Fully Polynomial Parameterized Algorithms For the T-Path Packing Problem

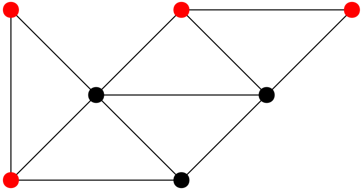
Narek Bojikian

Humboldt University of Berlin

12.12.2019

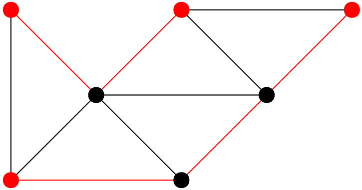
Introduction

- T-path packing.



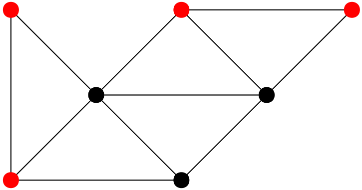
Introduction

- T-path packing.



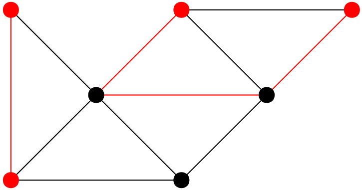
Introduction

- T-path packing.
- Odd T-path packing.



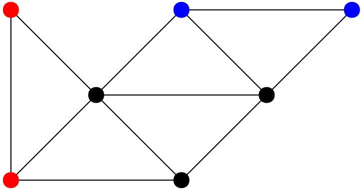
Introduction

- T-path packing.
- Odd T-path packing.



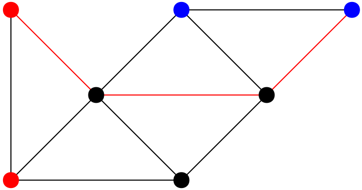
Introduction

- T-path packing.
- Odd T-path packing.
- $S - T$-path packing.



Introduction

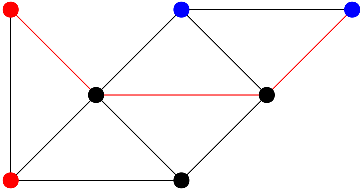
- T-path packing.
- Odd T-path packing.
- $S - T$-path packing.



Introduction

- T-path packing.
- Odd T-path packing.
- $S - T$-path packing.

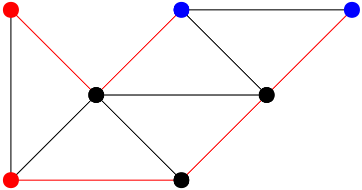
– Goal: Find maximum set of each.



Introduction

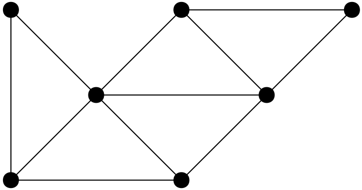
- T-path packing.
- Odd T-path packing.
- $S - T$-path packing.

– Goal: Find maximum set of each.



Introduction

- T-path packing.
- Odd T-path packing.
- $S - T$-path packing.
- – Goal: Find maximum set of each.
- Maximum Matching - $\nu(G)$.

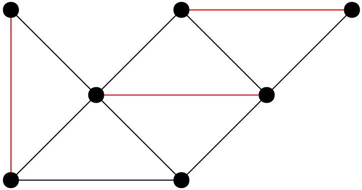


Introduction

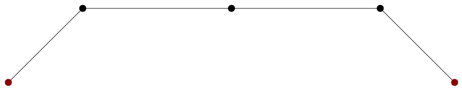
- T-path packing.
- Odd T-path packing.
- $S - T$-path packing.

– Goal: Find maximum set of each.

- Maximum Matching - $\nu(G)$.

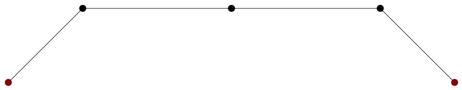


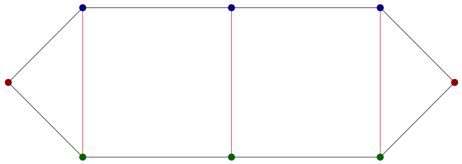
Reduction - T-Path Packing to Matching



Maximum-minimum Sätze über Graphen, T.Gallai, 1958

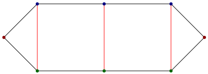
Reduction - T-Path Packing to Matching



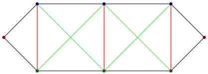


Maximum-minimum Sätze über Graphen, T.Gallai, 1958

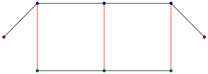
Reductions presented in the thesis



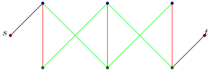
T-Path Packing Problem



T-Path Packing Problem



Odd T-Path Packing Problem



$S - T$-Path Packing Problem

Methods and results - Preserving properties

- Let $\mathcal{A}$ be an efficient **polynomial-time**$^{(*)}$ algorithm for matching in a class $\mathcal{C}$.

Methods and results - Preserving properties

- Let $\mathcal{A}$ be an efficient **polynomial-time**$^{(*)}$ algorithm for matching in a class $\mathcal{C}$.
- Let f be a reduction from a problem P to maximum matching.

Methods and results - Preserving properties

- Let $\mathcal{A}$ be an efficient **polynomial-time**$^{(*)}$ algorithm for matching in a class $\mathcal{C}$.
- Let f be a reduction from a problem P to maximum matching.
- Assume f can be computed efficiently and the size of the output is bounded in a **linear function**$^{(**)}$ in the size of G.

Methods and results - Preserving properties

- Let $\mathcal{A}$ be an efficient **polynomial-time**$^{(*)}$ algorithm for matching in a class $\mathcal{C}$.
- Let f be a reduction from a problem P to maximum matching.
- Assume f can be computed efficiently and the size of the output is bounded in a **linear function**$^{(**)}$ in the size of G.
- Assume $\mathcal{C}$ is closed under f.

Methods and results - Preserving properties

- Let $\mathcal{A}$ be an efficient **polynomial-time**$^{(*)}$ algorithm for matching in a class $\mathcal{C}$.
- Let f be a reduction from a problem P to maximum matching.
- Assume f can be computed efficiently and the size of the output is bounded in a **linear function**$^{(**)}$ in the size of G.
- Assume $\mathcal{C}$ is closed under f.

  - Then we get an efficient algorithm for P in $\mathcal{C}$.

Methods and results - Preserving properties

- Let $\mathcal{A}$ be an efficient **polynomial-time**$^{(*)}$ algorithm for matching in a class $\mathcal{C}$.
- Let f be a reduction from a problem P to maximum matching.
- Assume f can be computed efficiently and the size of the output is bounded in a **linear function**$^{(**)}$ in the size of G.
- Assume $\mathcal{C}$ is closed under f.
  - Then we get an efficient algorithm for P in $\mathcal{C}$.

A' - Efficient algorithm for P

Given a graph G. Compute $G' := f(G)$ and apply $\mathcal{A}$ on G'.

Methods and results - Preserving properties

- Let $\mathcal{A}$ be an efficient **polynomial-time**$^{(*)}$ algorithm for matching in a class $\mathcal{C}$.
- Let f be a reduction from a problem P to maximum matching.
- Assume f can be computed efficiently and the size of the output is bounded in a **linear function**$^{(**)}$ in the size of G.
- Assume $\mathcal{C}$ is closed under f.
  - Then we get an efficient algorithm for P in $\mathcal{C}$.

A' - Efficient algorithm for P

Given a graph G. Compute $G' := f(G)$ and apply $\mathcal{A}$ on G'.

$$\mathrm{time}_{A'}(G) = \mathrm{time}_f(G) + \mathrm{time}_A(G') \underset{(*,**)}{=} O(\mathrm{time}_f(G) + \mathrm{time}_A(G))$$

Methods and results - Preserving properties

- T-path packing (using the second reduction):
  - Tree-depth.
  - Tree-width.
  - s-plexes.

    The value at most doubles.
  - Modular-width.
  - Independence number.
  - Neighborhood diversity number.

    The value does not change.

    .. Running time $O(k(n + m))$
  - Strongly chordal -, Interval- and co-Comparability- graphs.

    The classes are closed under this reduction.

    .. Running time $O(n + m)$
  - Circular-arc graphs.

    The classes are closed under this reduction.

    .. Running time $O((n + m)\log(n))$

For n the number of the vertices and m the number edges in the graph G.

Methods and results - Preserving properties

- Odd T-path packing:
  - Neighborhood diversity number.
    - Bounded Replaceablity $O(k^2)$.

      .. Running time $O(k^2(n+m))$
- Odd T-path packing and $S-T$-path packing:
  - Tree-depth and tree-width.

    The value at most doubles.

    .. Running time $O(k(n+m))$
  - ~~Independence number - does not change.~~

Methods and results - Distance to triviality

- Vertex-deletion distance to a class - $d_{\mathcal{C}}(G)$.

Methods and results - Distance to triviality

- Vertex-deletion distance to a class - $d_{\mathcal{C}}(G)$.
- The reductions preserve the distance to triviality.

Methods and results - Distance to triviality

- Vertex-deletion distance to a class - $d_{\mathcal{C}}(G)$.
- The reductions preserve the distance to triviality.
  Given a graph G, a class $\mathcal{C}$ and any of the reductions f such that $\mathcal{C}$ is closed under f and $d_{\mathcal{C}}(G) \leq k$, then $d_{\mathcal{C}}(f(G)) \leq 2k$.

Methods and results - Distance to triviality

- Vertex-deletion distance to a class - $d_{\mathcal{C}}(G)$.
- The reductions preserve the distance to triviality.
  Given a graph G, a class $\mathcal{C}$ and any of the reductions f such that $\mathcal{C}$ is closed under f and $d_{\mathcal{C}}(G) \leq k$, then $d_{\mathcal{C}}(f(G)) \leq 2k$.
- The reductions preserve the distance to parameters values.

Methods and results - Distance to triviality

- Vertex-deletion distance to a class - $d_{\mathcal{C}}(G)$.
- The reductions preserve the distance to triviality.
  Given a graph G, a class $\mathcal{C}$ and any of the reductions f such that
  $\mathcal{C}$ is closed under f and $d_{\mathcal{C}}(G) \leq k$, then $d_{\mathcal{C}}(f(G)) \leq 2k$.
- The reductions preserve the distance to parameters values.
  Let ρ_G be the value of the parameter in G. Consider the class $\mathcal{C}_n := \{G : \rho_G \leq n\}$. For a reduction f, such that $f(C_n) \subseteq C_{g(n)}$ for some computeable function g,
  we get $d_{\mathcal{C}_{g(n)}}(f(G)) \leq 2d_{C_n}(G)$.

Methods and results - Distance to triviality

- Vertex-deletion distance to a class - $d_{\mathcal{C}}(G)$.
- The reductions preserve the distance to triviality.
  Given a graph G, a class $\mathcal{C}$ and any of the reductions f such that
  $\mathcal{C}$ is closed under f and $d_{\mathcal{C}}(G) \leq k$, then $d_{\mathcal{C}}(f(G)) \leq 2k$.
- The reductions preserve the distance to parameters values.
  Let ρ_G be the value of the parameter in G. Consider the class $\mathcal{C}_n := \{G : \rho_G \leq n\}$. For a reduction f, such that $f(C_n) \subseteq C_{g(n)}$ for some computeable function g,
  we get $d_{\mathcal{C}_{g(n)}}(f(G)) \leq 2d_{C_n}(G)$.
- If maximum matching can be solved efficiently in graphs with distance at most k to $\mathcal{C}$, so is T-path packing.

Methods and results - Distance to triviality

- T-Path Packing.
  - Neighborhood diversity number.
  - s-plexes.
  - Independence number.

.. Running time $O(\sqrt{d}k(n+m))$

[1] On adaptive algorithms for maximum matching, F.Hegerefeld and S.Kratsch, ICALP-2019.

Methods and results - Distance to triviality

- T-Path Packing.
  - Neighborhood diversity number.
  - s-plexes.
  - Independence number.

.. Running time $O(\sqrt{d}k(n+m))$

- Odd T-Path Packing.
  Neighborhood diversity number.

[1] On adaptive algorithms for maximum matching, F.Hegerefeld and S.Kratsch, ICALP-2019.

Methods and results - Distance to triviality

- T-Path Packing.
  - Neighborhood diversity number.
  - s-plexes.
  - Independence number.

.. Running time $O(\sqrt{d}k(n+m))$

- Odd T-Path Packing.
  Neighborhood diversity number.
  - The class of ℓ-Replaceable graphs - $\mathcal{R}[\ell]$.

[1] On adaptive algorithms for maximum matching, F.Hegerefeld and S.Kratsch, ICALP-2019.

Methods and results - Distance to triviality

- T-Path Packing.
  - Neighborhood diversity number.
  - s-plexes.
  - Independence number.

.. Running time $O(\sqrt{d}k(n+m))$

- Odd T-Path Packing.
  Neighborhood diversity number.
  - The class of ℓ-Replaceable graphs - $\mathcal{R}[\ell]$.
  - For $m := |E(G)|$, $l, d \in \mathbb{N}$, if $d_{\mathcal{R}[\ell]}(G) \leq d$,
    then $\nu(G)$ can be found in $O(\sqrt{d}\ell m)$ [1].

[1] On adaptive algorithms for maximum matching, F.Hegerefeld and S.Kratsch, ICALP-2019.

Methods and results - Distance to triviality

- T-Path Packing.
  - Neighborhood diversity number.
  - s-plexes.
  - Independence number.

.. Running time $O(\sqrt{d}k(n+m))$

- Odd T-Path Packing.
  Neighborhood diversity number.
  - The class of ℓ-Replaceable graphs - $\mathcal{R}[\ell]$.
  - For $m := |E(G)|$, $l, d \in \mathbb{N}$, if $d_{\mathcal{R}[\ell]}(G) \leq d$,
    then $\nu(G)$ can be found in $O(\sqrt{d}\ell m)$ [1].
  - For G' the graph resulting from G when we apply the second reduction. If the Neighborhood diversity number of G is at most k then G' is at most $O(k^2)$ replaceable.

[1] On adaptive algorithms for maximum matching, F.Hegerefeld and S.Kratsch, ICALP-2019.

Methods and results - Distance to triviality

- T-Path Packing.
  - Neighborhood diversity number.
  - s-plexes.
  - Independence number.

.. Running time $O(\sqrt{d}k(n+m))$

- Odd T-Path Packing.
  Neighborhood diversity number.
  - The class of ℓ-Replaceable graphs - $\mathcal{R}[\ell]$.
  - For $m := |E(G)|$, $l, d \in \mathbb{N}$, if $d_{\mathcal{R}[\ell]}(G) \leq d$,
    then $\nu(G)$ can be found in $O(\sqrt{d}\ell m)$ [1].
  - For G' the graph resulting from G when we apply the second reduction. If the Neighborhood diversity number of G is at most k then G' is at most $O(k^2)$ replaceable.

.. Running time $O(\sqrt{d}k^2(n+m))$

[1] On adaptive algorithms for maximum matching, F.Hegerefeld and S.Kratsch, ICALP-2019.

Methods and results - Simplify, solve and augment

- Sometimes parameters can be seen from
  a different perspective.

[2] For $\mathcal{T}$ the class of forests.

Methods and results - Simplify, solve and augment

- Sometimes parameters can be seen from
  a different perspective.

- The feedback vertex number of a graph
  is the size of the smallest set of vertices
  that intersects all cycles in the graph.

[2] For $\mathcal{T}$ the class of forests.

Methods and results - Simplify, solve and augment

- Sometimes parameters can be seen from
  a different perspective.
- The feedback vertex number of a graph
  is the size of the smallest set of vertices
  that intersects all cycles in the graph.
- Equivalently, the feedback vertex number of a graph G
  is the vertex-deletion distance of this graph
  from forests[2] - $d_{\mathcal{T}}(G)$.

[2] For $\mathcal{T}$ the class of forests.

Methods and results - Simplify, solve and augment

- Sometimes parameters can be seen from
  a different perspective.
  - The feedback vertex number of a graph
    is the size of the smallest set of vertices
    that intersects all cycles in the graph.
- Equivalently, the feedback vertex number of a graph G
  is the vertex-deletion distance of this graph
  from forests[2] - $d_{\mathcal{T}}(G)$.
- Let ρ be a parameter defined as
  - the vertex-deletion distance to a class $\mathcal{C}$.

[2] For $\mathcal{T}$ the class of forests.

Methods and results - Simplify, solve and augment

- Sometimes parameters can be seen from
  a different perspective.
- The feedback vertex number of a graph
  is the size of the smallest set of vertices
  that intersects all cycles in the graph.
- Equivalently, the feedback vertex number of a graph G
  is the vertex-deletion distance of this graph
  from forests[2] - $d_{\mathcal{T}}(G)$.
- Let ρ be a parameter defined as
  - the vertex-deletion distance to a class $\mathcal{C}$.
- Assume T-Path Packing admits an efficient algorithm $\mathcal{A}$ in $\mathcal{C}$.
  - Then we get an algorithm for G parameterized by ρ.

[2] For $\mathcal{T}$ the class of forests.

Methods and results - Simplify, solve and augment

A' - Parameterized algorithm

- Let S be a modulator in G and $H := G \setminus S$, i.e. $H \in \mathcal{C}$.

Methods and results - Simplify, solve and augment

A' - Parameterized algorithm

- Let S be a modulator in G and $H := G \setminus S$, i.e. $H \in \mathcal{C}$.
- Apply $\mathcal{A}$ on H.

Methods and results - Simplify, solve and augment

A' - Parameterized algorithm

- Let S be a modulator in G and $H := G \setminus S$, i.e. $H \in \mathcal{C}$.
- Apply $\mathcal{A}$ on H.
- Apply the first reduction on H to get H'.

Methods and results - Simplify, solve and augment

A' - Parameterized algorithm

- Let S be a modulator in G and $H := G \setminus S$, i.e. $H \in \mathcal{C}$.
- Apply $\mathcal{A}$ on H.
- Apply the first reduction on H to get H'.
- Find a maximum matching in H (using the reduction).

Methods and results - Simplify, solve and augment

A' - Parameterized algorithm

- Let S be a modulator in G and $H := G \setminus S$, i.e. $H \in \mathcal{C}$.
- Apply $\mathcal{A}$ on H.
- Apply the first reduction on H to get H'.
- Find a maximum matching in H (using the reduction).
- Turn H into G by adding S back to the graph
  - turns H' into G' by adding at most $2|S|$ vertices.

Methods and results - Simplify, solve and augment

A' - Parameterized algorithm

- Let S be a modulator in G and $H := G \setminus S$, i.e. $H \in \mathcal{C}$.
- Apply $\mathcal{A}$ on H.
- Apply the first reduction on H to get H'.
- Find a maximum matching in H (using the reduction).
- Turn H into G by adding S back to the graph
  - turns H' into G' by adding at most $2|S|$ vertices.
- Find a maximum matching in G'
  - by finding at most $2|S|$ augmenting paths.

Methods and results - Simplify, solve and augment

A' - Parameterized algorithm

- Let S be a modulator in G and $H := G \setminus S$, i.e. $H \in \mathcal{C}$.
- Apply $\mathcal{A}$ on H.
- Apply the first reduction on H to get H'.
- Find a maximum matching in H (using the reduction).
- Turn H into G by adding S back to the graph
  - turns H' into G' by adding at most $2|S|$ vertices.
- Find a maximum matching in G'
  - by finding at most $2|S|$ augmenting paths.
- Find a maximum T-path packing in G (using the reduction).

Methods and results - Simplify, solve and augment

A' - Parameterized algorithm

- Let S be a modulator in G and $H := G \setminus S$, i.e. $H \in \mathcal{C}$.
- Apply $\mathcal{A}$ on H.
- Apply the first reduction on H to get H'.
- Find a maximum matching in H (using the reduction).
- Turn H into G by adding S back to the graph
  - turns H' into G' by adding at most $2|S|$ vertices.
- Find a maximum matching in G'
  - by finding at most $2|S|$ augmenting paths.
- Find a maximum T-path packing in G (using the reduction).

$$\begin{aligned}\mathrm{time}_{A'}(G) &= O(n + m + \mathrm{time}_A(H) + (2|S|(n+m))) \\ &= O(\mathrm{time}_A(G \setminus S) + \rho(n+m)).\end{aligned}$$

For n the number of the vertices and m the number edges in the graph G.

Methods and results - Simplify, solve and augment

- Vertex Cover Number
  - simplify to an independent set.

  Running time $O(k(n+m)) \rightarrow O(\sqrt{k}(n+m))$[3]
- Feedback Vertex Number
  - simplify to a forest.

  .. Running time $O(k(n+m))$

Designed a dynamic programming algorithm for each of the problems in forests.

[3] suggested by Prof. Kratsch.