Checking Unique Hamiltonicity

Benedikt Klocker

Algorithms and Complexity Group
Institute of Computer Graphics and Algorithms
TU Wien

PHD Seminar, 18 Dezember 2017

Definition (UHG)

If a graph contains exactly one hamiltonian cycle it is called a *uniquely hamiltonian graph* (UHG).

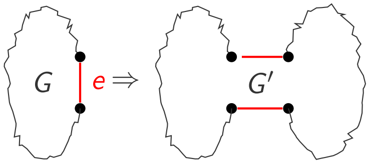
Conjecture by Bondy and Jackson

Every planar uniquely hamiltonian graph has at least two vertices of degree two.

Goal

Find a simple planar uniquely hamiltonian graph with minimum degree 3 and therefore disprove the conjecture of Bondy and Jackson.

Removing unvisited vertices



Removing unvisited vertices

b Let G be a graph. We call the pair (e, C), where C is a cycle and e an edge occuring in C a fixed edge cycle, or short FE-cycle. An FE-cycle (e, C) is called maximal if there is no other FE-cycle (e, C') with $V(C') \supsetneq V(C)$.
An FE-cycle (e, C) is called unique if there is no other FE-cycle (e, C') with $V(C') = V(C)$ and $C' \neq C$.

Simplified Goal

Find a simple planar graph with minimum degree 3 which contains a unique maximal dominating FE-cycle.

b at (0.8,0) U; [myedgestyle] (0.5,0) arc (360:0:0.5 and 0.5) node (1,0) ; at $((w) + (-1.2, 1))$ (tmp) [mynodestyle] edge (w) edge [red] $((tmp) + (-0.4, 0.4))$ edge [red] $((tmp) + (0.4, 0.4))$ edge $((tmp) + (-0.2, 0.4))$ edge $((tmp) + (0.2, 0.4))$; at $((tmp) + (0, 0.4))$ . . . ; at $((w) + (0, 1))$ (tmp) [mynodestyle] edge (w) edge [red] $((tmp) + (-0.4, 0.4))$ edge [red] $((tmp) + (0.4, 0.4))$ edge $((tmp) + (-0.2, 0.4))$ edge (tmp) [mynodestyle] edge (w) edge [red] $((tmp) + (-0.4, 0.4))$ edge [red] $((tmp) + (0.4, 0.4))$ edge $((tmp) + (-0.2, 0.4))$ edge $((tmp) + (0.2, 0.4))$; at $((tmp) + (0, 0.4))$ ...; $((w) + (-1.2, 1))$ – (w.160); $((w) + (-1.2, -1))$ – (w.200); at $((w) + (-1.2, -1))$ (tmp) [mynodestyle] edge (w) edge [red] $((tmp) + (-0.4, -0.4))$ edge [red] $((tmp) + (0.4, -0.4))$; at $((w) + (0, -1))$ (tmp) [mynodestyle] edge (w) edge [red] $((tmp) + (-0.4, -0.4))$ edge [red] $((tmp) + (0.4, -0.4))$; at $((w) + (1.2, -1))$ (tmp) [mynodestyle] edge (w) edge [red] $((tmp) + (-0.4, -0.4))$ edge [red] $((tmp) + (0.4, -0.4))$; at (5,0) (w) [mynodestyle] ; at $((w) + (0.3, 0))$ u; at $((w) + (-1.2, 1))$ (tmp) [mynodestyle] edge (w) edge [red] $((tmp) + (-0.4, 0.4))$ edge [red] $((tmp) + (0.4, 0.4))$ edge $((tmp) + (-0.2, 0.4))$ edge $((tmp) + (0.2, 0.4))$; at $((tmp) + (0, 0.4))$ ...; at $((w) + (0, 1))$ (tmp) [mynodestyle] edge (w) edge [red] $((tmp) + (-0.4, 0.4))$ edge [red] $((tmp) + (0.4, 0.4))$ edge $((tmp) + (-0.2, 0.4))$ edge $((tmp) + (0.2, 0.4))$; at $((tmp) + (0, 0.4))$ ...; at $((w) + (1.2, 1))$ (tmp) [mynodestyle] edge (w) edge [red] $((tmp) + (-0.4, 0.4))$ edge [red] $((tmp) + (0.4, 0.4))$ edge $((tmp) + (-0.2, 0.4))$ edge

$((tmp) + (0.2, 0.4))$; at $((tmp) + (0, 0.4))$ . . . ; at
$((w) + (-1.2, -1))$ (tmp) [mynodestyle] edge (w) edge [red]
$((tmp) + (-0.4, -0.4))$ edge [red] $((tmp) + (0.4, -0.4))$; at
$((w) + (0, -1))$ (tmp) [mynodestyle] edge (w) edge [red]
$((tmp) + (-0.4, -0.4))$ edge [red] $((tmp) + (0.4, -0.4))$; at
$((w) + (1.2, -1))$ (tmp) [mynodestyle] edge (w) edge [red]
$((tmp) + (-0.4, -0.4))$ edge [red] $((tmp) + (0.4, -0.4))$; at
(2.5,0) $\Rightarrow$; b at (0.8,0) U; [myedgestyle] (0.5,0) arc (360:0:0.5
and 0.5) node (1,0) ; at $((w) + (-1.2, 1))$ (tmp) [mynodestyle]
edge (w) edge [red] $((tmp) + (-0.4, 0.4))$ edge [red]
$((tmp) + (0.4, 0.4))$ edge $((tmp) + (-0.2, 0.4))$ edge
$((tmp) + (0.2, 0.4))$; at $((tmp) + (0, 0.4))$ . . . ; at $((w) + (0, 1))$
(tmp) [mynodestyle] edge (w) edge [red] $((tmp) + (-0.4, 0.4))$
edge [red] $((tmp) + (0.4, 0.4))$ edge $((tmp) + (-0.2, 0.4))$ edge
$((tmp) + (0.2, 0.4))$; at $((tmp) + (0, 0.4))$ . . . ; at $((w) + (1.2, 1))$
(tmp) [mynodestyle] edge (w) edge [red] $((tmp) + (-0.4, 0.4))$
edge [red] $((tmp) + (0.4, 0.4))$ edge $((tmp) + (-0.2, 0.4))$ edge
$((tmp) + (0.2, 0.4))$; at $((tmp) + (0, 0.4))$ . . . ; $((w) + (-1.2, 1))$
– (w.160); $((w) + (-1.2, -1))$ – (w.200); at $((w) + (-1.2, -1))$

(tmp) [mynodestyle] edge (w) edge [red] $((tmp)+(-0.4,-0.4))$ edge [red] $((tmp)+(0.4,-0.4))$; at $((w)+(0,-1))$ (tmp) [mynodestyle] edge (w) edge [red] $((tmp)+(-0.4,-0.4))$ edge [red] $((tmp)+(0.4,-0.4))$; at $((w)+(1.2,-1))$ (tmp) [mynodestyle] edge (w) edge [red] $((tmp)+(-0.4,-0.4))$ edge [red] $((tmp)+(0.4,-0.4))$; at (5,0) (w) [mynodestyle] ; at $((w)+(0.3,0))$ u; at $((w)+(-1.2,1))$ (tmp) [mynodestyle] edge (w) edge [red] $((tmp)+(-0.4,0.4))$ edge [red] $((tmp)+(0.4,0.4))$ edge $((tmp)+(-0.2,0.4))$ edge $((tmp)+(0.2,0.4))$; at $((tmp)+(0,0.4))$ $\ldots$; at $((w)+(0,1))$ (tmp) [mynodestyle] edge (w) edge [red] $((tmp)+(-0.4,0.4))$ edge [red] $((tmp)+(0.4,0.4))$ edge $((tmp)+(-0.2,0.4))$ edge $((tmp)+(0.2,0.4))$; at $((tmp)+(0,0.4))$ $\ldots$; at $((w)+(1.2,1))$ (tmp) [mynodestyle] edge (w) edge [red] $((tmp)+(-0.4,0.4))$ edge [red] $((tmp)+(0.4,0.4))$ edge $((tmp)+(-0.2,0.4))$ edge $((tmp)+(0.2,0.4))$; at $((tmp)+(0,0.4))$ $\ldots$; at $((w)+(-1.2,-1))$ (tmp) [mynodestyle] edge (w) edge [red] $((tmp)+(-0.4,-0.4))$ edge [red] $((tmp)+(0.4,-0.4))$; at $((w)+(0,-1))$ (tmp) [mynodestyle] edge (w) edge [red]

((tmp) + (−0.4, −0.4)) edge [red] ((tmp) + (0.4, −0.4)); at
((w) + (1.2, −1)) (tmp) [mynodestyle] edge (w) edge [red]
((tmp) + (−0.4, −0.4)) edge [red] ((tmp) + (0.4, −0.4)); at
(2.5,0) ⇒;

Observation: If G is 2-vertex-connected, the new vertex u has degree at least 2 and in this case we can still apply the previous transformation.

First Approach

New Goal

Find a simple planar graph with minimum degree 3 which contains a unique maximal ~~dominating~~ FE-cycle.

benumerate

Generate (all) planar graphs (with a fixed number of vertices) with minimum degree 3.

For each generated graph G, check if it contains a unique maximal FE-cycle by doing the following:

1. For each edge e in G repeat the following steps until no new maximal FE-cycle with e as the fixed edge could be found:
   1.1 Find a new maximal FE-cycle with e as the fixed edge.

1.2 Check if the FE-cycle is unique. benumerate

Generate (all) planar graphs (with a fixed number of vertices) with

For each generated graph G, check if it contains a unique maximal FE-cycle by doing the following:

1. For each edge e in G repeat the following steps until no new maximal FE-cycle with e as the fixed edge could be found:

1.1 Find a new maximal FE-cycle with e as the fixed edge.

benumerate

2. Generate (all) planar graphs (with a fixed number of vertices) with minimum degree 3.
3. For each generated graph G, check if it contains a unique maximal FE-cycle by doing the following:
   3.1 For each edge e in G repeat the following steps until no new maximal FE-cycle with e as the fixed edge could be found:

3.1.1 Find a new maximal FE-cycle with e as the fixed edge.

3.1.2 Check if the FE-cycle is unique.

Generate (all) planar graphs (with a fixed number of vertices) with minimum degree 3.

For each generated graph G, check if it contains a unique maximal FE-cycle by doing the following:

1. For each edge e in G repeat the following steps until no new maximal FE-cycle with e as the fixed edge could be found:

1.1 Find a new maximal FE-cycle with e as the fixed edge.

1.2 Check if the FE-cycle is unique.

ILP model for Finding a Maximal FE-cycle

Input

A graph $G = (V, E)$, an edge $e_0 = i_0 j_0 \in E$ and a set $\mathcal{C}$ of all maximal FE-cycles with e as the fixed edge found until now.

Variables

- $(x_v)_{v \in V} \ldots x_v = 1$ iff $v \in V$ is used in the cycle
- $(y_e)_{e \in E} \ldots y_e = 1$ iff $e \in E$ is used in the cycle

ILP model for Finding a Maximal FE-cycle cont.

Objective:

$$\max \sum_{i \in V} x_i$$

Constraints:

$$\sum_{j \in N(i)} y_{ij} = 2x_i \quad \forall i \in V \tag{1}$$

$$y_{i_0 j_0} = 1 \tag{2}$$

$$\sum_{i \in V \setminus C} x_i \geq 1 \quad \forall C \in \mathcal{C} \tag{3}$$

$$\sum_{e \in \delta(V')} y_e \geq 2x_i \quad \forall \emptyset \neq V' \subseteq V \setminus \{i_0\}, i \in V' \tag{4}$$

$$y_e \in \{0,1\} \quad \forall e \in E \tag{5}$$

$$x_i \in \{0,1\} \quad \forall i \in V \tag{6}$$

ILP model for Checking Uniqueness of FE-cycle

Input

A graph $G = (V, E)$ and a maximal FE-cycle (e, C).

Variables

- $(y_e)_{e \in E_C}$ ... $y_e = 1$ iff $e \in E$ is used in the cycle

ILP model for Checking Uniqueness of FE-cycle cont.

No objective (only feasibility interesting)

Constraints:

$$\sum_{j \in N(i)_C} y_{ij} = 2 \quad \forall i \in V_C \tag{7}$$

$$y_{i_1 i_2} = 1 \tag{8}$$

$$\sum_{e \in \delta(V')} y_e \geq 2 \quad \forall \emptyset \neq V' \subseteq V_C \setminus \{i_1\}, k \in V' \tag{9}$$

$$\sum_{ij \in E_C \setminus E(C)} y_{ij} \geq 2 \tag{10}$$

$$y_e \in \{0, 1\} \quad \forall e \in E_C \tag{11}$$

Reducing the number of ILPs to solve

- Store for each set of vertices V' and for each edge e a list of all cycles found until now using the edge e and the vertices V'.
- In the first phase only search cycles for new vertex sets or new edges.
- In the second phase we do not have to check FE-cycles for which there are already two cycles containing the fixed edge.

Reusing ILP-states

- Goal: Reuse ILP-tree after maximal cycle got found.
- Use callback to store every found cycle in $\mathcal{C}$ and add the constraint (3) for every cycle.
  - The found cycles don't have to be maximal!
  - The constraint (3) ensures that afterwards only larger cycles or not comparable cycles get found
  - If a larger cycle gets found remove all smaller cycles from the datastructure $\mathcal{C}$ and the according constraints from the model, since they get dominated from the new constraint.
- If no new cycle got found, all cycles in the datastructure are maximal and no other maximal cycle exists
- The ILP terminates as infeasible, since all work happens in the collection of the cycles during the callback.

Goal

Find properties for a minimal planar graph with minimum degree 3, which contains a unique maximal FE-cycle. Reduce the number of graphs to test drastically by only testing candidates for a minimal counter example.

By minimal we mean minimal according to the following relation.

Definition

Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs. Then we say $G_1 \leq G_2$ iff

$$|V_1| < |V_2| \vee (|V_1| = |V_2| \wedge |E_1| \leq |E_2|) .$$

Properties of a Minimal Counter Example

Definition

A vertex with degree 3 or less is called a *small* vertex and otherwise a *large* vertex.

Let $G = (V, E)$ be a minimal counter example with the unique FE-cycle (e, C):

- C is dominating
- G is 3-connected
- Every neighbor of a large vertex is in $V(C)$

Properties of a Minimal Counter Example

Definition

A vertex with degree 3 or less is called a *small* vertex and otherwise a *large* vertex.

Let $G = (V, E)$ be a minimal counter example with the unique FE-cycle (e, C):

- C is dominating
- G is 3-connected
- Every neighbor of a large vertex is in $V(C)$

Properties of a Minimal Counter Example

Definition

A vertex with degree 3 or less is called a *small* vertex and otherwise a *large* vertex.

Let $G = (V, E)$ be a minimal counter example with the unique FE-cycle (e, C):

- C is dominating
- G is 3-connected
- Every neighbor of a large vertex is in $V(C)$

Properties of a Minimal Counter Example

Definition

A vertex with degree 3 or less is called a *small* vertex and otherwise a *large* vertex.

Let $G = (V, E)$ be a minimal counter example with the unique FE-cycle (e, C):

- C is dominating
- G is 3-connected
- Every neighbor of a large vertex is in $V(C)$

Properties of a Minimal Counter Example

Definition

A vertex with degree 3 or less is called a *small* vertex and otherwise a *large* vertex.

Let $G = (V, E)$ be a minimal counter example with the unique FE-cycle (e, C):

- C is dominating
- G is 3-connected
- Every neighbor of a large vertex is in $V(C)$

Properties of a Minimal Counter Example cont.

Let $G = (V, E)$ be a minimal counter example with the unique FE-cycle (e, C):

- Every arc between large vertices is in $E(C)$
- No vertex has 3 large neighbors
- There is no cycle consisting only of large vertices in G
- $|E| \leq |V| + n_3 - \delta$ where $n_3 = |\{v \in V : \deg(v) = 3\}|$ and δ is the number of small vertices incident to e
- ~~$|E| \leq \frac{5}{3}|V|$~~
- G does not contain any triangles
- $|E| \leq 2|V| - 4$

Properties of a Minimal Counter Example cont.

Let $G = (V, E)$ be a minimal counter example with the unique FE-cycle (e, C):

- Every arc between large vertices is in $E(C)$
- No vertex has 3 large neighbors
- There is no cycle consisting only of large vertices in G
- $|E| \leq |V| + n_3 - \delta$ where $n_3 = |\{v \in V : \deg(v) = 3\}|$ and δ is the number of small vertices incident to e
- $|E| \leq \frac{5}{3}|V|$
- G does not contain any triangles
- $|E| \leq 2|V| - 4$

Properties of a Minimal Counter Example cont.

Let $G = (V, E)$ be a minimal counter example with the unique FE-cycle (e, C):

- Every arc between large vertices is in $E(C)$
- No vertex has 3 large neighbors
- There is no cycle consisting only of large vertices in G
- $|E| \leq |V| + n_3 - \delta$ where $n_3 = |\{v \in V : \deg(v) = 3\}|$ and δ is the number of small vertices incident to e
- ~~$|E| \leq \frac{5}{3}|V|$~~
- G does not contain any triangles
- $|E| \leq 2|V| - 4$

Properties of a Minimal Counter Example cont.

Let $G = (V, E)$ be a minimal counter example with the unique FE-cycle (e, C):

- Every arc between large vertices is in $E(C)$
- No vertex has 3 large neighbors
- There is no cycle consisting only of large vertices in G
- $|E| \leq |V| + n_3 - \delta$ where $n_3 = |\{v \in V : \deg(v) = 3\}|$ and δ is the number of small vertices incident to e
- ~~$|E| \leq \frac{5}{3}|V|$~~
- G does not contain any triangles
- $|E| \leq 2|V| - 4$

Construction of Candidate Graphs

- We use plantri to construct planar graphs
- Plantri constructs only one graph per isomporphism-class
- We can fix a number of vertices and give an upper bound for the number of edges ($|E| \leq 2n - 4$). Then we can filter the results by the other properties of a minimal counter example.
- Disadvantages:
  - The upper bound for the edges is only a filter and therefore not efficient
  - All filters together filter out most of the generated graphs, only a small part is really interesting (especially the property that the graph has no triangles)

Construction of Candidate Graphs through Dual Graphs

New idea: Generate dual graphs with plantri

- Use the edge upper bound to get an upper bound for the faces:

$$|F| = |E| - |V| + 2 \leq 2|V| - 4 - |V| + 2 \leq |V| - 2$$

- The dual graph of a 3-connected graph is also 3-connected
- The dual graph has minimum degree 4 since the original graph contained no triangles

To get all relevant graphs with at most n vertices we construct all dual graphs with the above properties with at most $n - 2$ vertices and build the dual graphs of them.