Even delta-matroids and the complexity of planar Boolean CSPs

Alexandr Kazda, Vladimir Kolmogorov, Michal Rolínek

IST AUSTRIA
Institute of Science and Technology

History

- Our world: CSP$(\{0,1\},\Gamma)$ where Γ contains constants $\{0\}$ and $\{1\}$.
- We limit the instance shape – each variable appears at most k times. For which Γs do we get easier CSP?
- T. Feder: Fanout limitations on constraint systems, 2001.
- V. Dalmau, D. Ford: Generalized satisfiability with k occurences per variable: A study through delta-matroid parity, 2003.
- Only interesting case: $k = 2$.

History

- Our world: CSP$(\{0,1\},\Gamma)$ where Γ contains constants $\{0\}$ and $\{1\}$.
- We limit the instance shape – each variable appears at most k times. For which Γs do we get easier CSP?
- T. Feder: Fanout limitations on constraint systems, 2001.
- V. Dalmau, D. Ford: Generalized satisfiability with k occurences per variable: A study through delta-matroid parity, 2003.
- Only interesting case: $k = 2$.

History

- Our world: CSP($\{0,1\},\Gamma$) where Γ contains constants $\{0\}$ and $\{1\}$.
- We limit the instance shape – each variable appears at most k times. For which Γs do we get easier CSP?
- T. Feder: Fanout limitations on constraint systems, 2001.
- V. Dalmau, D. Ford: Generalized satisfiability with k occurences per variable: A study through delta-matroid parity, 2003.
- Only interesting case: $k = 2$.

History

- Our world: CSP$(\{0,1\},\Gamma)$ where Γ contains constants $\{0\}$ and $\{1\}$.
- We limit the instance shape – each variable appears at most k times. For which Γs do we get easier CSP?
- T. Feder: Fanout limitations on constraint systems, 2001.
- V. Dalmau, D. Ford: Generalized satisfiability with k occurences per variable: A study through delta-matroid parity, 2003.
- Only interesting case: $k = 2$.

History

- Our world: CSP($\{0,1\},\Gamma$) where Γ contains constants $\{0\}$ and $\{1\}$.
- We limit the instance shape – each variable appears at most k times. For which Γs do we get easier CSP?
- T. Feder: Fanout limitations on constraint systems, 2001.
- V. Dalmau, D. Ford: Generalized satisfiability with k occurences per variable: A study through delta-matroid parity, 2003.
- Only interesting case: $k = 2$.

History

- Our world: CSP($\{0,1\},\Gamma$) where Γ contains constants $\{0\}$ and $\{1\}$.
- We limit the instance shape – each variable appears at most k times. For which Γs do we get easier CSP?
- T. Feder: Fanout limitations on constraint systems, 2001.
- V. Dalmau, D. Ford: Generalized satisfiability with k occurences per variable: A study through delta-matroid parity, 2003.
- Only interesting case: $k = 2$.

Edge CSP

- Wlog each variable appears in exactly two constrains.
- We can draw instances of this CSP as graphs with variables = edges.
- Some people call this binary CSP, we prefer edge CSP.
- Feder: The only new case is when all relations in Γ are Δ-matroids.

Edge CSP

- Wlog each variable appears in exactly two constrains.
- We can draw instances of this CSP as graphs with variables = edges.
- Some people call this binary CSP, we prefer edge CSP.
- Feder: The only new case is when all relations in Γ are Δ-matroids.

Edge CSP

- Wlog each variable appears in exactly two constrains.
- We can draw instances of this CSP as graphs with variables = edges.
- Some people call this binary CSP, we prefer edge CSP.
- Feder: The only new case is when all relations in Γ are Δ-matroids.

Edge CSP

- Wlog each variable appears in exactly two constrains.
- We can draw instances of this CSP as graphs with variables = edges.
- Some people call this binary CSP, we prefer edge CSP.
- Feder: The only new case is when all relations in Γ are Δ-matroids.

Edge CSP

- Wlog each variable appears in exactly two constrains.
- We can draw instances of this CSP as graphs with variables = edges.
- Some people call this binary CSP, we prefer edge CSP.
- Feder: The only new case is when all relations in Γ are Δ-matroids.

Δ-matroids

- AKA "generalized matroids"
- $R \neq \emptyset$ is an even Δ-matroid if all tuples in R have the same parity and for all $\alpha, \beta \in R$ and for all u variables such that $\alpha(u) \neq \beta(u)$ there exists $v \neq u$ such that $\alpha(v) \neq \beta(v)$ and $\alpha \oplus u \oplus v \in R$:
- Δ-matroids: No parity restriction, enought to have $\alpha \oplus u \in R$ instead of $\alpha \oplus u \oplus v \in R$.

Δ-matroids

- AKA "generalized matroids"
- $R \neq \emptyset$ is an even Δ-matroid if all tuples in R have the same parity and for all $\alpha, \beta \in R$ and for all u variables such that $\alpha(u) \neq \beta(u)$ there exists $v \neq u$ such that $\alpha(v) \neq \beta(v)$ and $\alpha \oplus u \oplus v \in R$:
- Δ-matroids: No parity restriction, enought to have $\alpha \oplus u \in R$ instead of $\alpha \oplus u \oplus v \in R$.

Δ-matroids

- AKA "generalized matroids"
- $R \neq \emptyset$ is an even Δ-matroid if all tuples in R have the same parity and for all $\alpha, \beta \in R$ and for all u variables such that $\alpha(u) \neq \beta(u)$ there exists $v \neq u$ such that $\alpha(v) \neq \beta(v)$ and $\alpha \oplus u \oplus v \in R$:

$$\begin{array}{lcccccc} \alpha & = & 0 & 0 & 0 & 1 & 1 \\ \beta & = & 1 & 1 & 1 & 0 & 1 \end{array}$$

- Δ-matroids: No parity restriction, enought to have $\alpha \oplus u \in R$ instead of $\alpha \oplus u \oplus v \in R$.

Δ-matroids

- AKA "generalized matroids"
- $R \neq \emptyset$ is an even Δ-matroid if all tuples in R have the same parity and for all $\alpha, \beta \in R$ and for all u variables such that $\alpha(u) \neq \beta(u)$ there exists $v \neq u$ such that $\alpha(v) \neq \beta(v)$ and $\alpha \oplus u \oplus v \in R$:

$$\begin{array}{lcccccc} \alpha & = & \mathbf{0} & 0 & 0 & 1 & 1 \\ \beta & = & \mathbf{1} & 1 & 1 & 0 & 1 \end{array}$$

- Δ-matroids: No parity restriction, enought to have $\alpha \oplus u \in R$ instead of $\alpha \oplus u \oplus v \in R$.

Δ-matroids

- AKA "generalized matroids"
- $R \neq \emptyset$ is an even Δ-matroid if all tuples in R have the same parity and for all $\alpha, \beta \in R$ and for all u variables such that $\alpha(u) \neq \beta(u)$ there exists $v \neq u$ such that $\alpha(v) \neq \beta(v)$ and $\alpha \oplus u \oplus v \in R$:

$$\begin{array}{lcccccc} \alpha & = & \mathbf{0} & \mathbf{0} & 0 & 1 & 1 \\ \beta & = & \mathbf{1} & \mathbf{1} & 1 & 0 & 1 \end{array}$$

- Δ-matroids: No parity restriction, enought to have $\alpha \oplus u \in R$ instead of $\alpha \oplus u \oplus v \in R$.

Δ-matroids

- AKA "generalized matroids"
- $R \neq \emptyset$ is an even Δ-matroid if all tuples in R have the same parity and for all $\alpha, \beta \in R$ and for all u variables such that $\alpha(u) \neq \beta(u)$ there exists $v \neq u$ such that $\alpha(v) \neq \beta(v)$ and $\alpha \oplus u \oplus v \in R$:

$$\begin{array}{rcccccc} \alpha & = & \mathbf{0} & \mathbf{0} & 0 & 1 & 1 \\ \beta & = & \mathbf{1} & \mathbf{1} & 1 & 0 & 1 \\ \alpha \oplus u \oplus v & = & \mathbf{1} & \mathbf{1} & 0 & 1 & 1 \end{array}$$

- Δ-matroids: No parity restriction, enought to have $\alpha \oplus u \in R$ instead of $\alpha \oplus u \oplus v \in R$.

Δ-matroids

- AKA "generalized matroids"
- $R \neq \emptyset$ is an even Δ-matroid if all tuples in R have the same parity and for all $\alpha, \beta \in R$ and for all u variables such that $\alpha(u) \neq \beta(u)$ there exists $v \neq u$ such that $\alpha(v) \neq \beta(v)$ and $\alpha \oplus u \oplus v \in R$:

$$\begin{array}{ccccccc} \alpha & = & \mathbf{0} & \mathbf{0} & 0 & 1 & 1 \\ \beta & = & \mathbf{1} & \mathbf{1} & 1 & 0 & 1 \\ \alpha \oplus u \oplus v & = & \mathbf{1} & \mathbf{1} & 0 & 1 & 1 \end{array}$$

- Δ-matroids: No parity restriction, enought to have $\alpha \oplus u \in R$ instead of $\alpha \oplus u \oplus v \in R$.

Good and bad news about Δ-matroids

- Intersection of two even Δ-matroids need not be a Δ-matroid:

$$\begin{Bmatrix} (0 & 0 & 0 & 0) \\ (1 & 1 & 0 & 0) \\ (0 & 0 & 1 & 1) \\ (1 & 1 & 1 & 1) \end{Bmatrix} \cap \begin{Bmatrix} (0 & 0 & 0 & 0) \\ (1 & 0 & 1 & 0) \\ (0 & 1 & 0 & 1) \\ (1 & 1 & 1 & 1) \end{Bmatrix} = \begin{Bmatrix} (0 & 0 & 0 & 0) \\ (1 & 1 & 1 & 1) \end{Bmatrix}$$

- If there is any way to use polymorphisms here, we did not find it.
- However, (even) Δ-matroids are closed under primitive positive definitions where each bound variable appears exactly twice and each free variable exactly once.

Good and bad news about Δ-matroids

- Intersection of two even Δ-matroids need not be a Δ-matroid:

$$\begin{Bmatrix} (0 & 0 & 0 & 0) \\ (1 & 1 & 0 & 0) \\ (0 & 0 & 1 & 1) \\ (1 & 1 & 1 & 1) \end{Bmatrix} \cap \begin{Bmatrix} (0 & 0 & 0 & 0) \\ (1 & 0 & 1 & 0) \\ (0 & 1 & 0 & 1) \\ (1 & 1 & 1 & 1) \end{Bmatrix} = \begin{Bmatrix} (0 & 0 & 0 & 0) \\ (1 & 1 & 1 & 1) \end{Bmatrix}$$

- If there is any way to use polymorphisms here, we did not find it.
- However, (even) Δ-matroids are closed under primitive positive definitions where each bound variable appears exactly twice and each free variable exactly once.

Good and bad news about Δ-matroids

- Intersection of two even Δ-matroids need not be a Δ-matroid:

$$\begin{Bmatrix} (0 & 0 & 0 & 0) \\ (1 & 1 & 0 & 0) \\ (0 & 0 & 1 & 1) \\ (1 & 1 & 1 & 1) \end{Bmatrix} \cap \begin{Bmatrix} (0 & 0 & 0 & 0) \\ (1 & 0 & 1 & 0) \\ (0 & 1 & 0 & 1) \\ (1 & 1 & 1 & 1) \end{Bmatrix} = \begin{Bmatrix} (0 & 0 & 0 & 0) \\ (1 & 1 & 1 & 1) \end{Bmatrix}$$

- If there is any way to use polymorphisms here, we did not find it.
- However, (even) Δ-matroids are closed under primitive positive definitions where each bound variable appears exactly twice and each free variable exactly once.

Good and bad news about Δ-matroids

- Intersection of two even Δ-matroids need not be a Δ-matroid:

$$\left\{\begin{matrix}(0 & 0 & 0 & 0)\\(1 & 1 & 0 & 0)\\(0 & 0 & 1 & 1)\\(1 & 1 & 1 & 1)\end{matrix}\right\} \cap \left\{\begin{matrix}(0 & 0 & 0 & 0)\\(1 & 0 & 1 & 0)\\(0 & 1 & 0 & 1)\\(1 & 1 & 1 & 1)\end{matrix}\right\} = \left\{\begin{matrix}(0 & 0 & 0 & 0)\\(1 & 1 & 1 & 1)\end{matrix}\right\}$$

- If there is any way to use polymorphisms here, we did not find it.
- However, (even) Δ-matroids are closed under primitive positive definitions where each bound variable appears exactly twice and each free variable exactly once.

Edge CSPs generalize perfect matchings

- Perfect matchings in graphs: Given G, assign 0 or 1 to each edge so that each vertex of G is incident to exactly one edge labelled by 1.
- Known to be polynomial (J. Edmonds, 1965).
- Perfect matchings correspond to edge CSP with constraints of the form

$$\begin{array}{c} \{(1 \quad 0 \quad 0 \quad \dots \quad 0) \\ (0 \quad 1 \quad 0 \quad \dots \quad 0) \\ (0 \quad 0 \quad 1 \quad \dots \quad 0) \\ \ddots \\ (0 \quad 0 \quad 0 \quad \dots \quad 1)\} \end{array}$$

- These are even Δ-matroids!

Edge CSPs generalize perfect matchings

- Perfect matchings in graphs: Given G, assign 0 or 1 to each edge so that each vertex of G is incident to exactly one edge labelled by 1.
- Known to be polynomial (J. Edmonds, 1965).
- Perfect matchings correspond to edge CSP with constraints of the form

$$\begin{array}{c}\{(1 \quad 0 \quad 0 \quad \dots \quad 0)\\ (0 \quad 1 \quad 0 \quad \dots \quad 0)\\ (0 \quad 0 \quad 1 \quad \dots \quad 0)\\ \ddots \\ (0 \quad 0 \quad 0 \quad \dots \quad 1)\}\end{array}$$

- These are even Δ-matroids!

Edge CSPs generalize perfect matchings

- Perfect matchings in graphs: Given G, assign 0 or 1 to each edge so that each vertex of G is incident to exactly one edge labelled by 1.
- Known to be polynomial (J. Edmonds, 1965).
- Perfect matchings correspond to edge CSP with constraints of the form

$$\begin{array}{l} \{(1 \quad 0 \quad 0 \quad \dots \quad 0) \\ \ (0 \quad 1 \quad 0 \quad \dots \quad 0) \\ \ (0 \quad 0 \quad 1 \quad \dots \quad 0) \\ \qquad\qquad\qquad \ddots \\ \ (0 \quad 0 \quad 0 \quad \dots \quad 1)\} \end{array}$$

- These are even Δ-matroids!

Edge CSPs generalize perfect matchings

- Perfect matchings in graphs: Given G, assign 0 or 1 to each edge so that each vertex of G is incident to exactly one edge labelled by 1.
- Known to be polynomial (J. Edmonds, 1965).
- Perfect matchings correspond to edge CSP with constraints of the form

$$\begin{array}{l} \{(1 \quad 0 \quad 0 \quad \dots \quad 0) \\ (0 \quad 1 \quad 0 \quad \dots \quad 0) \\ (0 \quad 0 \quad 1 \quad \dots \quad 0) \\ \qquad\qquad\quad \ddots \\ (0 \quad 0 \quad 0 \quad \dots \quad 1)\} \end{array}$$

- These are even Δ-matroids!

Edge CSPs generalize perfect matchings

- Perfect matchings in graphs: Given G, assign 0 or 1 to each edge so that each vertex of G is incident to exactly one edge labelled by 1.
- Known to be polynomial (J. Edmonds, 1965).
- Perfect matchings correspond to edge CSP with constraints of the form

$$\begin{array}{ccccc} \{(1 & 0 & 0 & \ldots & 0) \\ (0 & 1 & 0 & \ldots & 0) \\ (0 & 0 & 1 & \ldots & 0) \\ & & & \ddots & \\ (0 & 0 & 0 & \ldots & 1)\} \end{array}$$

- These are even Δ-matroids!

Planar CSPs

- Z. Dvořák and M. Kupec: On Planar Boolean CSP, 2015.
- CSP($\{0, 1\}, \Gamma$) with incidence graphs of instances planar.
- Constraints – faces of a planar graph, variables – vertices.
- Dvořák and Kupec show that all interesting cases of planar CSP can be reduced to edge CSP with Δ-matroid constraints.
- If there is a polynomial algorithm for edge CSP with even Δ-matroid constraints, we have a dichotomy for planar CSP.

Planar CSPs

- Z. Dvořák and M. Kupec: On Planar Boolean CSP, 2015.
- $\mathrm{CSP}(\{0,1\},\Gamma)$ with incidence graphs of instances planar.
- Constraints – faces of a planar graph, variables – vertices.
- Dvořák and Kupec show that all interesting cases of planar CSP can be reduced to edge CSP with Δ-matroid constraints.
- If there is a polynomial algorithm for edge CSP with even Δ-matroid constraints, we have a dichotomy for planar CSP.

Planar CSPs

- Z. Dvořák and M. Kupec: On Planar Boolean CSP, 2015.
- $\mathrm{CSP}(\{0, 1\}, \Gamma)$ with incidence graphs of instances planar.
- Constraints – faces of a planar graph, variables – vertices.
- Dvořák and Kupec show that all interesting cases of planar CSP can be reduced to edge CSP with Δ-matroid constraints.
- If there is a polynomial algorithm for edge CSP with even Δ-matroid constraints, we have a dichotomy for planar CSP.

Planar CSPs

- Z. Dvořák and M. Kupec: On Planar Boolean CSP, 2015.
- $\mathrm{CSP}(\{0,1\},\Gamma)$ with incidence graphs of instances planar.
- Constraints – faces of a planar graph, variables – vertices.
- Dvořák and Kupec show that all interesting cases of planar CSP can be reduced to edge CSP with Δ-matroid constraints.
- If there is a polynomial algorithm for edge CSP with even Δ-matroid constraints, we have a dichotomy for planar CSP.

Planar CSPs

- Z. Dvořák and M. Kupec: On Planar Boolean CSP, 2015.
- $\mathrm{CSP}(\{0,1\},\Gamma)$ with incidence graphs of instances planar.
- Constraints – faces of a planar graph, variables – vertices.
- Dvořák and Kupec show that all interesting cases of planar CSP can be reduced to edge CSP with Δ-matroid constraints.
- If there is a polynomial algorithm for edge CSP with even Δ-matroid constraints, we have a dichotomy for planar CSP.

Planar CSPs

- Z. Dvořák and M. Kupec: On Planar Boolean CSP, 2015.
- CSP($\{0, 1\}, \Gamma$) with incidence graphs of instances planar.
- Constraints – faces of a planar graph, variables – vertices.
- Dvořák and Kupec show that all interesting cases of planar CSP can be reduced to edge CSP with Δ-matroid constraints.
- If there is a polynomial algorithm for edge CSP with even Δ-matroid constraints, we have a dichotomy for planar CSP.

Our strategy

- We generalize Edmond's blossom algorithm for perfect matchings.
- Edge labeling f assigns 0 or 1 to each half-edge: Pair $\{v, C\}$ where v lies in constraint C so that all constraints are satisfied.
- Variable is consistent in f if both half edges corresponding to v have the same labels.
- Edge labeling with all variables consistent = a solution of the instance.
- We want to augment a given labeling f: Find g labeling with fewer inconsistencies.
- If f is an edge labeling that can be improved, there is an augmenting f-walk p from one inconsistent variable to another.

Our strategy

- We generalize Edmond's blossom algorithm for perfect matchings.
- Edge labeling f assigns 0 or 1 to each half-edge: Pair $\{v, C\}$ where v lies in constraint C so that all constraints are satisfied.
- Variable is consistent in f if both half edges corresponding to v have the same labels.
- Edge labeling with all variables consistent = a solution of the instance.
- We want to augment a given labeling f: Find g labeling with fewer inconsistencies.
- If f is an edge labeling that can be improved, there is an augmenting f-walk p from one inconsistent variable to another.

Our strategy

- We generalize Edmond's blossom algorithm for perfect matchings.
- Edge labeling f assigns 0 or 1 to each half-edge: Pair $\{v, C\}$ where v lies in constraint C so that all constraints are satisfied.
- Variable is consistent in f if both half edges corresponding to v have the same labels.
- Edge labeling with all variables consistent = a solution of the instance.
- We want to augment a given labeling f: Find g labeling with fewer inconsistencies.
- If f is an edge labeling that can be improved, there is an augmenting f-walk p from one inconsistent variable to another.

Our strategy

- We generalize Edmond's blossom algorithm for perfect matchings.
- Edge labeling f assigns 0 or 1 to each half-edge: Pair $\{v, C\}$ where v lies in constraint C so that all constraints are satisfied.
- Variable is consistent in f if both half edges corresponding to v have the same labels.
- Edge labeling with all variables consistent = a solution of the instance.
- We want to augment a given labeling f: Find g labeling with fewer inconsistencies.
- If f is an edge labeling that can be improved, there is an augmenting f-walk p from one inconsistent variable to another.

Our strategy

- We generalize Edmond's blossom algorithm for perfect matchings.
- Edge labeling f assigns 0 or 1 to each half-edge: Pair $\{v, C\}$ where v lies in constraint C so that all constraints are satisfied.
- Variable is consistent in f if both half edges corresponding to v have the same labels.
- Edge labeling with all variables consistent = a solution of the instance.
- We want to augment a given labeling f: Find g labeling with fewer inconsistencies.
- If f is an edge labeling that can be improved, there is an augmenting f-walk p from one inconsistent variable to another.

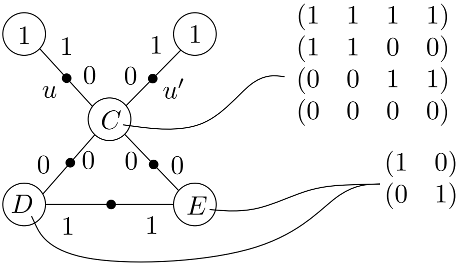
Our strategy

- We generalize Edmond's blossom algorithm for perfect matchings.
- Edge labeling f assigns 0 or 1 to each half-edge: Pair $\{v, C\}$ where v lies in constraint C so that all constraints are satisfied.
- Variable is consistent in f if both half edges corresponding to v have the same labels.
- Edge labeling with all variables consistent = a solution of the instance.
- We want to augment a given labeling f: Find g labeling with fewer inconsistencies.
- If f is an edge labeling that can be improved, there is an augmenting f-walk p from one inconsistent variable to another.

Our strategy

- We generalize Edmond's blossom algorithm for perfect matchings.
- Edge labeling f assigns 0 or 1 to each half-edge: Pair $\{v, C\}$ where v lies in constraint C so that all constraints are satisfied.
- Variable is consistent in f if both half edges corresponding to v have the same labels.
- Edge labeling with all variables consistent $=$ a solution of the instance.
- We want to augment a given labeling f: Find g labeling with fewer inconsistencies.
- If f is an edge labeling that can be improved, there is an augmenting f-walk p from one inconsistent variable to another.

Example



Sketch of the algorithm

- Take f, search from all inconsistent variables, building a forest of visited variables and constraints.
- If we can find f-walks $u \dots Cv$ and $u' \dots Dv$ for u, u' inconsistent, we can augment and make u, u' consistent.
- If we find f-walks $u \dots Cv$ and $u \dots Dv$, we have found a blossom. This we contract and re-run the algorithm on a smaller instance.
- If we don't find any of the above, then f can not be augmented.

Sketch of the algorithm

- Take f, search from all inconsistent variables, building a forest of visited variables and constraints.
- If we can find f-walks $u \dots Cv$ and $u' \dots Dv$ for u, u' inconsistent, we can augment and make u, u' consistent.
- If we find f-walks $u \dots Cv$ and $u \dots Dv$, we have found a blossom. This we contract and re-run the algorithm on a smaller instance.
- If we don't find any of the above, then f can not be augmented.

Sketch of the algorithm

- Take f, search from all inconsistent variables, building a forest of visited variables and constraints.
- If we can find f-walks $u \dots Cv$ and $u' \dots Dv$ for u, u' inconsistent, we can augment and make u, u' consistent.
- If we find f-walks $u \dots Cv$ and $u \dots Dv$, we have found a blossom. This we contract and re-run the algorithm on a smaller instance.
- If we don't find any of the above, then f can not be augmented.

Sketch of the algorithm

- Take f, search from all inconsistent variables, building a forest of visited variables and constraints.
- If we can find f-walks $u \dots Cv$ and $u' \dots Dv$ for u, u' inconsistent, we can augment and make u, u' consistent.
- If we find f-walks $u \dots Cv$ and $u \dots Dv$, we have found a blossom. This we contract and re-run the algorithm on a smaller instance.
- If we don't find any of the above, then f can not be augmented.

Sketch of the algorithm

- Take f, search from all inconsistent variables, building a forest of visited variables and constraints.
- If we can find f-walks $u \dots Cv$ and $u' \dots Dv$ for u, u' inconsistent, we can augment and make u, u' consistent.
- If we find f-walks $u \dots Cv$ and $u \dots Dv$, we have found a blossom. This we contract and re-run the algorithm on a smaller instance.
- If we don't find any of the above, then f can not be augmented.

Consequences and future work

- We have finished the classification started by Dvořák and Kupec.
- To get dichotomy for edge CSPs, all that is needed is to generalize our argument from even Δ-matroids to all Δ-matroids.
- We can go beyond even Δ-matroids and cover many previously known polynomial classes, but there still remains a large gap.
- We are now begining to look at valued version of edge CSP for even Δ-matroids.
- Generalization to value sets larger than 2 is going to be hard.

Consequences and future work

- We have finished the classification started by Dvořák and Kupec.
- To get dichotomy for edge CSPs, all that is needed is to generalize our argument from even Δ-matroids to all Δ-matroids.
- We can go beyond even Δ-matroids and cover many previously known polynomial classes, but there still remains a large gap.
- We are now begining to look at valued version of edge CSP for even Δ-matroids.
- Generalization to value sets larger than 2 is going to be hard.

Consequences and future work

- We have finished the classification started by Dvořák and Kupec.
- To get dichotomy for edge CSPs, all that is needed is to generalize our argument from even Δ-matroids to all Δ-matroids.
- We can go beyond even Δ-matroids and cover many previously known polynomial classes, but there still remains a large gap.
- We are now begining to look at valued version of edge CSP for even Δ-matroids.
- Generalization to value sets larger than 2 is going to be hard.

Consequences and future work

- We have finished the classification started by Dvořák and Kupec.
- To get dichotomy for edge CSPs, all that is needed is to generalize our argument from even Δ-matroids to all Δ-matroids.
- We can go beyond even Δ-matroids and cover many previously known polynomial classes, but there still remains a large gap.
- We are now begining to look at valued version of edge CSP for even Δ-matroids.
- Generalization to value sets larger than 2 is going to be hard.

Consequences and future work

- We have finished the classification started by Dvořák and Kupec.
- To get dichotomy for edge CSPs, all that is needed is to generalize our argument from even Δ-matroids to all Δ-matroids.
- We can go beyond even Δ-matroids and cover many previously known polynomial classes, but there still remains a large gap.
- We are now begining to look at valued version of edge CSP for even Δ-matroids.
- Generalization to value sets larger than 2 is going to be hard.

Consequences and future work

- We have finished the classification started by Dvořák and Kupec.
- To get dichotomy for edge CSPs, all that is needed is to generalize our argument from even Δ-matroids to all Δ-matroids.
- We can go beyond even Δ-matroids and cover many previously known polynomial classes, but there still remains a large gap.
- We are now begining to look at valued version of edge CSP for even Δ-matroids.
- Generalization to value sets larger than 2 is going to be hard.

Thank you for your attention.