Independent Set Reconfiguration via Token Sliding

Nicolas Bousquet

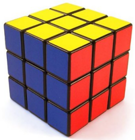
joint works with

Valentin Bartier, Marthe Bonamy, Clément Dallard, Kyle Lomer, Amer Mouawad and Moritz Mühlenthaler

December 2020

Reconfiguration

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

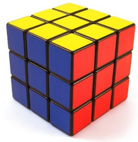


Question :

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.



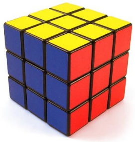
Question :

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

- Reconfiguration introduced for colorings, satisfiability problems, dominating sets, cliques, list colorings, bases of matroids, boolean formulas...

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 introduced for colorings, satisfiability problems, dominating sets, cliques, list colorings, bases of matroids, boolean formulas...
- Applications to random sampling, bioinformatics...etc...

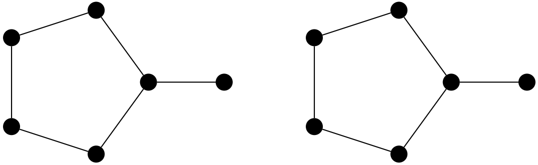
Main questions

- **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...).

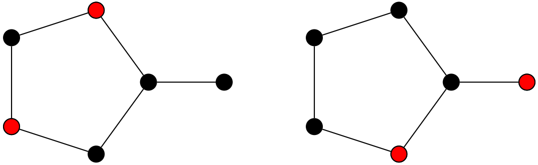
Token Sliding



Definition (TS-sequence)

A TS-sequence $I_1, \ldots, I_\ell$ of independent sets is a sequence such that there exist $v \in I_{j+1}$ and $u \in I_j$ such that $I_{j+1} = I_j \cup \{v\} \setminus \{u\}$ and uv is an edge.

Token Sliding



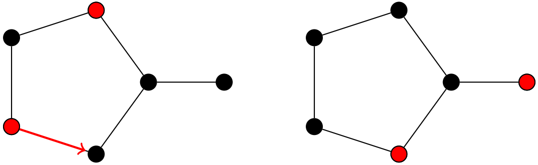
Definition (TS-sequence)

A TS-sequence $I_1, \ldots, I_\ell$ of independent sets is a sequence such that there exist $v \in I_{j+1}$ and $u \in I_j$ such that $I_{j+1} = I_j \cup \{v\} \setminus \{u\}$ and uv is an edge.

Equivalent formulation :

We slide tokens along the edges in such a way the set remains an independent set at any step.

Token Sliding



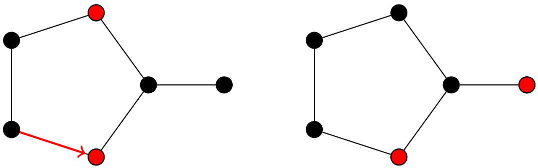
Definition (TS-sequence)

A TS-sequence $I_1, \ldots, I_\ell$ of independent sets is a sequence such that there exist $v \in I_{j+1}$ and $u \in I_j$ such that $I_{j+1} = I_j \cup \{v\} \setminus \{u\}$ and uv is an edge.

Equivalent formulation :

We slide tokens along the edges in such a way the set remains an independent set at any step.

Token Sliding



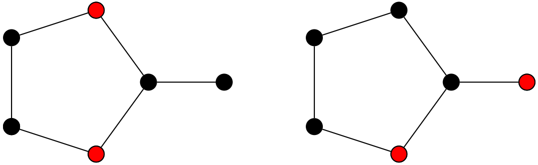
Definition (TS-sequence)

A TS-sequence $I_1, \ldots, I_\ell$ of independent sets is a sequence such that there exist $v \in I_{j+1}$ and $u \in I_j$ such that $I_{j+1} = I_j \cup \{v\} \setminus \{u\}$ and uv is an edge.

Equivalent formulation :

We slide tokens along the edges in such a way the set remains an independent set at any step.

Token Sliding



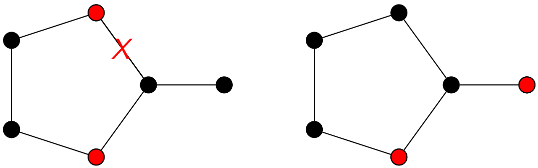
Definition (TS-sequence)

A TS-sequence $I_1, \ldots, I_\ell$ of independent sets is a sequence such that there exist $v \in I_{j+1}$ and $u \in I_j$ such that $I_{j+1} = I_j \cup \{v\} \setminus \{u\}$ and uv is an edge.

Equivalent formulation :

We slide tokens along the edges in such a way the set remains an independent set at any step.

Token Sliding



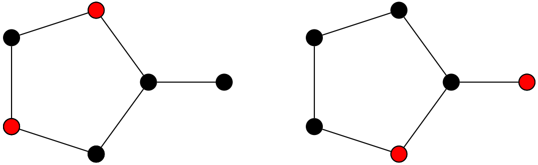
Definition (TS-sequence)

A TS-sequence $I_1, \ldots, I_\ell$ of independent sets is a sequence such that there exist $v \in I_{j+1}$ and $u \in I_j$ such that $I_{j+1} = I_j \cup \{v\} \setminus \{u\}$ and uv is an edge.

Equivalent formulation :

We slide tokens along the edges in such a way the set remains an independent set at any step.

Token Sliding



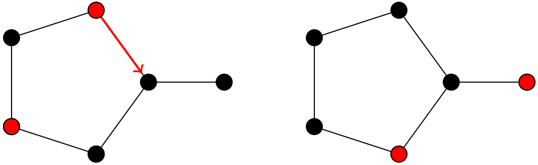
Definition (TS-sequence)

A TS-sequence $I_1, \ldots, I_\ell$ of independent sets is a sequence such that there exist $v \in I_{j+1}$ and $u \in I_j$ such that $I_{j+1} = I_j \cup \{v\} \setminus \{u\}$ and uv is an edge.

Equivalent formulation :

We slide tokens along the edges in such a way the set remains an independent set at any step.

Token Sliding



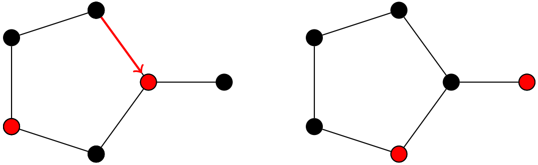
Definition (TS-sequence)

A TS-sequence $I_1, \ldots, I_\ell$ of independent sets is a sequence such that there exist $v \in I_{j+1}$ and $u \in I_j$ such that $I_{j+1} = I_j \cup \{v\} \setminus \{u\}$ and uv is an edge.

Equivalent formulation :

We slide tokens along the edges in such a way the set remains an independent set at any step.

Token Sliding



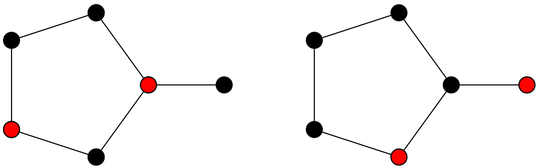
Definition (TS-sequence)

A TS-sequence $I_1, \ldots, I_\ell$ of independent sets is a sequence such that there exist $v \in I_{j+1}$ and $u \in I_j$ such that $I_{j+1} = I_j \cup \{v\} \setminus \{u\}$ and uv is an edge.

Equivalent formulation :

We slide tokens along the edges in such a way the set remains an independent set at any step.

Token Sliding



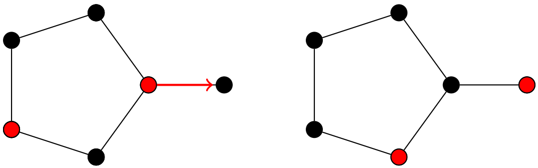
Definition (TS-sequence)

A TS-sequence $I_1, \ldots, I_\ell$ of independent sets is a sequence such that there exist $v \in I_{j+1}$ and $u \in I_j$ such that $I_{j+1} = I_j \cup \{v\} \setminus \{u\}$ and uv is an edge.

Equivalent formulation :

We slide tokens along the edges in such a way the set remains an independent set at any step.

Token Sliding

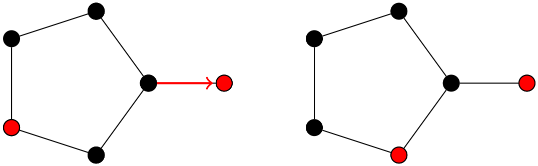


Definition (TS-sequence)

A TS-sequence $I_1, \ldots, I_\ell$ of independent sets is a sequence such that there exist $v \in I_{j+1}$ and $u \in I_j$ such that $I_{j+1} = I_j \cup \{v\} \setminus \{u\}$ and uv is an edge.

Equivalent formulation :
We slide tokens along the edges in such a way the set remains an independent set at any step.

Token Sliding



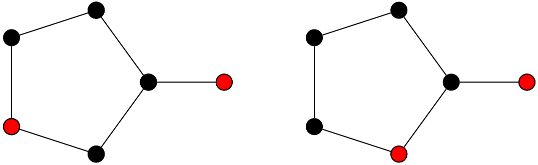
Definition (TS-sequence)

A TS-sequence $I_1, \ldots, I_\ell$ of independent sets is a sequence such that there exist $v \in I_{j+1}$ and $u \in I_j$ such that $I_{j+1} = I_j \cup \{v\} \setminus \{u\}$ and uv is an edge.

Equivalent formulation :

We slide tokens along the edges in such a way the set remains an independent set at any step.

Token Sliding



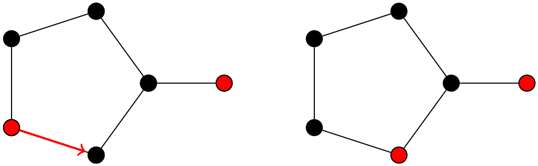
Definition (TS-sequence)

A TS-sequence $I_1, \ldots, I_\ell$ of independent sets is a sequence such that there exist $v \in I_{j+1}$ and $u \in I_j$ such that $I_{j+1} = I_j \cup \{v\} \setminus \{u\}$ and uv is an edge.

Equivalent formulation :

We slide tokens along the edges in such a way the set remains an independent set at any step.

Token Sliding



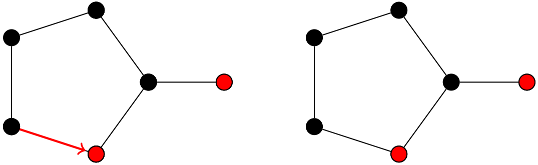
Definition (TS-sequence)

A TS-sequence $I_1, \dots, I_\ell$ of independent sets is a sequence such that there exist $v \in I_{j+1}$ and $u \in I_j$ such that $I_{j+1} = I_j \cup \{v\} \setminus \{u\}$ and uv is an edge.

Equivalent formulation :

We slide tokens along the edges in such a way the set remains an independent set at any step.

Token Sliding



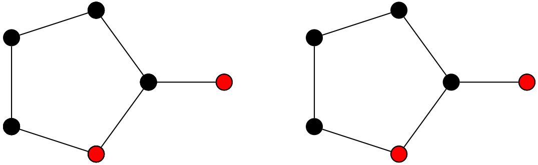
Definition (TS-sequence)

A TS-sequence $I_1, \ldots, I_\ell$ of independent sets is a sequence such that there exist $v \in I_{j+1}$ and $u \in I_j$ such that $I_{j+1} = I_j \cup \{v\} \setminus \{u\}$ and uv is an edge.

Equivalent formulation :

We slide tokens along the edges in such a way the set remains an independent set at any step.

Token Sliding



Definition (TS-sequence)

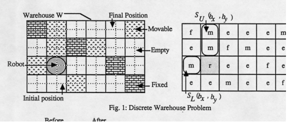
A TS-sequence $I_1, \ldots, I_\ell$ of independent sets is a sequence such that there exist $v \in I_{j+1}$ and $u \in I_j$ such that $I_{j+1} = I_j \cup \{v\} \setminus \{u\}$ and uv is an edge.

Equivalent formulation :

We slide tokens along the edges in such a way the set remains an independent set at any step.

Genesis

- [Hopcroft, Schwartz, Sharir '83] Warehouseman's problem - Motion of rectangular robots in a grid.
  ⇒ PSPACE-complete (but they need large robots).



Genesis

- [Hopcroft, Schwartz, Sharir '83] Warehouseman's problem - Motion of rectangular robots in a grid.
  $\Rightarrow$ PSPACE-complete (but they need large robots).
- [Flake, Baum '03] Rush hour is PSPACE-complete.

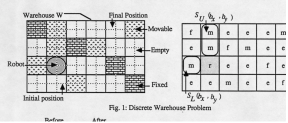


Fig. 1: Discrete Warehouse Problem

Genesis

- [Hopcroft, Schwartz, Sharir '83] Warehouseman's problem - Motion of rectangular robots in a grid.
  ⇒ PSPACE-complete (but they need large robots).
- [Flake, Baum '03] Rush hour is PSPACE-complete.

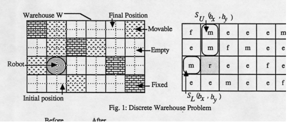


Fig. 1: Discrete Warehouse Problem

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

TS-Reachability

TS-Reachability
Input : A graph G, $k \in \mathbb{N}$, two independent sets I, J of size k.
Output : YES iff there exists a TS-sequence from I to J.

Theorem (Hearn, Demaine '05)

TS-Reachability is PSPACE-complete even restricted to planar graphs of maximum degree at most 3.

$\rightarrow$ On which graph class is the problem polynomial ?

$\rightarrow$ On which graph class is the problem polynomial ?

Polynomial time algorithms for :

- [Demaine et al.] Trees.
- [Kamiński, Medvedev, Milanič] Cographs.
- [Bonsma, Kamiński, Wrochna] Claw-free graphs.
- [Fox-Epstein et al.] Bipartite permutation graphs.

Graph classes

→ On which graph class is the problem polynomial ?

Polynomial time algorithms for :

- [Demaine et al.] Trees.
- [Kamiński, Medvedev, Milanič] Cographs.
- [Bonsma, Kamiński, Wrochna] Claw-free graphs.
- [Fox-Epstein et al.] Bipartite permutation graphs.

Question (Demaine et al.)

Can the TS-Reachability problem be decided on polynomial time on interval graphs ? On chordal graphs ?

Graph classes

→ On which graph class is the problem polynomial ?

Polynomial time algorithms for :

- [Demaine et al.] Trees.
- [Kamiński, Medvedev, Milanič] Cographs.
- [Bonsma, Kamiński, Wrochna] Claw-free graphs.
- [Fox-Epstein et al.] Bipartite permutation graphs.

Question (Demaine et al.)

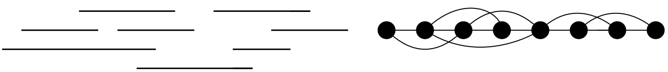
Can the TS-Reachability problem be decided on polynomial time on interval graphs ? On chordal graphs ?

Answers :

- [Bonamy, B. '18] YES on interval graphs.
- [Belmonte et al. '19] NO on split graphs.

(split graph = $V = V_1 \cup V_2$ where V_1 induces a clique and V_2 a stable set)

Interval graphs

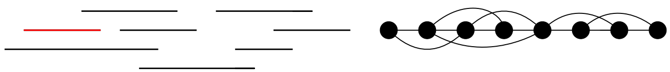


An interval graph is an intersection graph of intervals on the line.

Remark :

A geometric representation can be obtained in polynomial time.

Interval graphs



An interval graph is an intersection graph of intervals on the line.

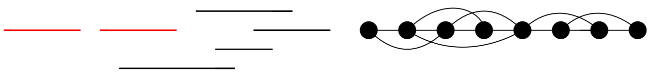
Remark :

A geometric representation can be obtained in polynomial time.

The Leftmost Independent Set (LIS) satisfies :

- The LIS contains the leftmost vertex, i.e. the vertex x with minimum right-end.

Interval graphs



An interval graph is an intersection graph of intervals on the line.

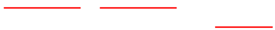
Remark :

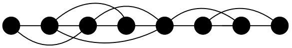
A geometric representation can be obtained in polynomial time.

The Leftmost Independent Set (LIS) satisfies :

- The LIS contains the leftmost vertex, i.e. the vertex x with minimum right-end.
- $LIS(G) = x \cup LIS(G[V \setminus N[x]])$.

Interval graphs





An interval graph is an intersection graph of intervals on the line.

Remark :

A geometric representation can be obtained in polynomial time.

The Leftmost Independent Set (LIS) satisfies :

- The LIS contains the leftmost vertex, i.e. the vertex x with minimum right-end.
- $LIS(G) = x \cup LIS(G[V \setminus N[x]])$.

Interval graphs

An interval graph is an intersection graph of intervals on the line.

Remark :

A geometric representation can be obtained in polynomial time.

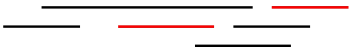
The Leftmost Independent Set (LIS) satisfies :

- The LIS contains the leftmost vertex, i.e. the vertex x with minimum right-end.
- $LIS(G) = x \cup LIS(G[V \setminus N[x]])$.

Today :

Decide if an independent set of size k can be transformed into the LIS.

First try : Naive Method

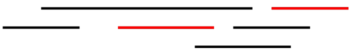


Lemma

I can be transformed into the LIS iff

- The leftmost vertex of x of I can be pushed to the leftmost vertex y of $LIS(G)$.
- $I \setminus x$ can be transformed into $LIS(G) \setminus y$ in $G[V \setminus N[y]]$.

First try : Naive Method



Lemma

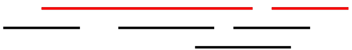
I can be transformed into the LIS iff

- The leftmost vertex of x of I can be pushed to the leftmost vertex y of $LIS(G)$.
- $I \setminus x$ can be transformed into $LIS(G) \setminus y$ in $G[V \setminus N[y]]$.

Naive algorithm :

- Push the leftmost vertex of the independent set to the left and check that it can be transformed into the leftmost vertex.

First try : Naive Method



Lemma

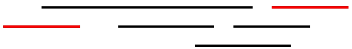
I can be transformed into the LIS iff

- The leftmost vertex of x of I can be pushed to the leftmost vertex y of $LIS(G)$.
- $I \setminus x$ can be transformed into $LIS(G) \setminus y$ in $G[V \setminus N[y]]$.

Naive algorithm :

- Push the leftmost vertex of the independent set to the left and check that it can be transformed into the leftmost vertex.

First try : Naive Method



Lemma

I can be transformed into the LIS iff

- The leftmost vertex of x of I can be pushed to the leftmost vertex y of $LIS(G)$.
- $I \setminus x$ can be transformed into $LIS(G) \setminus y$ in $G[V \setminus N[y]]$.

Naive algorithm :

- Push the leftmost vertex of the independent set to the left and check that it can be transformed into the leftmost vertex.

First try : Naive Method

Lemma

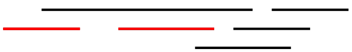
I can be transformed into the LIS iff

- The leftmost vertex of x of I can be pushed to the leftmost vertex y of $LIS(G)$.
- $I \setminus x$ can be transformed into $LIS(G) \setminus y$ in $G[V \setminus N[y]]$.

Naive algorithm :

- Push the leftmost vertex of the independent set to the left and check that it can be transformed into the leftmost vertex.
- Repeat in $V \setminus N[y]$ for the remaining vertices.

First try : Naive Method



Lemma

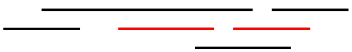
I can be transformed into the LIS iff

- The leftmost vertex of x of I can be pushed to the leftmost vertex y of $LIS(G)$.
- $I \setminus x$ can be transformed into $LIS(G) \setminus y$ in $G[V \setminus N[y]]$.

Naive algorithm :

- Push the leftmost vertex of the independent set to the left and check that it can be transformed into the leftmost vertex.
- Repeat in $V \setminus N[y]$ for the remaining vertices.

First try : Naive Method



Lemma

I can be transformed into the LIS iff

- The leftmost vertex of x of I can be pushed to the leftmost vertex y of $LIS(G)$.
- $I \setminus x$ can be transformed into $LIS(G) \setminus y$ in $G[V \setminus N[y]]$.

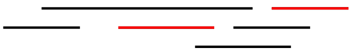
Naive algorithm :

- Push the leftmost vertex of the independent set to the left and check that it can be transformed into the leftmost vertex.
- Repeat in $V \setminus N[y]$ for the remaining vertices.

Problem :

We might need to move vertices to the right to push the leftmost vertex to the left.

First try : Naive Method



Lemma

I can be transformed into the LIS iff

- The leftmost vertex of x of I can be pushed to the leftmost vertex y of $LIS(G)$.
- $I \setminus x$ can be transformed into $LIS(G) \setminus y$ in $G[V \setminus N[y]]$.

Naive algorithm :

- Push the leftmost vertex of the independent set to the left and check that it can be transformed into the leftmost vertex.
- Repeat in $V \setminus N[y]$ for the remaining vertices.

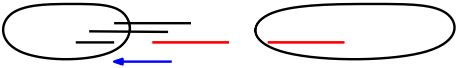
Problem :

We might need to move vertices to the right to push the leftmost vertex to the left.

Second try

Repeat the following procedure

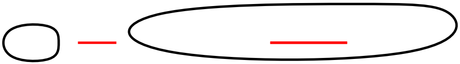
Second try



Repeat the following procedure

- Push the first vertex to the left.

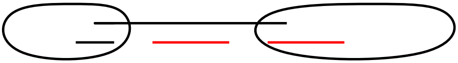
Second try



Repeat the following procedure

- Push the first vertex to the left.

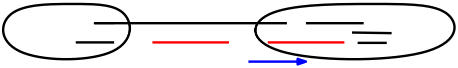
Second try



Repeat the following procedure

- Push the first vertex to the left.
- Push the independent set minus its first vertex to the right.

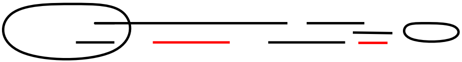
Second try



Repeat the following procedure

- Push the first vertex to the left.
- Push the independent set minus its first vertex to the right.

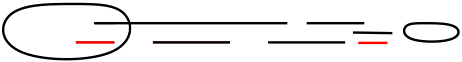
Second try



Repeat the following procedure

- Push the first vertex to the left.
- Push the independent set minus its first vertex to the right.

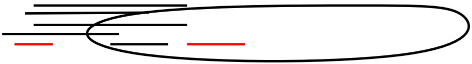
Second try



Repeat the following procedure

- Push the first vertex to the left.
- Push the independent set minus its first vertex to the right.

Second try



Repeat the following procedure

- Push the first vertex to the left.
- Push the independent set minus its first vertex to the right.
- If the leftmost vertex is the first vertex of the LIS, apply induction (with $k \leftarrow k-1$).

Second try

Repeat the following procedure

- Push the first vertex to the left.
- Push the independent set minus its first vertex to the right.
- If the leftmost vertex is the first vertex of the LIS, apply induction (with $k \leftarrow k-1$).

Second try

Repeat the following procedure

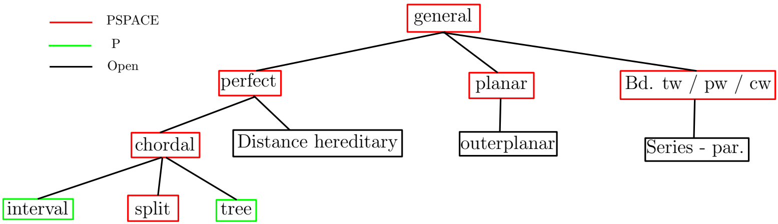
- Push the first vertex to the left.
- Push the independent set minus its first vertex to the right.
- If the leftmost vertex is the first vertex of the LIS, apply induction (with $k \leftarrow k - 1$).
- Otherwise we can't reach the LIS.

Questions

- Our algorithm is polynomial but does not guarantee a polynomial transformation.
  Question : Does there always exist a polynomial transformation ?

Questions

- Our algorithm is polynomial but does not guarantee a polynomial transformation.
  Question : Does there always exist a polynomial transformation ?



Matching Reconfiguration

A matching is a subset of edges that are pairwise endpoint disjoint.

Theorem (Ito et al. '12)

TS-Matching Reachability can be decided in polynomial time.

Matching Reconfiguration

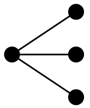
A matching is a subset of edges that are pairwise endpoint disjoint.

Theorem (Ito et al. '12)

TS-MATCHING REACHABILITY can be decided in polynomial time.

Theorem (Bonsma, Kamiński, Wrochna '14)

TS-REACHABILITY can be decided in polynomial time in claw-free graphs.



Matching Reconfiguration

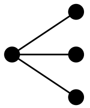
A matching is a subset of edges that are pairwise endpoint disjoint.

Theorem (Ito et al. '12)

TS-MATCHING REACHABILITY can be decided in polynomial time.

Theorem (Bonsma, Kamiński, Wrochna '14)

TS-REACHABILITY can be decided in polynomial time in claw-free graphs.



Question : For which H is TS-Independent Set Reconfiguration polynomial on H-free graphs ?

Theorem (Bartier, B., Mühlenthaler '20+)

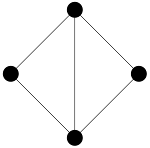
TS-REACHABILITY is PSPACE-complete on H-free graphs for every H which is not a subdivided claw.

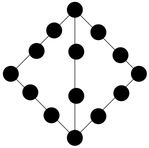
Theorem (Bartier, B., Mühlenthaler '20+)

TS-REACHABILITY is PSPACE-complete on H-free graphs for every H which is not a subdivided claw.

Sketch of the proof : *a la Alekseev*

- Subdivision of every edge of G twice $\Rightarrow$ MIS increases by $|E|$.



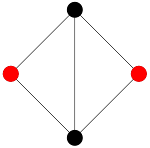


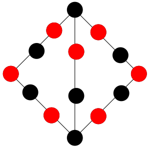
Theorem (Bartier, B., Mühlenthaler '20+)

TS-REACHABILITY is PSPACE-complete on H-free graphs for every H which is not a subdivided claw.

Sketch of the proof : *a la Alekseev*

- Subdivision of every edge of G twice $\Rightarrow$ MIS increases by $|E|$.



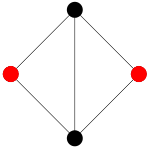


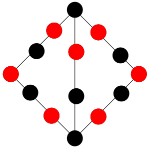
Theorem (Bartier, B., Mühlenthaler '20+)

TS-REACHABILITY is PSPACE-complete on H-free graphs for every H which is not a subdivided claw.

Sketch of the proof : *a la Alekseev*

- Subdivision of every edge of G twice $\Rightarrow$ MIS increases by $|E|$.
- The same trick can be adapted in the reconfiguration setting.





Theorem (Bartier, B., Mühlenthaler '20+)

TS-REACHABILITY is PSPACE-complete on H-free graphs for every H which is not a subdivided claw.

Sketch of the proof : *a la Alekseev*

- Subdivision of every edge of G twice $\Rightarrow$ MIS increases by $|E|$.
- The same trick can be adapted in the reconfiguration setting.

- Start with a planar graph of maximum degree 3.
- Repeat the subdivision process $|H|$ times.

$\Rightarrow$ No copy of H if H has a vertex of degree ≥ 4 or a cycle or two vertices of degree ≥ 3.

$\mathbf{TS}_m$-Reachability

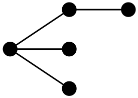
Input : A graph G, two independent sets I, J of maximum size.
Output : YES iff there exists a TS-sequence from I to J.

TS_m-Reachability

Input : A graph G, two independent sets I, J of maximum size.
Output : YES iff there exists a TS-sequence from I to J.

Theorem (Bartier, B., Mühlenthaler '20+)

TS_m-Reachability can be decided in polynomial time in fork-free graphs.



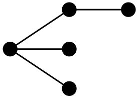
TS$_m$-Reachability

Input : A graph G, two independent sets I, J of maximum size.

Output : YES iff there exists a TS-sequence from I to J.

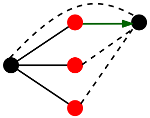
Theorem (Bartier, B., Mühlenthaler '20+)

TS$_m$-Reachability can be decided in polynomial time in fork-free graphs.



Sketch of the proof :

- If a vertex has at least 3 tokens in its neighborhood $\rightarrow$ Delete it.



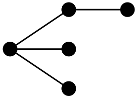
TS_m-Reachability

Input : A graph G, two independent sets I, J of maximum size.

Output : YES iff there exists a TS-sequence from I to J.

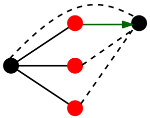
Theorem (Bartier, B., Mühlenthaler '20+)

TS_m-REACHABILITY can be decided in polynomial time in fork-free graphs.



Sketch of the proof :

- If a vertex has at least 3 tokens in its neighborhood $\rightarrow$ Delete it.
- Prove that we can reduce to claw-free graphs.

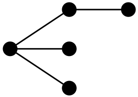


TS_m-Reachability

Input : A graph G, two independent sets I, J of maximum size.
Output : YES iff there exists a TS-sequence from I to J.

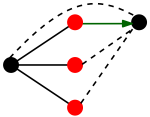
Theorem (Bartier, B., Mühlenthaler '20+)

TS_m-REACHABILITY can be decided in polynomial time in fork-free graphs.



Sketch of the proof :

- If a vertex has at least 3 tokens in its neighborhood → Delete it.
- Prove that we can reduce to claw-free graphs.



Questions :

- Complexity of TS-REACHABILITY on fork-free graphs ?

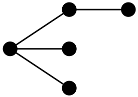
  Last case to completely characterize the complexity of TS-REACHABILITY on connected graphs.

TS_m-Reachability

Input : A graph G, two independent sets I, J of maximum size.
Output : YES iff there exists a TS-sequence from I to J.

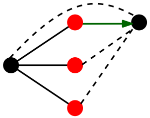
Theorem (Bartier, B., Mühlenthaler '20+)

TS_m-REACHABILITY can be decided in polynomial time in fork-free graphs.



Sketch of the proof :

- If a vertex has at least 3 tokens in its neighborhood $\rightarrow$ Delete it.
- Prove that we can reduce to claw-free graphs.



Questions :

- Complexity of TS-REACHABILITY on fork-free graphs ?
  Last case to completely characterize the complexity of TS-REACHABILITY on connected graphs.
- A few non connected graphs (on which we are currently working on).

Parameterized complexity

A problem Π parameterized by k is FPT if it can be decided in $f(k) \cdot Poly(n)$.

Very few is known on the parameterized complexity for TS-Reachability.

Parameterized complexity

A problem Π parameterized by k is FPT if it can be decided in $f(k) \cdot Poly(n)$.

Very few is known on the parameterized complexity for TS-REACHABILITY.

- TS-REACHABILITY is PSPACE-complete for graphs of bounded treewidth / pathwidth / bandwidth / cliquewidth.

  No hope for a Courcelle like theorem.
- No real understanding on what happens even on very sparse graphs...

Parameterized complexity

A problem Π parameterized by k is FPT if it can be decided in $f(k) \cdot Poly(n)$.

Very few is known on the parameterized complexity for TS-Reachability.

- TS-Reachability is PSPACE-complete for graphs of bounded treewidth / pathwidth / bandwidth / cliquewidth.
  No hope for a Courcelle like theorem.
- No real understanding on what happens even on very sparse graphs...

Two research directions :

- Consider sparse graph classes (e.g. $|E| \leq c|V|$).
- Consider graph classes with "girth" restriction.

Restrictions on the possible cycles

[Lokshtanov, Mouawad '19] TS-Reachability is PSPACE-complete on bipartite graphs.

Restrictions on the possible cycles

[Lokshtanov, Mouawad '19] TS-Reachability is PSPACE-complete on bipartite graphs.

Theorem (Bartier, B., Dallard, Lomer, Mouawad '20)

TS-Reachability is W[1]-hard (very likely not FPT) parameterized by k on bipartite graphs.

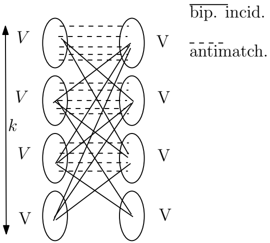
Restrictions on the possible cycles

[Lokshtanov, Mouawad '19] TS-REACHABILITY is PSPACE-complete on bipartite graphs.

Theorem (Bartier, B., Dallard, Lomer, Mouawad '20)

TS-REACHABILITY is W[1]-hard (very likely not FPT) parameterized by k on bipartite graphs.

Proof by picture :



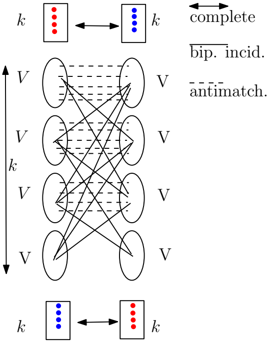
Restrictions on the possible cycles

[Lokshtanov, Mouawad '19] TS-REACHABILITY is PSPACE-complete on bipartite graphs.

Theorem (Bartier, B., Dallard, Lomer, Mouawad '20)

TS-REACHABILITY is W[1]-hard (very likely not FPT) parameterized by k on bipartite graphs.

Proof by picture :



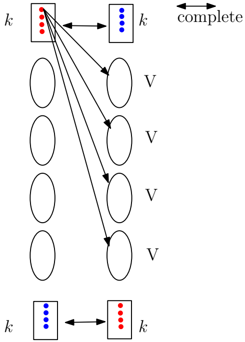
Restrictions on the possible cycles

[Lokshtanov, Mouawad '19] TS-Reachability is PSPACE-complete on bipartite graphs.

Theorem (Bartier, B., Dallard, Lomer, Mouawad '20)

TS-Reachability is W[1]-hard (very likely not FPT) parameterized by k on bipartite graphs.

Proof by picture :



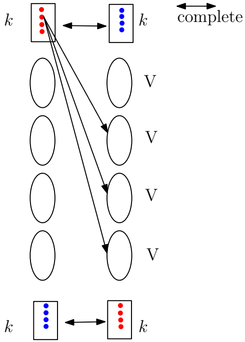
Restrictions on the possible cycles

[Lokshtanov, Mouawad '19] TS-REACHABILITY is PSPACE-complete on bipartite graphs.

Theorem (Bartier, B., Dallard, Lomer, Mouawad '20)

TS-REACHABILITY is W[1]-hard (very likely not FPT) parameterized by k on bipartite graphs.

Proof by picture :



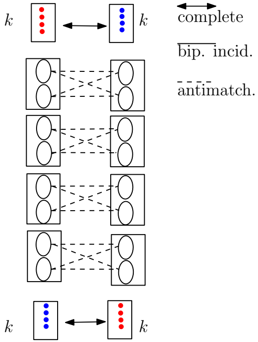
Restrictions on the possible cycles

[Lokshtanov, Mouawad '19] TS-REACHABILITY is PSPACE-complete on bipartite graphs.

Theorem (Bartier, B., Dallard, Lomer, Mouawad '20)

TS-REACHABILITY is W[1]-hard (very likely not FPT) parameterized by k on bipartite graphs.

Proof by picture :



Larger girth

Theorem (Bartier, B., Dallard, Lomer, Mouawad '20)

TS-Reachability is FPT on C_4-free bipartite graphs.

Larger girth

Theorem (Bartier, B., Dallard, Lomer, Mouawad '20)

TS-REACHABILITY is FPT on C_4-free bipartite graphs.

High level idea :

- Bound the degree of the graph.
- [Fox Epstein et al.] Determine the frozen tokens is in P.
- [Fox Epstein et al.] Any reachable vertex can be reached via a sequence all the tokens but ≤ 1 slide $\leq$ once.

$\Rightarrow$ If IS at large distance, we can reach it.

Which cycles are important ?

C_4 ?

Which cycles are important ?

C_4 ? $\rightarrow$ NO !

Theorem (Bartier, B., Dallard, Lomer, Mouawad '20)

TS-Reachability is W[1]-hard parameterized by k on graphs with no induced $C_4, \ldots, C_p$ for $p \in \mathbb{N}$.

Which cycles are important ?

C_4 ? $\rightarrow$ NO !

Theorem (Bartier, B., Dallard, Lomer, Mouawad '20)

TS-Reachability is W[1]-hard parameterized by k on graphs with no induced $C_4, \ldots, C_p$ for $p \in \mathbb{N}$.

High level idea :

- [Bonnet et al. '19] MIS is W[1]-hard on graphs with no induced $C_4, \ldots, C_p$ for $p \in \mathbb{N}$.
- "Gadgetize" from this construction to obtain the reconfiguration counterpart.

Which cycles are important ?

C_4 ? $\rightarrow$ NO !

Theorem (Bartier, B., Dallard, Lomer, Mouawad '20)

TS-REACHABILITY is W[1]-hard parameterized by k on graphs with no induced $C_4, \ldots, C_p$ for $p \in \mathbb{N}$.

High level idea :

- [Bonnet et al. '19] MIS is W[1]-hard on graphs with no induced $C_4, \ldots, C_p$ for $p \in \mathbb{N}$.
- "Gadgetize" from this construction to obtain the reconfiguration counterpart.

Questions :

- TS-REACHABILITY on graphs of girth ≥ 5 ? $\geq \ell$ for some fixed ℓ ?
- TS-REACHABILITY on even hole free graphs ?

  It would imply chordal graphs.

On going work

(Bartier, B., Dallard, Lomer, Mouawad '20+)

On going work

(Bartier, B., Dallard, Lomer, Mouawad '20+)

Hammer # 1 : If $\exists X$ "small" and $G[V \setminus X]$ contains many components, the graph can be reduced.

On going work

(Bartier, B., Dallard, Lomer, Mouawad '20+)

Hammer # 1 : If $\exists X$ "small" and $G[V \setminus X]$ contains many components, the graph can be reduced.

Hammer # 2 : If G contains a long geodesic path, then the graph can be reduced.

On going work

(Bartier, B., Dallard, Lomer, Mouawad '20+)

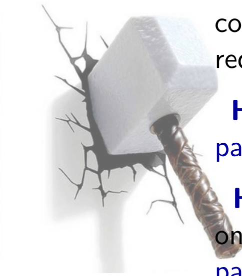
Hammer # 1 : If $\exists X$ "small" and $G[V \setminus X]$ contains many components, the graph can be reduced.

Hammer # 2 : If G contains a long geodesic path, then the graph can be reduced.

Hammer # 3 : If $\exists X$ "small" with no token on X and $G[V \setminus X]$ contains a long geodesic path, then the graph can be reduced.

On going work

(Bartier, B., Dallard, Lomer, Mouawad '20+)



Hammer # 1 : If $\exists X$ "small" and $G[V \setminus X]$ contains many components, the graph can be reduced.

Hammer # 2 : If G contains a long geodesic path, then the graph can be reduced.

Hammer # 3 : If $\exists X$ "small" with no token on X and $G[V \setminus X]$ contains a long geodesic path, then the graph can be reduced.

Consequences : FPT algorithms for

- Bounded degree graphs.
- Planar graphs.
- Graphs of treewidth ≤ 4.
- Graphs of bounded treedepth.

What's next

For sparse graphs :

- Graphs of bounded treewidth ?
- Graph nowhere dense ?
- $K_{\ell,\ell}$-free graphs ?

For dense graphs :

- Chordal graphs ?
- Split graphs ?

Conclusion

Another model : Token Jumping (TJ)

$\Leftrightarrow$ a token can jump anywhere else in the graph.

Conclusion

Another model : Token Jumping (TJ)
$\Leftrightarrow$ a token can jump anywhere else in the graph.

- Most of the results of this talk holds in the TJ-model.
- Much simpler on sparse graphs (cograph, bounded expansion, $K_{\ell,\ell}$-free...).

Conclusion

Another model : Token Jumping (TJ)
$\Leftrightarrow$ a token can jump anywhere else in the graph.

- Most of the results of this talk holds in the TJ-model.
- Much simpler on sparse graphs (cograph, bounded expansion, $K_{\ell,\ell}$-free...).

Questions :

- TJ-Reachability in P on P_5-free graphs ?
  Potential maximum cliques ?

Conclusion

Another model : Token Jumping (TJ)
$\Leftrightarrow$ a token can jump anywhere else in the graph.

- Most of the results of this talk holds in the TJ-model.
- Much simpler on sparse graphs (cograph, bounded expansion, $K_{\ell,\ell}$-free...).

Questions :

- TJ-Reachability in P on P_5-free graphs ?
  Potential maximum cliques ?
- TJ-Reachability FPT on bipartite graphs ?
  ([Lokshtanov, Mouawad '19] The problem only is NP-complete and not PSPACE complete)

Conclusion

Another model : Token Jumping (TJ)
$\Leftrightarrow$ a token can jump anywhere else in the graph.

- Most of the results of this talk holds in the TJ-model.
- Much simpler on sparse graphs (cograph, bounded expansion, $K_{\ell,\ell}$-free...).

Questions :

- TJ-Reachability in P on P_5-free graphs ?
  Potential maximum cliques ?
- TJ-Reachability FPT on bipartite graphs ?
  ([Lokshtanov, Mouawad '19] The problem only is NP-complete and not PSPACE complete)

Thanks for your attention !