Tight Lower Bounds for List Edge Coloring

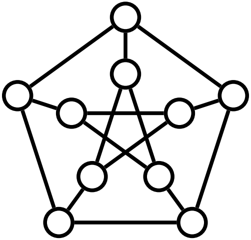
Łukasz Kowalik, Arkadiusz Socała

UNIVERSITY OF WARSAW

SWAT, Malmö, 20.06.2018

Edge Coloring

Assign colors to edges so that incident edges get distinct colors.

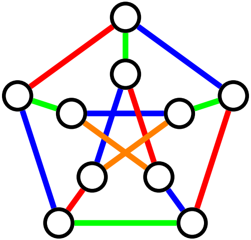


Edge Coloring (decision version)

Input: Graph $G = (V, E)$, integer k
Question: Does G admit edge coloring in k colors?

Edge Coloring

Assign colors to edges so that incident edges get distinct colors.



EDGE COLORING (decision version)

Input: Graph $G = (V, E)$, integer k

Question: Does G admit edge coloring in k colors?

Time complexity of EDGE COLORING

Denote $n = |V(G)|$, $m = |E(G)|$.

- NP-complete (Holyer 1981)

Time complexity of Edge Coloring

Denote $n = |V(G)|$, $m = |E(G)|$.

- NP-complete (Holyer 1981)
- $2^m n^{O(1)}$ time exp space (Björklund, Husfeldt, Koivisto 2006)

Time complexity of EDGE COLORING

Denote $n = |V(G)|$, $m = |E(G)|$.

- NP-complete (Holyer 1981)
- $2^m n^{O(1)}$ time exp space (Björklund, Husfeldt, Koivisto 2006)
- $2^m n^{O(1)}$ randomized time poly space (Björklund et al 2010)

Time complexity of Edge Coloring

Denote $n = |V(G)|$, $m = |E(G)|$.

- NP-complete (Holyer 1981)
- $2^m n^{O(1)}$ time exp space (Björklund, Husfeldt, Koivisto 2006)
- $2^m n^{O(1)}$ randomized time poly space (Björklund et al 2010)

Can we do better than $2^m = 2^{O(n^2)}$?

Lower bounds for Edge Coloring?

Exponential Time Hypothesis (ETH)

There is a constant $c > 0$ such that 3-SAT cannot be solved in time $O^*(2^{cn})$.

Lower bounds for EDGE COLORING?

Exponential Time Hypothesis (ETH)

There is a constant $c > 0$ such that 3-SAT cannot be solved in time $O^*(2^{cn})$.

Theorem (Folklore)

EDGE COLORING does not admit an algorithm running in time $2^{o(n)}$, unless ETH is false.

Lower bounds for Edge Coloring?

Exponential Time Hypothesis (ETH)

There is a constant $c > 0$ such that 3-SAT cannot be solved in time $O^*(2^{cn})$.

Theorem (Folklore)

Edge Coloring does not admit an algorithm running in time $2^{o(n)}$, unless ETH is false.

- The algorithm in time $2^m n^{O(1)}$ is essentially optimal for **sparse** graphs.

Lower bounds for Edge Coloring?

Exponential Time Hypothesis (ETH)

There is a constant $c > 0$ such that 3-SAT cannot be solved in time $O^*(2^{cn})$.

Theorem (Folklore)

Edge Coloring does not admit an algorithm running in time $2^{o(n)}$, unless ETH is false.

- The algorithm in time $2^m n^{O(1)}$ is essentially optimal for **sparse** graphs.
- What about **dense** graphs?

Lower bounds for EDGE COLORING?

Exponential Time Hypothesis (ETH)

There is a constant $c > 0$ such that 3-SAT cannot be solved in time $O^*(2^{cn})$.

Theorem (Folklore)

EDGE COLORING does not admit an algorithm running in time $2^{o(n)}$, unless ETH is false.

- The algorithm in time $2^m n^{O(1)}$ is essentially optimal for **sparse** graphs.
- What about **dense** graphs?
- Algorithm runs in $2^{O(n^2)}$.

Lower bounds for EDGE COLORING?

Exponential Time Hypothesis (ETH)

There is a constant $c > 0$ such that 3-SAT cannot be solved in time $O^*(2^{cn})$.

Theorem (Folklore)

EDGE COLORING does not admit an algorithm running in time $2^{o(n)}$, unless ETH is false.

- The algorithm in time $2^m n^{O(1)}$ is essentially optimal for **sparse** graphs.
- What about **dense** graphs?
- Algorithm runs in $2^{O(n^2)}$.
- **Huge gap!**

Open problem

Is there an algorithm for EDGE COLORING running in time

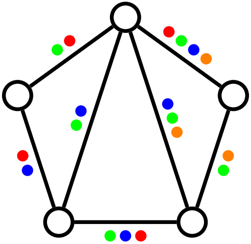
$$2^{o(n^2)} \quad ?$$

(Assuming ETH or other well-justified assumption.)

List Edge Coloring

LIST EDGE COLORING (decision version)

Input: Graph $G = (V, E)$, function $L : E \to 2^{\mathbb{N}}$

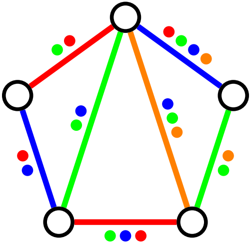


For $e \in E$, we call $L(e)$ a **list** of e.

List Edge Coloring

LIST EDGE COLORING (decision version)

Input: Graph $G = (V, E)$, function $L : E \to 2^{\mathbb{N}}$
Question: Does G admit an edge coloring c such that $c(e) \in L(e)$ for every edge $e \in E$?



For $e \in E$, we call $L(e)$ a **list** of e.

More general problem, same algorithms

Algorithms for Edge Coloring can be easily adapted to solve List Edge Coloring

- $2^m n^{O(1)}$ time exp space (Björklund, Husfeldt, Koivisto 2006)
- $2^m n^{O(1)}$ randomized time poly space (Björklund et al 2010)

(assuming $\sum_{e \in E} |L(e)| = n^{O(1)}$).

More general problem, same algorithms

Algorithms for Edge Coloring can be easily adapted to solve List Edge Coloring

- $2^m n^{O(1)}$ time exp space (Björklund, Husfeldt, Koivisto 2006)
- $2^m n^{O(1)}$ randomized time poly space (Björklund et al 2010)

(assuming $\sum_{e \in E} |L(e)| = n^{O(1)}$).

Even more general, both work for multigraphs.

More general problem, same algorithms

Algorithms for Edge Coloring can be easily adapted to solve List Edge Coloring

- $2^m n^{O(1)}$ time exp space (Björklund, Husfeldt, Koivisto 2006)
- $2^m n^{O(1)}$ randomized time poly space (Björklund et al 2010)

(assuming $\sum_{e \in E} |L(e)| = n^{O(1)}$).

Even more general, both work for multigraphs.

Can we do better? $2^{o(n^2)}$ **running time?**

Our results

Theorem 1

If there is an algorithm for LIST EDGE COLORING *for simple graphs that runs in time* $2^{o(n^2)}$*, then Exponential Time Hypothesis fails.*

Our results

Theorem 1

If there is an algorithm for LIST EDGE COLORING *for simple graphs that runs in time $2^{o(n^2)}$, then Exponential Time Hypothesis fails.*

Note

Holds even if lists are of length a most 6.

A consequence for EDGE COLORING

A simple way of verifying if a new idea for a faster algorithm for EDGE COLORING works:

A consequence for Edge Coloring

A simple way of verifying if a new idea for a faster algorithm for Edge Coloring works:

if it applies to the list version as well, there is no hope for it.

Proof for multigraphs

Theorem 2

If there is a function $f : \mathbb{N} \to \mathbb{N}$ such that List Edge Coloring *can be solved for multigraphs in time $f(n) \cdot m^{O(1)}$ for any input graph on n vertices and m edges, then $P = NP$.*

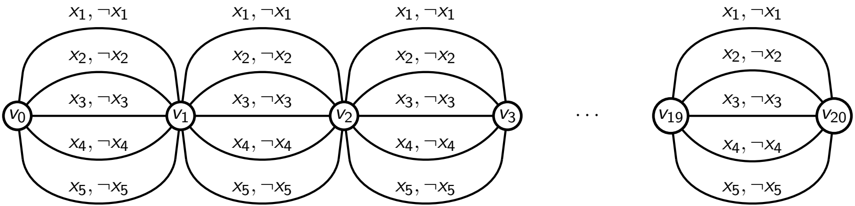
Proof plan

- $(3,4)$-SAT is 3-SAT restricted to formulas with every variable appearing in at most 4 clauses.
- $(3,4)$-SAT is NP-complete [Tovey 1984].
- Let φ be an instance of $(3,4)$-SAT.
- Reduce φ in polynomial time to an instance (G, L) of List Edge Coloring such that $|V(G)| = O(1)$.

literals as colors

Main idea

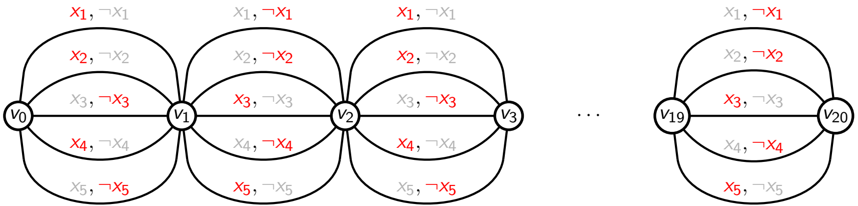
- Colors form the set $\{x_i, \neg x_i \mid i = 1, \ldots, n\}$
- In every coloring c, for every $i = 1, \ldots, n$, $c^{-1}(\{x_i, \neg x_i\})$ forms a path P_i with alternating colors.
- For multigraphs, $V = \{v_0, \ldots, v_{20}\}$ and $P_i = v_0, v_1, \ldots, v_{20}$.



literals as colors

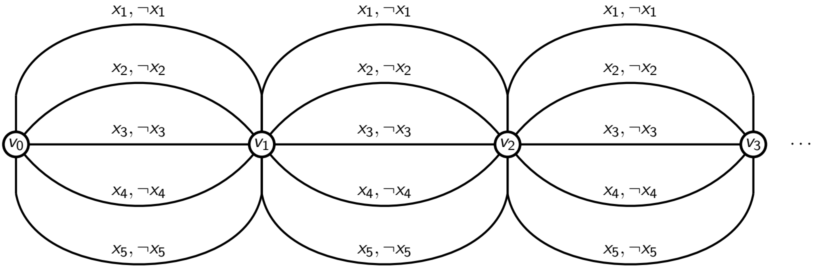
Main idea

- Colors form the set $\{x_i, \neg x_i \mid i = 1, \ldots, n\}$
- In every coloring c, for every $i = 1, \ldots, n$, $c^{-1}(\{x_i, \neg x_i\})$ forms a path P_i with alternating colors.
- For multigraphs, $V = \{v_0, \ldots, v_{20}\}$ and $P_i = v_0, v_1, \ldots, v_{20}$.



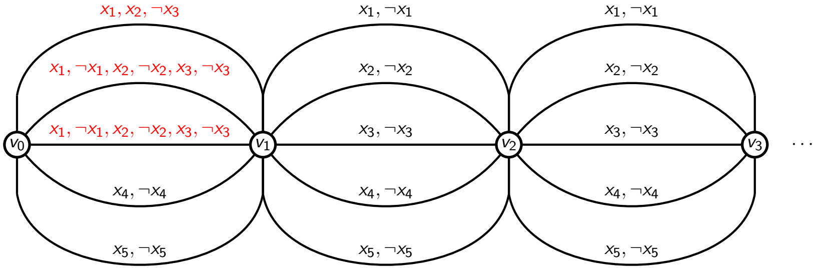
- Colors on edges $v_0 v_1$ (also $v_2 v_3, v_4 v_5, \ldots$) define a boolean assignment: $x_1 = T$, $x_2 = T$, $x_3 = F$, $x_4 = T$, $x_5 = F$.

Representing clauses



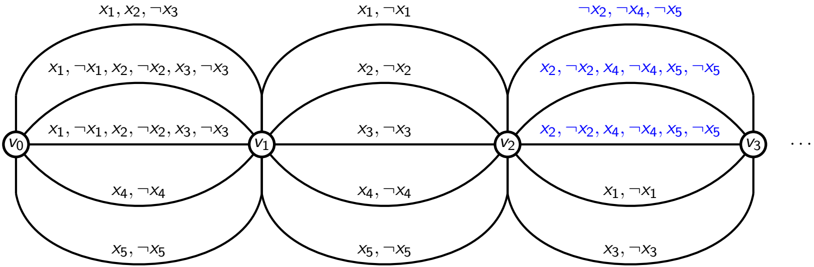
Representing clauses

- Add clause $x_1 \vee x_2 \vee \neg x_3$



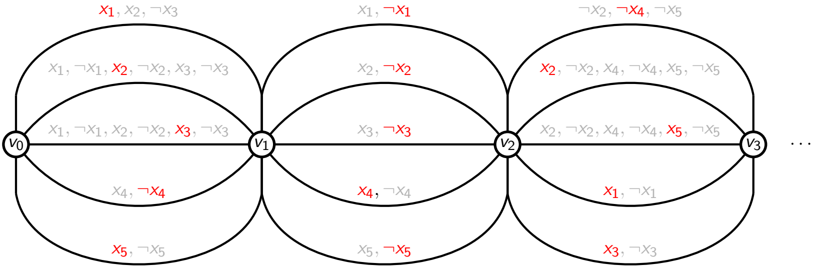
Representing clauses

- Add clause $x_1 \vee x_2 \vee \neg x_3$
- Add clause $\neg x_2 \vee \neg x_4 \vee \neg x_5$



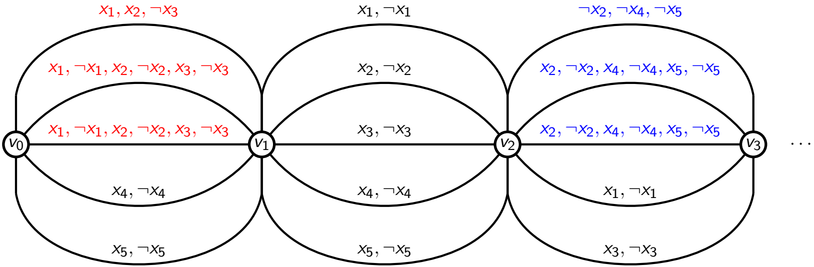
Representing clauses

- Add clause $x_1 \vee x_2 \vee \neg x_3$
- Add clause $\neg x_2 \vee \neg x_4 \vee \neg x_5$



Representing clauses

- Add clause $x_1 \vee x_2 \vee \neg x_3$
- Add clause $\neg x_2 \vee \neg x_4 \vee \neg x_5$



Observation

Two clauses must be disjoint to use edges with the same endpoints

Partitioning clauses into disjoint groups

- Recall that in φ every variable appears in at most 4 clauses.

Partitioning clauses into disjoint groups

- Recall that in φ every variable appears in at most 4 clauses.
- Build a graph $G_\varphi = (\text{Clauses}, E_\varphi)$

Partitioning clauses into disjoint groups

- Recall that in φ every variable appears in at most 4 clauses.
- Build a graph $G_\varphi = (\text{Clauses}, E_\varphi)$
- Intersecting clauses adjacent in G_φ.

Partitioning clauses into disjoint groups

- Recall that in φ every variable appears in at most 4 clauses.
- Build a graph $G_\varphi = (\text{Clauses}, E_\varphi)$
- Intersecting clauses adjacent in G_φ.
- Maximum degree in G_φ is at most $3 \cdot (4-1) = 9$

Partitioning clauses into disjoint groups

- Recall that in φ every variable appears in at most 4 clauses.
- Build a graph $G_\varphi = (\text{Clauses}, E_\varphi)$
- Intersecting clauses adjacent in G_φ.
- Maximum degree in G_φ is at most $3 \cdot (4-1) = 9$
- Color greedily vertices in G_φ in 10 colors $C_0, \ldots, C_9$.

Partitioning clauses into disjoint groups

- Recall that in φ every variable appears in at most 4 clauses.
- Build a graph $G_\varphi = (\text{Clauses}, E_\varphi)$
- Intersecting clauses adjacent in G_φ.
- Maximum degree in G_φ is at most $3 \cdot (4-1) = 9$
- Color greedily vertices in G_φ in 10 colors $C_0, \ldots, C_9$.
- For every i clauses in C_i disjoint.

Partitioning clauses into disjoint groups

- Recall that in φ every variable appears in at most 4 clauses.
- Build a graph $G_\varphi = (\text{Clauses}, E_\varphi)$
- Intersecting clauses adjacent in G_φ.
- Maximum degree in G_φ is at most $3 \cdot (4-1) = 9$
- Color greedily vertices in G_φ in 10 colors $C_0, \ldots, C_9$.
- For every i clauses in C_i disjoint.
- Every clause in C_i corresponds to three edges $x_{2i}x_{2i+1}$

Proof idea for simple graphs

- By Sparsification Lemma and the reduction to $(3,4)$-SAT it suffices to solve $(3,4)$-SAT in $2^{o(n)}$ time to refute ETH.

Proof idea for simple graphs

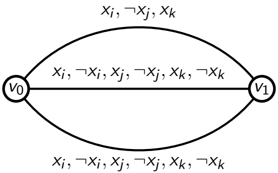
- By Sparsification Lemma and the reduction to $(3,4)$-SAT it suffices to solve $(3,4)$-SAT in $2^{o(n)}$ time to refute ETH.
- Instead of $O(1)$ vertices, use $O(1)$ layers of $\Theta(\sqrt{n})$ vertices.

Proof idea for simple graphs

- By Sparsification Lemma and the reduction to $(3,4)$-SAT it suffices to solve $(3,4)$-SAT in $2^{o(n)}$ time to refute ETH.
- Instead of $O(1)$ vertices, use $O(1)$ layers of $\Theta(\sqrt{n})$ vertices.
- Then indeed solving the output instance in time $2^{o(|V|^2)} = 2^{o(n)}$ refutes ETH.

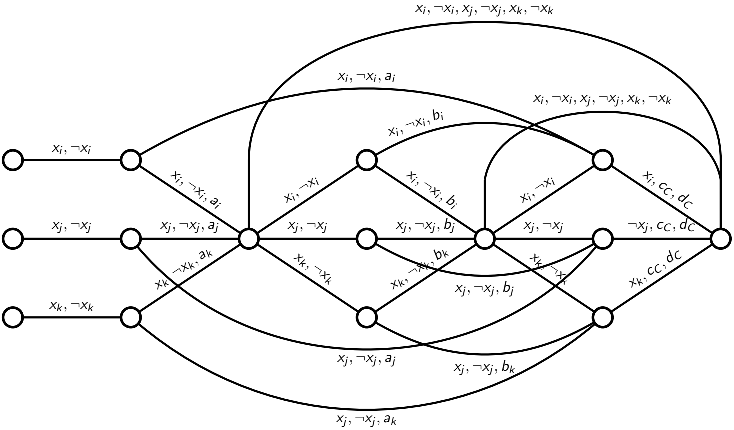
Avoiding parallel edges: a new clause gadget

Previous clause gadget (for $x_i \vee \neg x_j, x_k$):



Avoiding parallel edges: a new clause gadget

New clause gadget (for $x_i \vee \neg x_j, x_k$):



Conclusion

Conclusion

- The complexity of LIST EDGE COLORING is well understood.

Conclusion

- The complexity of List Edge Coloring is well understood.
- The complexity of Edge Coloring is not.