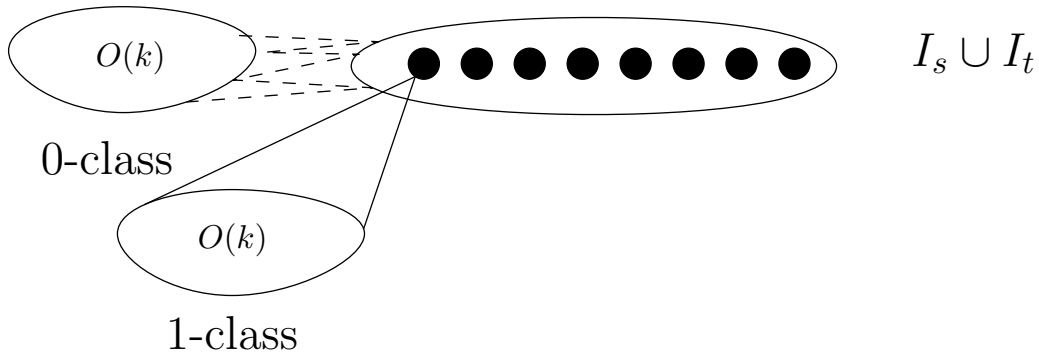


- Consider the neighborhood types in $I_s \cup I_t$ (as in [Ito et al. '14])
- Bounded expansion $\Rightarrow$ $O(k)$ types
  $\Rightarrow$ Partition of $V \setminus (I_s \cup I_t)$ into $O(k)$ classes

Roadmap :

- The 0-class has size $O(k)$

Warm up : A quadratic kernel

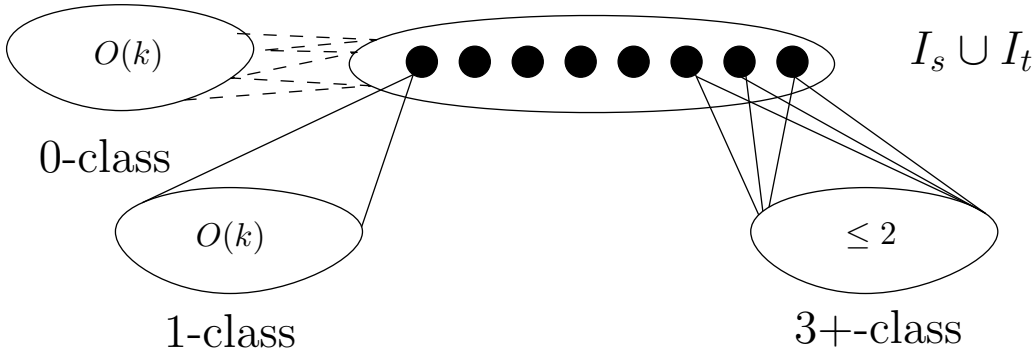


- Consider the neighborhood types in $I_s \cup I_t$ (as in [Ito et al. '14])
- Bounded expansion $\Rightarrow$ $O(k)$ types
  $\Rightarrow$ Partition of $V \setminus (I_s \cup I_t)$ into $O(k)$ classes

Roadmap :

- The 0-class has size $O(k)$ as well as the union of the 1-classes

Warm up : A quadratic kernel

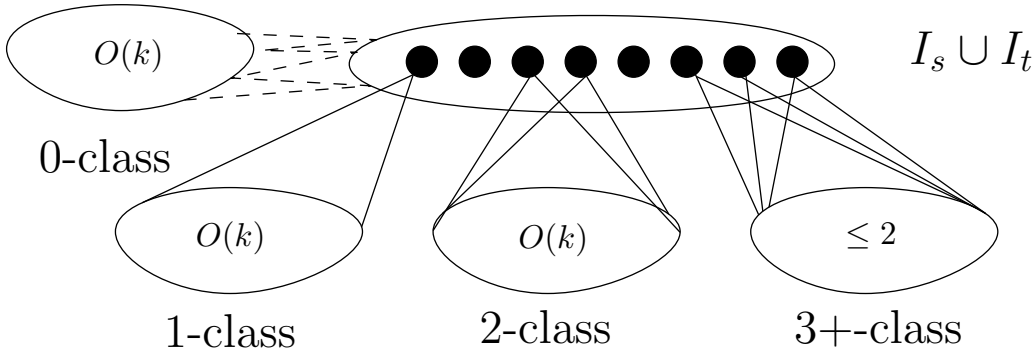


- Consider the neighborhood types in $I_s \cup I_t$ (as in [Ito et al. '14])
- Bounded expansion $\Rightarrow$ $O(k)$ types
  $\Rightarrow$ Partition of $V \setminus (I_s \cup I_t)$ into $O(k)$ classes

Roadmap :

- The 0-class has size $O(k)$ as well as the union of the 1-classes
- 3+-classes have constant size (otherwise G would contain a $K_{3,3}$)

Warm up : A quadratic kernel

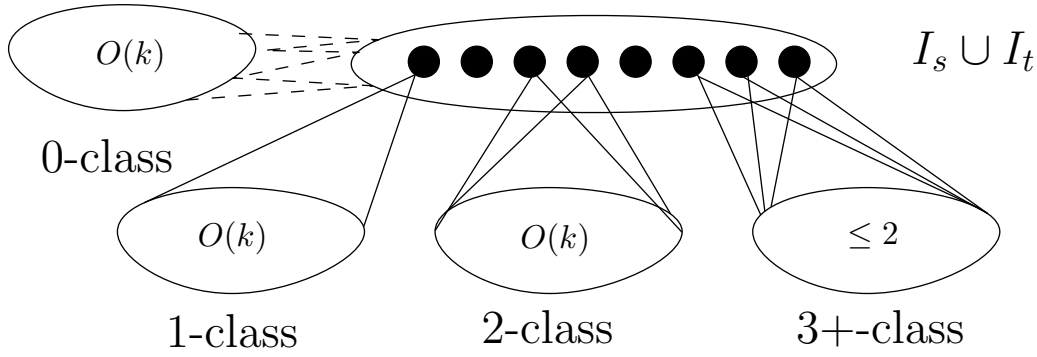


- Consider the neighborhood types in $I_s \cup I_t$ (as in [Ito et al. '14])
- Bounded expansion $\Rightarrow$ $O(k)$ types
  $\Rightarrow$ Partition of $V \setminus (I_s \cup I_t)$ into $O(k)$ classes

Roadmap :

- The 0-class has size $O(k)$ as well as the union of the 1-classes
- 3+-classes have constant size (otherwise G would contain a $K_{3,3}$)
- **Each** 2-class has linear size.

Warm up : A quadratic kernel



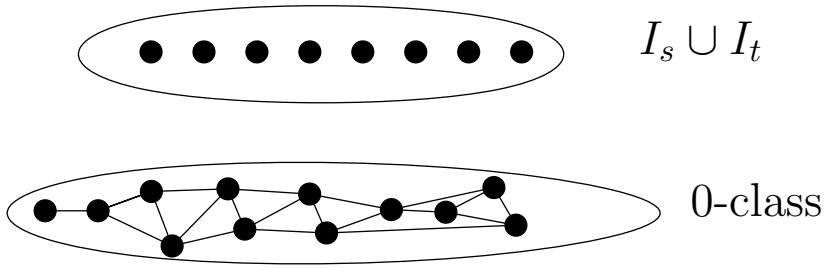
- Consider the neighborhood types in $I_s \cup I_t$ (as in [Ito et al. '14])
- Bounded expansion $\Rightarrow$ $O(k)$ types
  $\Rightarrow$ Partition of $V \setminus (I_s \cup I_t)$ into $O(k)$ classes

Roadmap :

- The 0-class has size $O(k)$ as well as the union of the 1-classes
- 3+-classes have constant size (otherwise G would contain a $K_{3,3}$)
- **Each** 2-class has linear size.

Conclusion : $O(k)$ classes of size $O(k) \Rightarrow O(k^2)$ kernel.

Reducing 0-classes

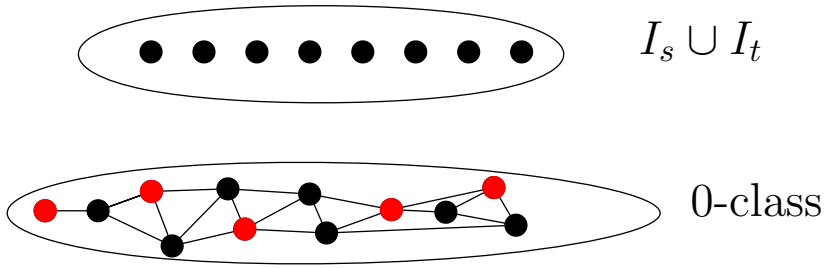


Claim :

Size of the 0-class $\geq 4k \Rightarrow$ Project I_s and I_t on the 0-class

$\Rightarrow$ yes-instance

Reducing 0-classes



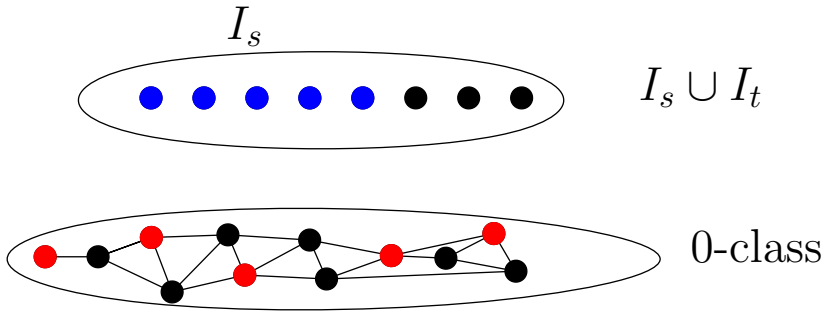
Claim :

Size of the 0-class $\geq 4k \Rightarrow$ Project I_s and I_t on the 0-class
$\Rightarrow$ yes-instance

Sketch of the proof

- Select an IS of size k arbitrarily in the 0-class

Reducing 0-classes



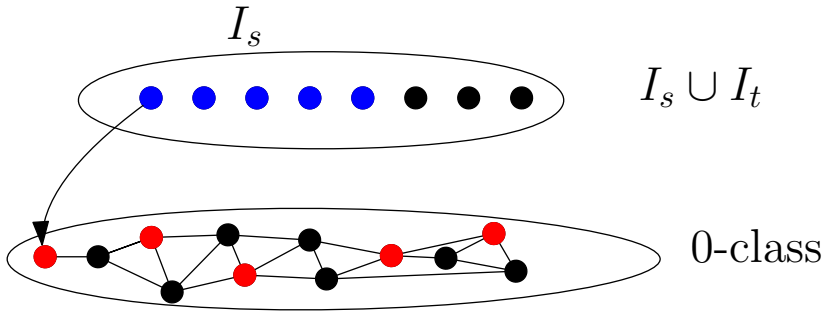
Claim :

Size of the 0-class $\geq 4k \Rightarrow$ Project I_s and I_t on the 0-class
$\Rightarrow$ yes-instance

Sketch of the proof

- Select an IS of size k arbitrarily in the 0-class

Reducing 0-classes



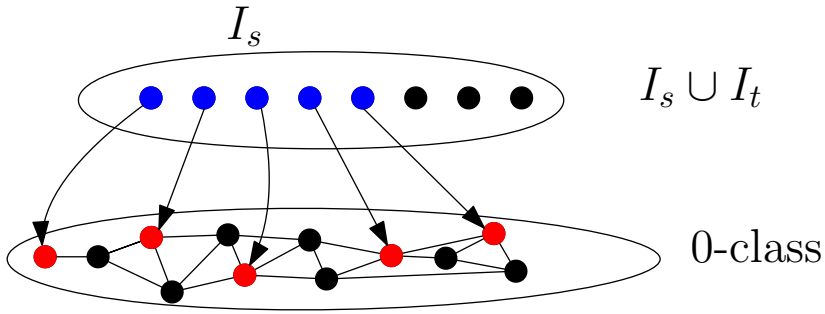
Claim :

Size of the 0-class $\geq 4k \Rightarrow$ Project I_s and I_t on the 0-class
$\Rightarrow$ yes-instance

Sketch of the proof

- Select an IS of size k arbitrarily in the 0-class
- Project one by one the vertices of I_s (resp. I_t) on •

Reducing 0-classes



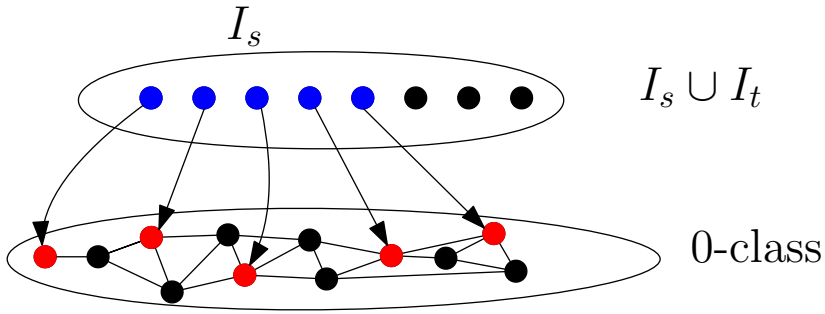
Claim :

Size of the 0-class $\geq 4k$ $\Rightarrow$ Project I_s and I_t on the 0-class
$\Rightarrow$ yes-instance

Sketch of the proof

- Select an IS of size k arbitrarily in the 0-class
- Project one by one the vertices of I_s (resp. I_t) on •

Reducing 0-classes



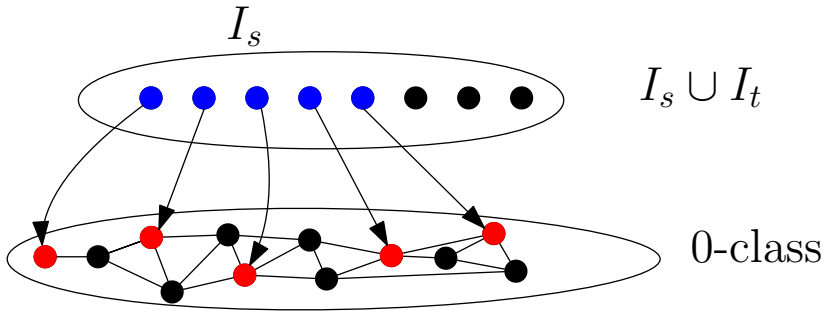
Claim :

Size of the 0-class $\geq 4k \Rightarrow$ Project I_s and I_t on the 0-class
$\Rightarrow$ yes-instance

Sketch of the proof

- Select an IS of size k arbitrarily in the 0-class
- Project one by one the vertices of I_s (resp. I_t) on $\bullet$
- $I_s \rightsquigarrow \bullet \rightsquigarrow I_t$

Reducing 0-classes



Claim :

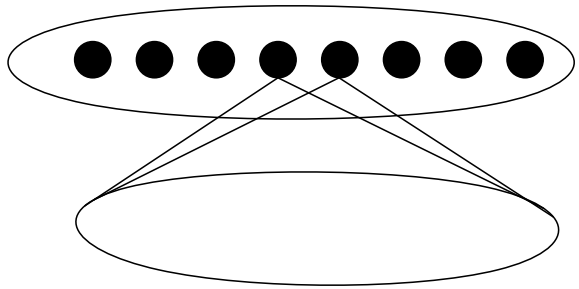
Size of the 0-class $\geq 4k \Rightarrow$ Project I_s and I_t on the 0-class
$\Rightarrow$ yes-instance

Sketch of the proof

- Select an IS of size k arbitrarily in the 0-class
- Project one by one the vertices of I_s (resp. I_t) on $\bullet$
- $I_s \rightsquigarrow \bullet \rightsquigarrow I_t$

With a similar argument : $|\cup$ 1-classes$| = O(k)$

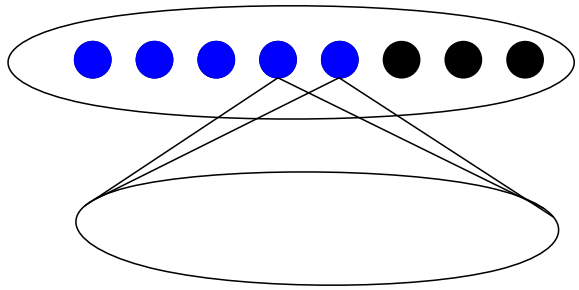
Reducing 2-classes (I)



Claim. If a 2-class C has size $\geq 8k \Rightarrow$ Remove a vertex of C.

Sketch of the proof.

Reducing 2-classes (I)

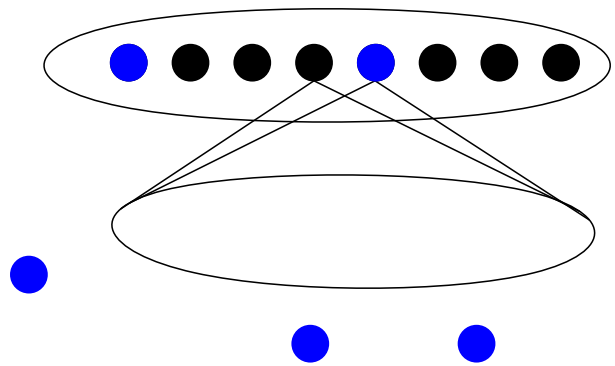


Claim. If a 2-class C has size $\geq 8k \Rightarrow$ Remove a vertex of C.

Sketch of the proof.

- If we never use a vertex of C in $I_s \rightsquigarrow I_t \rightarrow$ ✓

Reducing 2-classes (I)



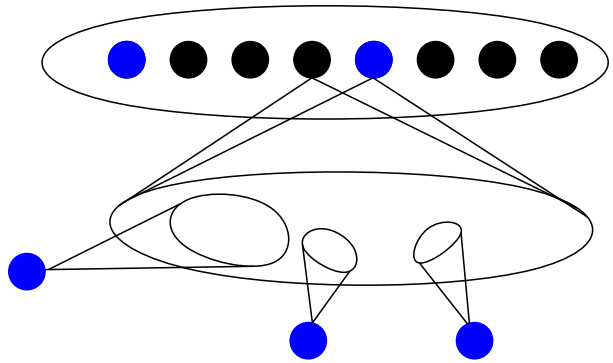
Claim. If a 2-class C has size $\geq 8k \Rightarrow$ Remove a vertex of C.

Sketch of the proof.

- If we never use a vertex of C in $I_s \rightsquigarrow I_t \rightarrow$ ✓
- Consider the IS • just before we use a vertex of C.

  (It contains at most one of the two vertices of I_s incident to the class)

Reducing 2-classes (I)

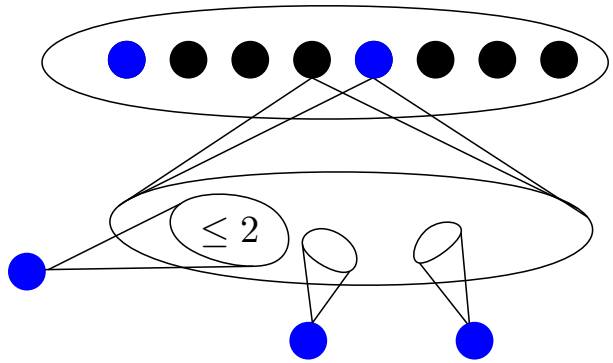


Claim. If a 2-class C has size $\geq 8k \Rightarrow$ Remove a vertex of C.

Sketch of the proof.

- If we never use a vertex of C in $I_s \rightsquigarrow I_t \rightarrow$ ✓
- Consider the IS • just before we use a vertex of C.
  (It contains at most one of the two vertices of I_s incident to the class)
- Each • vertex has at most 2 neighbors in C
  $\Rightarrow$ The union of the neighborhood of • has size $\leq 2k$.

Reducing 2-classes (I)



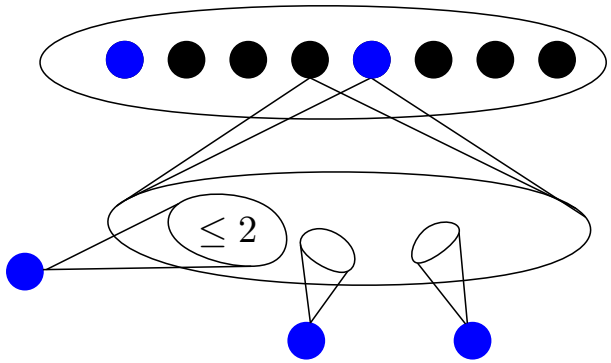
$I_s \cup I_t$

Claim. If a 2-class C has size $\geq 8k \Rightarrow$ Remove a vertex of C.

Sketch of the proof.

- If we never use a vertex of C in $I_s \rightsquigarrow I_t \rightarrow$ ✓
- Consider the IS • just before we use a vertex of C.
  (It contains at most one of the two vertices of I_s incident to the class)
- Each • vertex has at most 2 neighbors in C
  $\Rightarrow$ The union of the neighborhood of • has size $\leq 2k$.

Reducing 2-classes (I)



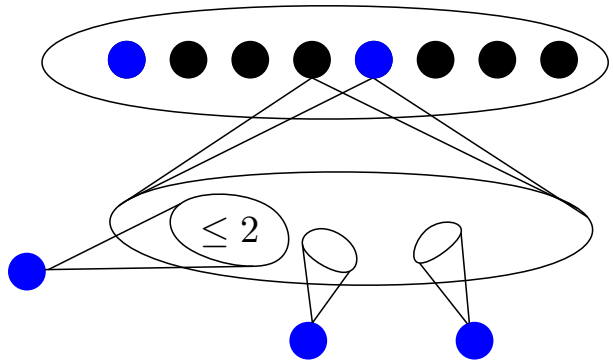
$I_s \cup I_t$

Claim. If a 2-class C has size $\geq 8k \Rightarrow$ Remove a vertex of C.

Sketch of the proof.

- If we never use a vertex of C in $I_s \rightsquigarrow I_t \rightarrow$ ✓
- Consider the IS • just before we use a vertex of C.
  (It contains at most one of the two vertices of I_s incident to the class)
- Each • vertex has at most 2 neighbors in C
  $\Rightarrow$ The union of the neighborhood of • has size $\leq 2k$.
- We can project on an IS of the complement (which has size $\geq 4k$).

Reducing 2-classes (I)



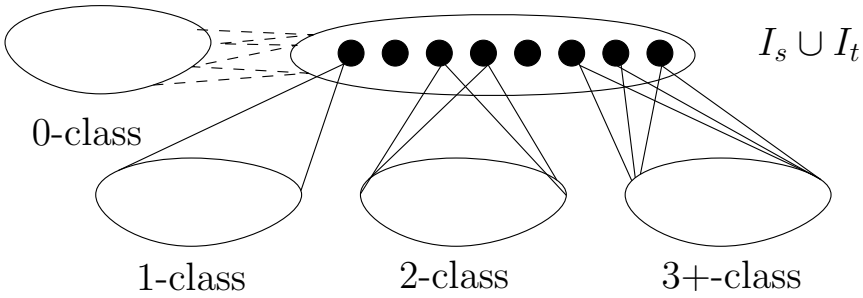
$I_s \cup I_t$

Claim. If a 2-class C has size $\geq 8k \Rightarrow$ Remove a vertex of C.

Sketch of the proof.

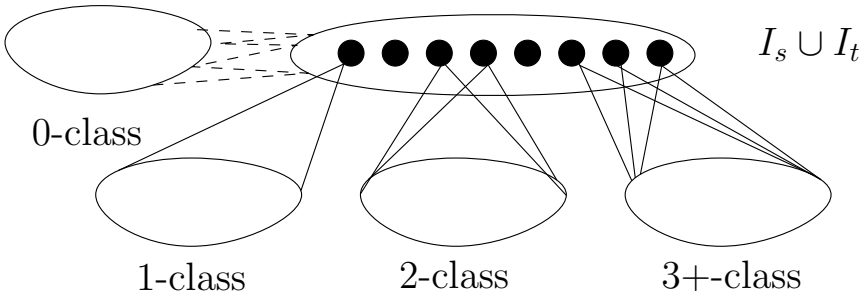
- If we never use a vertex of C in $I_s \rightsquigarrow I_t \rightarrow$ ✓
- Consider the IS • just before we use a vertex of C.
  (It contains at most one of the two vertices of I_s incident to the class)
- Each • vertex has at most 2 neighbors in C
  $\Rightarrow$ The union of the neighborhood of • has size $\leq 2k$.
- We can project on an IS of the complement (which has size $\geq 4k$).
- Apply the same argument on I_t.

Roadmap towards a linear kernel



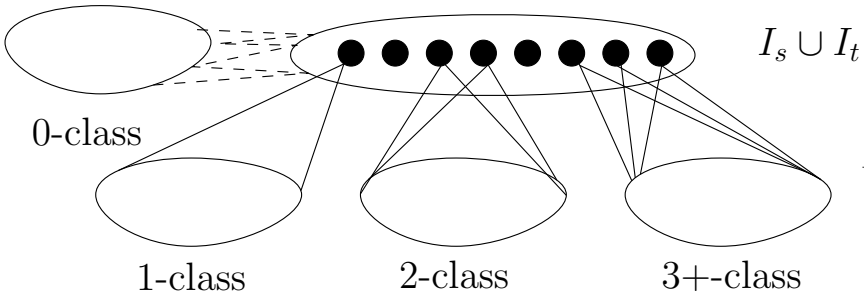
- The 0-class has size $O(k)$ as well as the union of the 1-classes
- 3+-classes have constant size
- Each 2-class has linear size

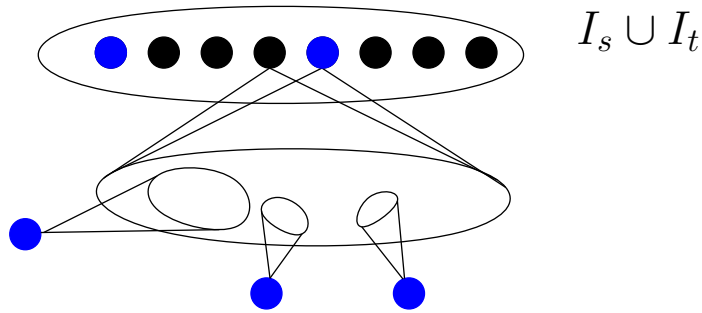
Roadmap towards a linear kernel



- The 0-class has size $O(k)$ as well as the union of the 1-classes
- 3+-classes have constant size
- ~~Each~~ **The union of** the 2-class has linear size

Roadmap towards a linear kernel



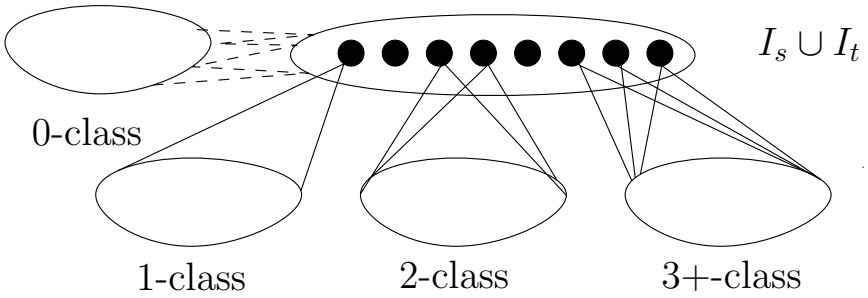


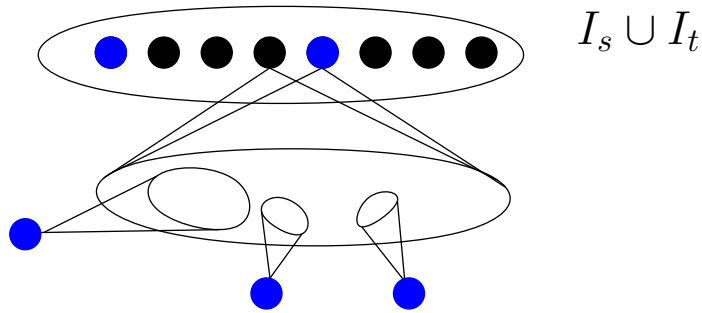
- The 0-class has size $O(k)$ as well as the union of the 1-classes
- 3+-classes have constant size
- ~~Each~~ **The union of** the 2-class has linear size

In the reduction of 2-classes, we needed :

- The size of $N(\bullet)$ is $\leq 2k$
- The size of the complement of $N(\bullet)$ is $\geq 4k$

Roadmap towards a linear kernel



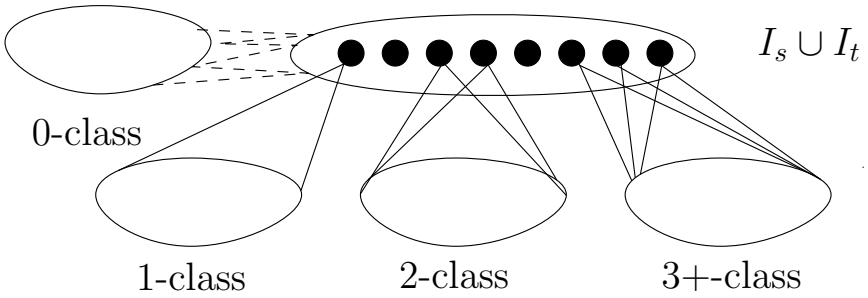


- The 0-class has size $O(k)$ as well as the union of the 1-classes
- 3+-classes have constant size
- ~~Each~~ **The union of** the 2-class has linear size

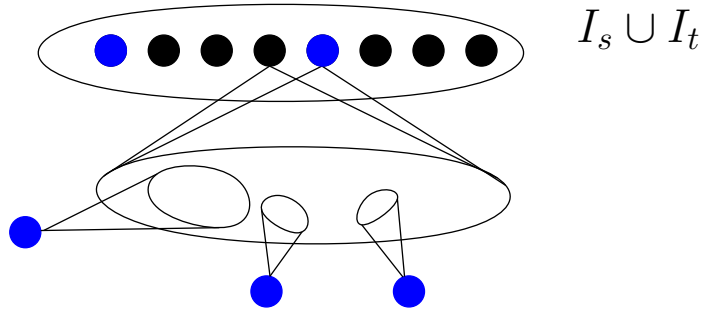
In the reduction of 2-classes, we needed :

- The size of $N(\bullet)$ is ~~$\leq 2k$~~ ≤ 2
- The size of the complement of $N(\bullet)$ is $\geq 4k$

Roadmap towards a linear kernel



- The 0-class has size $O(k)$ as well as the union of the 1-classes
- 3+-classes have constant size
- ~~Each~~ **The union of** the 2-class has linear size



In the reduction of 2-classes, we needed :

- The size of $N(\bullet)$ is ~~$\leq 2k$~~ ≤ 2
- ~~The complement of $N(\bullet)$ has size $\geq 4k$~~ The size of the union of the 2-classes is $O(k)$

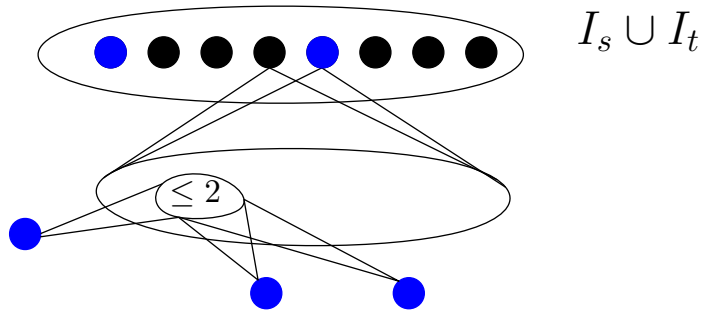
Reducing 2-classes with minors (II)

Idea 1 : Use the full power of minors

Reducing 2-classes with minors (II)

Idea 1 : Use the full power of minors

Informal claim : The size of $N(\bullet)$ in C is ≤ 2.

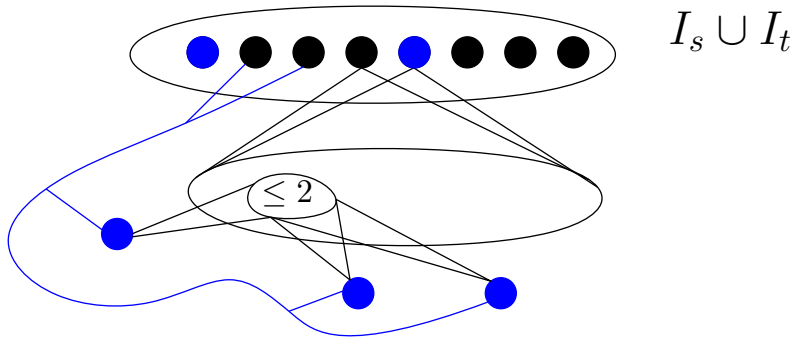


- Consider a shortest sequence $\mathcal{S}$ that allows to move a token on C

Reducing 2-classes with minors (II)

Idea 1 : Use the full power of minors

Informal claim : The size of $N(\bullet)$ in C is ≤ 2.

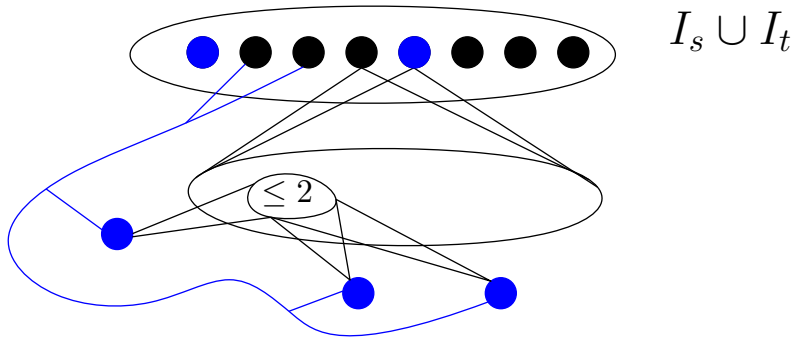


- Consider a shortest sequence $\mathcal{S}$ that allows to move a token on C
- Claim : The union of the vertices involved in a move of $\mathcal{S}$ is connected (by minimality of the sequence)

Reducing 2-classes with minors (II)

Idea 1 : Use the full power of minors

Informal claim : The size of $N(\bullet)$ in C is ≤ 2.



- Consider a shortest sequence $\mathcal{S}$ that allows to move a token on C
- Claim : The union of the vertices involved in a move of $\mathcal{S}$ is connected (by minimality of the sequence)
- Since G is $K_{3,3}$-minor free $\Rightarrow$ $N(\bullet) \leq 2$.

Conclusion

Remark : Our result extends to $K_{3,\ell}$-minor-free graphs.

Conclusion

Remark : Our result extends to $K_{3,\ell}$-minor-free graphs.

Kernalization.

- Beyond planar : minor-free ? $K_{\ell,\ell}$-free ? Geometric graphs ?

Conclusion

Remark : Our result extends to $K_{3,\ell}$-minor-free graphs.

Kernalization.

- Beyond planar : minor-free ? $K_{\ell,\ell}$-free ? Geometric graphs ?
- Token Sliding variant ? (Polynomial kernel open for planar graphs)

Conclusion

Remark : Our result extends to $K_{3,\ell}$-minor-free graphs.

Kernalization.

- Beyond planar : minor-free ? $K_{\ell,\ell}$-free ? Geometric graphs ?
- Token Sliding variant ? (Polynomial kernel open for planar graphs)

Exponential time algorithms.

Our kernel $\Rightarrow$ The first $2^{O(k)}$ algorithm.

Conclusion

Remark : Our result extends to $K_{3,\ell}$-minor-free graphs.

Kernalization.

- Beyond planar : minor-free ? $K_{\ell,\ell}$-free ? Geometric graphs ?
- Token Sliding variant ? (Polynomial kernel open for planar graphs)

Exponential time algorithms.

Our kernel $\Rightarrow$ The first $2^{O(k)}$ algorithm.

- More single-exponential time algorithms ? (With direct proofs ?)

Conclusion

Remark : Our result extends to $K_{3,\ell}$-minor-free graphs.

Kernalization.

- Beyond planar : minor-free ? $K_{\ell,\ell}$-free ? Geometric graphs ?
- Token Sliding variant ? (Polynomial kernel open for planar graphs)

Exponential time algorithms.

Our kernel $\Rightarrow$ The first $2^{O(k)}$ algorithm.

- More single-exponential time algorithms ? (With direct proofs ?)
- Subexponential time algorithms ?

Conclusion

Remark : Our result extends to $K_{3,\ell}$-minor-free graphs.

Kernalization.

- Beyond planar : minor-free ? $K_{\ell,\ell}$-free ? Geometric graphs ?
- Token Sliding variant ? (Polynomial kernel open for planar graphs)

Exponential time algorithms.
Our kernel $\Rightarrow$ The first $2^{O(k)}$ algorithm.

- More single-exponential time algorithms ? (With direct proofs ?)
- Subexponential time algorithms ?

Extremal combinatorics. Maximum length of transformations ?

Conclusion

Remark : Our result extends to $K_{3,\ell}$-minor-free graphs.

Kernalization.

- Beyond planar : minor-free ? $K_{\ell,\ell}$-free ? Geometric graphs ?
- Token Sliding variant ? (Polynomial kernel open for planar graphs)

Exponential time algorithms.
Our kernel $\Rightarrow$ The first $2^{O(k)}$ algorithm.

- More single-exponential time algorithms ? (With direct proofs ?)
- Subexponential time algorithms ?

Extremal combinatorics. Maximum length of transformations ?

Thanks for your attention !