Radovan Červený

cervera3@fit.cvut.cz

presented:

Hitting paths in graphs

Radovan Červený, Ondřej Suchý

https://doi.org/10.1142/S0218196704001700

5-Path Vertex Cover, 5-PVC	
Input:	A graph $G = (V, E)$, an integer $k \in Z_0^+$.
Output:	A set $F \subseteq V$, such that $\lvert F\rvert \leq k$ and $G \setminus F$ is a P_5-free graph.

Definition 1. A *star* is a graph S with vertices $V(S) = \{s\} \cup \{l_1, \ldots, l_k\}$, $|V(S)| \geq 4$ and edges $E(S) = \{\{s, l_i\} \mid i \in \{1, \ldots, k\}\}$. Vertex s is called a center, vertices $L = l_1, \ldots, l_k$ are called leaves.

Definition 2. A *star with a triangle* is a graph $S^{\triangle}$ with vertices $V(S^{\triangle}) = \{s, t_1, t_2\} \cup \{l_1, \ldots, l_k\}$, $|V(S^{\triangle})| \geq 4$ and edges $E(S^{\triangle}) = \{\{s, t_1\}, \{s, t_2\}, \{t_1, t_2\}\} \cup \{\{s, l_i\} \mid i \in \{1, \ldots, k\}\}$. Vertex s is called a center, vertices $T = \{t_1, t_2\}$ are called triangle vertices and vertices $L = l_1, \ldots, l_k$ are called leaves.

Definition 3. A *di-star* is a graph D with vertices $V(D) = \{s, s'\} \cup \{l_1, \ldots, l_k\} \cup \{l'_1, \ldots, l'_m\}$, $|V(D)| \geq 4$, $k \geq 1, m \geq 1$ and edges $E(D) = \{\{s, s'\}\} \cup \{\{s, l_i\} \mid i \in \{1, \ldots, k\}\} \cup \{\{s', l'_j\} \mid j \in \{1, \ldots, m\}\}$. Vertices s, s' are called centers, vertices $L = \{l_1, \ldots, l_k\}$ and $L' = \{l'_1, \ldots, l'_m\}$ are called leaves.

Definition 4. A P_5-free bipartition of graph $G = (V, E)$ is a pair (V_1, V_2) such that $V = V_1 \cup V_2, V_1 \cap V_2 = \emptyset$ and $G[V_1], G[V_2]$ are P_5-free.

5-PVC with P_5-free Bipartition, 5-PVCwB	
Input:	A graph $G = (V, E)$ with P_5-free bipartition (V_1, V_2), an integer $k \in Z_0^+$.
Output:	A set $F \subseteq V_2$, such that $\lvert F\rvert \leq k$ and $G \setminus F$ is a P_5-free graph.

Lemma 1. *If a connected graph is P_5-free and has more than 5 vertices, then it is a star, a star with a triangle, or a di-star.*

Lemma 2. *Assume that the Rules (R0) – (R2) are not applicable. Then for each vertex $v \in V(G)$ there exists a P_5 in G that uses v; every P_5 in G uses exactly one red vertex; and there are only isolated vertices in $G[V_1]$.*

Lemma 3. *Assume that the Rules (R0) – (R3) are not applicable. Let v be a blue vertex to which at least two red vertices are connected and let C_v be a connected component of $G[V_2]$ which contains v. Then for each red vertex w connected to v we have that $N(w) \subseteq V(C_v)$.*

Lemma 4. *Let X be a subset of V_2 such that $N(X) \cap V_1 = \emptyset$ and $|N(X) \cap V_2| = 1$. If there exists a solution F such that $F \cap X \neq \emptyset$, then there exists a solution F' such that $F' \cap X = \emptyset$ and $|F'| \leq |F|$.*

Lemma 5. *Let x, y be blue vertices that are symmetric. Let F be a solution and $x \in F$. Then at least one of the following holds:*

(1) $y \in F$

(2) $F' = (F \setminus \{x\}) \cup \{y\}$ is a solution

Lemma 6. *Let C be a connected component of $G[V_2]$ and $X = V(C) \cap N(V_1)$. Let F be a solution that deletes at least $|X|$ vertices in C. Then $F' = (F \setminus V(C)) \cup X$ is also a solution and $|F'| \leq |F|$.*

Rule (R0). This rule stops the recursion of DISJOINT_R. It has three stopping conditions:

1. If $k < 0$, return *no solution*;
2. else if G is P_5-free, return F;
3. else if $k = 0$, return *no solution*.

Rule (R1). Let $v \in V(G)$ be a vertex such that there is no P_5 in G that uses v. Then remove v from G.
Rule (R2). Let P be a P_5 in G with $X = V(P) \cap V_2$ such that $|X| \leq 3$. Then branch on $\langle x_1 \mid x_2 \mid \ldots \rangle, x_i \in X$, i.e. branch on the blue vertices of P.
Rule (R3). Let v be an isolated vertex in $G[V_2]$ and let $P = (v, w, x, y, z)$ be a P_5 where w is a red vertex. Then branch on $\langle x \mid y \mid z \rangle$.
Rule (R17). Let there be a di-star D and the two red vertices w, w' connected to D are connected to leaves l_1, l'_1, respectively, and both centers have degree exactly two. Then branch on $\langle l_1 \mid l'_1 \rangle$.

Illustrative pseudocode of iterative compression

```
1: procedure ALGO(G = (V, E), k)
2:     V' ← ∅, F ← ∅
3:     while V \ V' ≠ ∅ do
4:         with v ∈ V \ V'
5:         V' ← V' ∪ {v}, F ← F ∪ {v}
6:         if |F| = k + 1 then
7:             F̂ ← no solution
8:             for X ⊊ F do
9:                 Y ← F \ X
10:                if G[Y] is P5-free then
11:                    G' ← G[V'] \ X
12:                    F' ← DISJOINT(G', Y, V(G') \ Y, |Y| − 1)
13:                    if F' ≠ no solution then
14:                        F̂ = X ∪ F'
15:                        break
16:                    end if
17:                end if
18:            end for
19:            if F̂ ≠ no solution then
20:                F ← F̂
21:            else
22:                return no solution
23:            end if
24:        end if
25:    end while
26:    return F
27: end procedure
```

Illustrative pseudocode of the recursive procedure

```
1: procedure DISJOINT(G, V1, V2, k)
2:     return DISJOINT_R(G, V1, V2, ∅, k)
3: end procedure

4: procedure DISJOINT_R(G, V1, V2, F, k)
5:     F_result ← no solution
6:     R ← the first rule that is applicable
7:     if R is (R0) then
8:         F_result ← either F or no solution based on which stopping condition
                      of (R0) was triggered
9:     else if R is a reduction rule then
10:        let G', V1', V2' be simplified by R and let X be the vertices that R
           wants to add to F
11:        F_result ← DISJOINT_R(G', V1', V2', F ∪ X, k − |X|)
12:    else
13:        let the branches of R be ⟨ X1 | X2 | ... | Xl ⟩
14:        for i ← 1, ..., l do
15:            F_candidate ← DISJOINT_R(G \ Xi, V1, V2 \ Xi, F ∪ Xi, k − |Xi|)
16:            if F_candidate ≠ no solution and
                  (F_result = no solution or |F_candidate| ≤ |F_result|) then
17:                F_result ← F_candidate
18:            end if
19:        end for
20:    end if
21:    return F_result
22: end procedure
```