Reductions & *NP*-completeness

Alexandros Angelopoulos

M.P.L.A.

February 7, 2014

Outline

0/1 Integer Programming

3-colorability

Hamilton Path (HP)

Traveling Salesman Problem (TSP)

Reducing 3-SAT to 0/1 IP

Definition 1.1 (0/1 IP).

Input: an integer matrix C and vector b.
Output: decide if there is a 0/1 vector x such that: $Cx \geq b$.

- 0/1 IP $\in$ *NP*(why?)
- We choose 3-SAT as our known *NP*-complete problem and consider the formula:

$$\phi = C_1 \wedge C_2 \wedge \ldots \wedge C_m$$

with literals $x_1, \ldots x_n$

We will construct our $m \times n$ matrix C : $c_{ij} = \begin{cases} 1, & \text{if } x_j \in C_i \\ -1, & \text{if } \bar{x_j} \in C_i \\ 0, & \text{otherwise} \end{cases}$ and

$b_i = 1-$(the number of $\overline{complemented}$ variables in C_i)

Reducing 3-SAT to 0/1 IP

Definition 1.1 (0/1 IP).

Input: an integer matrix C and vector b.
Output: decide if there is a 0/1 vector x such that: $Cx \geq b$.

- 0/1 IP $\in$ *NP*(why?)
- We choose 3-SAT as our known *NP*-complete problem and consider the formula:

$$\phi = C_1 \wedge C_2 \wedge \ldots \wedge C_m$$

with literals $x_1, \ldots x_n$

We will construct our $m \times n$ matrix C : $c_{ij} = \begin{cases} 1, & \text{if } x_j \in C_i \\ -1, & \text{if } \bar{x_j} \in C_i \\ 0, & \text{otherwise} \end{cases}$ and

$b_i = 1-$(the number of $\overline{complemented}$ variables in C_i)

Reducing 3-SAT to 0/1 IP

Definition 1.1 (0/1 IP).

Input: an integer matrix C and vector b.
Output: decide if there is a 0/1 vector x such that: $Cx \geq b$.

- 0/1 IP $\in$ *NP*(why?)
- We choose 3-SAT as our known *NP*-complete problem and consider the formula:

$$\phi = C_1 \wedge C_2 \wedge \ldots \wedge C_m$$

with literals $x_1, \ldots x_n$

We will construct our $m \times n$ matrix C : $c_{ij} = \begin{cases} 1, & \text{if } x_j \in C_i \\ -1, & \text{if } \bar{x_j} \in C_i \\ 0, & \text{otherwise} \end{cases}$ and

$b_i = 1-$(the number of $\overline{complemented}$ variables in C_i)

Reducing 3-SAT to 0/1 IP

Definition 1.1 (0/1 IP).

Input: an integer matrix C and vector b.
Output: decide if there is a 0/1 vector x such that: $Cx \geq b$.

- 0/1 IP $\in$ *NP*(why?)
- We choose 3-SAT as our known *NP*-complete problem and consider the formula:

$$\phi = C_1 \wedge C_2 \wedge \ldots \wedge C_m$$

with literals $x_1, \ldots x_n$

We will construct our $m \times n$ matrix C : $c_{ij} = \begin{cases} 1, & \text{if } x_j \in C_i \\ -1, & \text{if } \bar{x_j} \in C_i \\ 0, & \text{otherwise} \end{cases}$ and

$b_i = 1-$(the number of $\overline{complemented}$ variables in C_i)

Reducing 3-SAT to 0/1 IP

Note that: $Cx \geq b$ actually means $\sum_{j=1}^{n} c_{ij} x_j \geq b_i$, $\forall i$.

Reducing 3-SAT to 0/1 IP

Note that: $Cx \geq b$ actually means $\sum_{j=1}^{n} c_{ij}x_j \geq b_i$, $\forall i$.

- If 3-SAT is satisfiable, then every C_i is True. Focus on a line of C and discard the zeros:

Reducing 3-SAT to 0/1 IP

Note that: $Cx \geq b$ actually means $\sum_{j=1}^{n} c_{ij} x_j \geq b_i$, $\forall i$.

- If 3-SAT is satisfiable, then every C_i is True. Focus on a line of C and discard the zeros:
- $c_{ij_1} x_{j_1} + c_{ij_2} x_{j2} + c_{ij_3} x_{j3} \geq 1 - \#(\overline{complemented}) \Rightarrow$
$$\begin{cases} \mathbf{1} x_{j_1} + \mathbf{1} x_{j_2} + \mathbf{1} x_{j_3} \geq 1 \\ \mathbf{1} x_{j_1} + \mathbf{1} x_{j_2} - \mathbf{1} x_{j_3} \geq 0 \\ \mathbf{1} x_{j_1} - \mathbf{1} x_{j_2} - \mathbf{1} x_{j_3} \geq -1 \\ -\mathbf{1} x_{j_1} - \mathbf{1} x_{j_2} - \mathbf{1} x_{j_3} \geq -2 \end{cases}$$

Reducing 3-SAT to 0/1 IP

Note that: $Cx \geq b$ actually means $\sum_{j=1}^{n} c_{ij} x_j \geq b_i$, $\forall i$.

- If 3-SAT is satisfiable, then every C_i is True. Focus on a line of C and discard the zeros:
- $c_{ij_1} x_{j_1} + c_{ij_2} x_{j2} + c_{ij_3} x_{j3} \geq 1 - \#(\overline{complemented}) \Rightarrow$
$$\begin{cases} \mathbf{1} x_{j_1} + \mathbf{1} x_{j_2} + \mathbf{1} x_{j_3} \geq 1 \\ \mathbf{1} x_{j_1} + \mathbf{1} x_{j_2} \mathbf{-1} x_{j_3} \geq 0 \\ \mathbf{1} x_{j_1} \mathbf{-1} x_{j_2} \mathbf{-1} x_{j_3} \geq -1 \\ \mathbf{-1} x_{j_1} \mathbf{-1} x_{j_2} \mathbf{-1} x_{j_3} \geq -2 \end{cases}$$
✔

Reducing 3-SAT to 0/1 IP

Note that: $Cx \geq b$ actually means $\sum_{j=1}^{n} c_{ij}x_j \geq b_i$, $\forall i$.

- If 3-SAT is satisfiable, then every C_i is True. Focus on a line of C and discard the zeros:
- $c_{ij_1}x_{j_1} + c_{ij_2}x_{j2} + c_{ij_3}x_{j3} \geq 1 - \#(\overline{complemented}) \Leftrightarrow$

$$\begin{cases} \mathbf{1}x_{j_1} + \mathbf{1}x_{j_2} + \mathbf{1}x_{j_3} \geq 1 \\ \mathbf{1}x_{j_1} + \mathbf{1}x_{j_2} \mathbf{-1}x_{j_3} \geq 0 \\ \mathbf{1}x_{j_1} \mathbf{-1}x_{j_2} \mathbf{-1}x_{j_3} \geq -1 \\ \mathbf{-1}x_{j_1} \mathbf{-1}x_{j_2} \mathbf{-1}x_{j_3} \geq -2 \end{cases}$$

✔

Note that: $Cx \geq b$ actually means $\sum_{j=1}^{n} c_{ij}x_j \geq b_i,\ \forall i$.

- If 3-SAT is satisfiable, then every C_i is True. Focus on a line of C and discard the zeros:
- $c_{ij_1}x_{j_1} + c_{ij_2}x_{j2} + c_{ij_3}x_{j3} \geq 1 - \#(\overline{complemented}) \Leftrightarrow$

$$\begin{cases} \mathbf{1}x_{j_1} + \mathbf{1}x_{j_2} + \mathbf{1}x_{j_3} \geq 1 \\ \mathbf{1}x_{j_1} + \mathbf{1}x_{j_2} - \mathbf{1}x_{j_3} \geq 0 \\ \mathbf{1}x_{j_1} - \mathbf{1}x_{j_2} - \mathbf{1}x_{j_3} \geq -1 \\ -\mathbf{1}x_{j_1} - \mathbf{1}x_{j_2} - \mathbf{1}x_{j_3} \geq -2 \end{cases}$$

3-SAT $\leq^{P}$ 0/1 IP

Outline

0/1 Integer Programming

3-colorability

Hamilton Path (HP)

Traveling Salesman Problem (TSP)

Reducing 3-SAT to 3-COLOR

Definition 2.1 (3-COLOR).

Input: a graph $G(V, E)$.
Output: decide if $\chi(G) \leq 3$?

- 3-COLOR $\in$ *NP*(why?)
- We choose 3-SAT as our known *NP*-complete problem and consider (again) the formula:

$$\phi = C_1 \wedge C_2 \wedge ... \wedge C_m$$

with literals $x_1, ...x_n$

Reducing 3-SAT to 3-COLOR

Definition 2.1 (3-COLOR).

Input: a graph $G(V, E)$.
Output: decide if $\chi(G) \leq 3$?

- 3-COLOR $\in$ *NP*(why?)
- We choose 3-SAT as our known *NP*-complete problem and consider (again) the formula:

$$\phi = C_1 \wedge C_2 \wedge \ldots \wedge C_m$$

with literals $x_1, \ldots x_n$

Reducing 3-SAT to 3-COLOR

Definition 2.1 (3-COLOR).

Input: a graph $G(V, E)$*.*
Output: decide if $\chi(G) \leq 3$*?*

- 3-COLOR $\in$ *NP*(why?)
- We choose 3-SAT as our known *NP*-complete problem and consider (again) the formula:

$$\phi = C_1 \wedge C_2 \wedge ... \wedge C_m$$

with literals $x_1, ...x_n$

Constructing the graph G_ϕ

- We'll consider the forumla $\phi = (x \vee y \vee \neg z) \wedge (\neg y \vee z \vee \neg w)$.

Constructing the graph G_ϕ

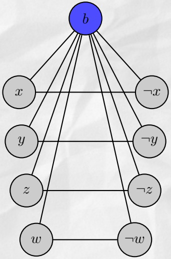
- We'll consider the forumla $\phi = (x \vee y \vee \neg z) \wedge (\neg y \vee z \vee \neg w)$.
- Let's start with the vertices of the literals: for each x_i we create v_i and $\bar{v_i}$.

Constructing the graph G_ϕ

- We'll consider the forumla $\phi = (x \vee y \vee \neg z) \wedge (\neg y \vee z \vee \neg w)$.
- Let's start with the vertices of the literals: for each x_i we create v_i and $\bar{v_i}$.
- In order to dictate an equivalent True/False coloring of $v_i, \bar{v_i}$, we draw all edges $v_i\bar{v_i}$ **plus** we link all $v_i, \bar{v_i}$ with a "base" vertex b. Check that now we have n triangles, all having b in common.

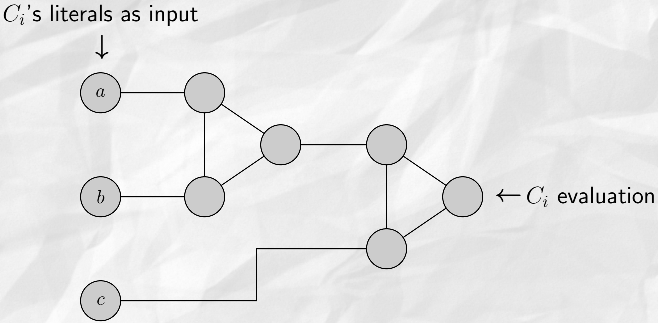
Constructing the graph G_ϕ

- We'll consider the forumla $\phi = (x \vee y \vee \neg z) \wedge (\neg y \vee z \vee \neg w)$.
- Let's start with the vertices of the literals: for each x_i we create v_i and $\bar{v_i}$.
- In order to dictate an equivalent True/False coloring of $v_i, \bar{v_i}$, we draw all edges $v_i\bar{v_i}$ **plus** we link all $v_i, \bar{v_i}$ with a "base" vertex b. Check that now we have n triangles, all having b in common.



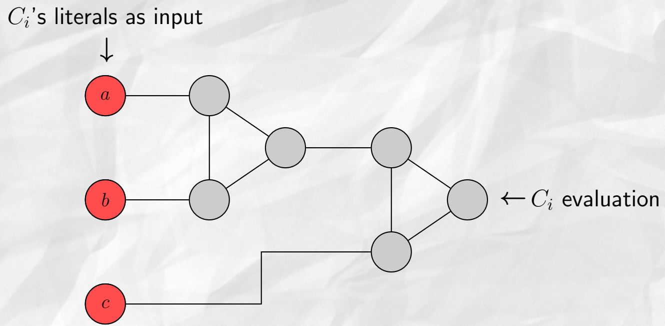
Constructing the graph G_ϕ

The gadget: a color-driven "or" gate



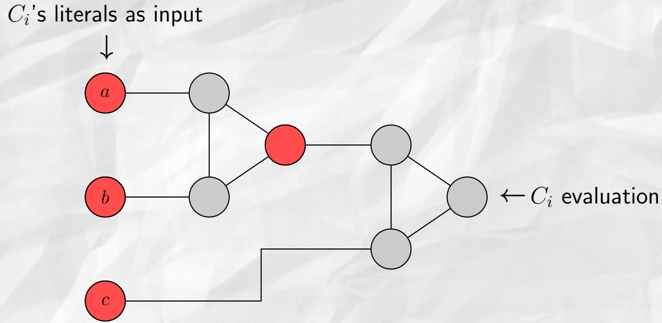
Constructing the graph G_ϕ

The gadget: a color-driven "or" gate



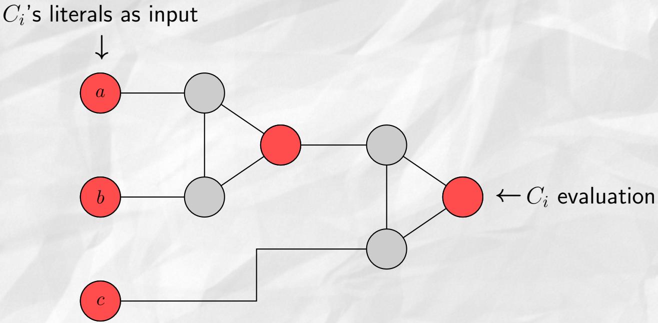
Constructing the graph G_ϕ

The gadget: a color-driven "or" gate



Constructing the graph G_ϕ

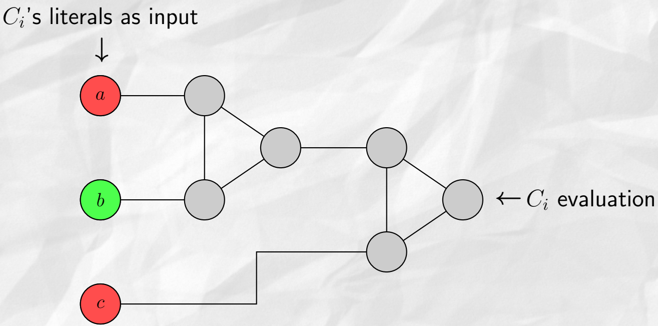
The gadget: a color-driven "or" gate



- If all a,b,c are colored "False", the output vertex **has** to be **False**.

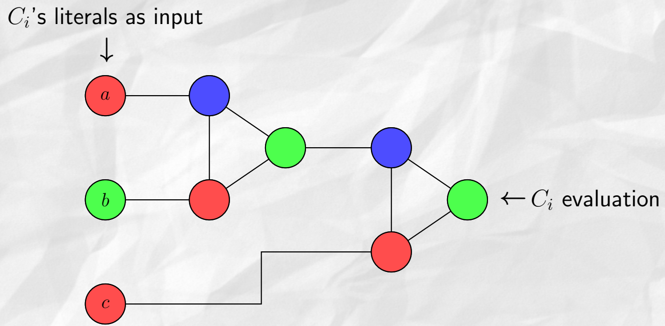
Constructing the graph G_ϕ

The gadget: a color-driven "or" gate



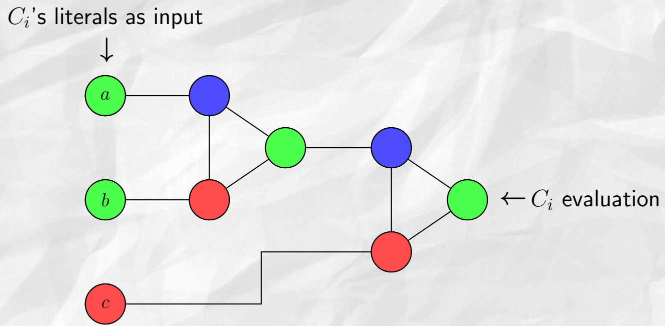
- If all a,b,c are colored "False", the output vertex **has** to be **False**.

The gadget: a color-driven "or" gate



- If all a,b,c are colored "False", the output vertex **has** to be **False**.
- If a or b or c is "True", then the output vertex **can** also be **True**.

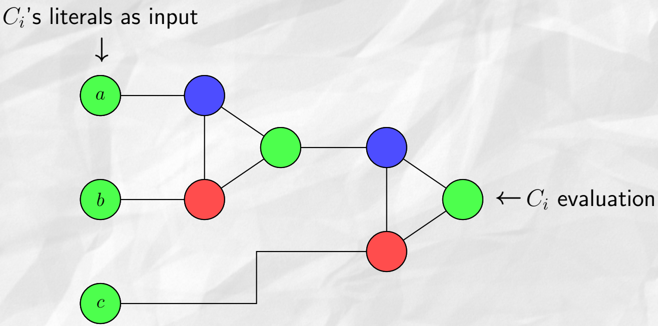
The gadget: a color-driven "or" gate



- If all a,b,c are colored "False", the output vertex **has** to be **False**.
- If a or b or c is "True", then the output vertex **can** also be **True**.

Constructing the graph G_ϕ

The gadget: a color-driven "or" gate



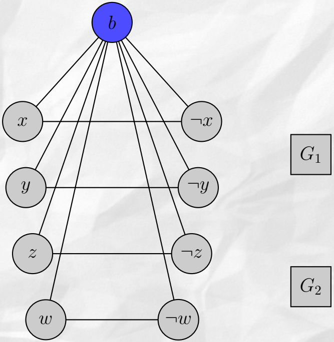
- If all a,b,c are colored "False", the output vertex **has** to be **False**.
- If a or b or c is "True", then the output vertex **can** also be **True**.

Completing G_ϕ

- $\phi = (x \vee y \vee \neg z) \wedge (\neg y \vee z \vee \neg w)$

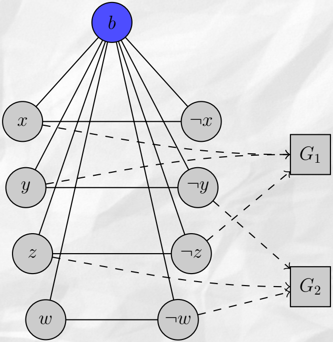
Completing G_ϕ

- $\phi = (x \vee y \vee \neg z) \wedge (\neg y \vee z \vee \neg w)$



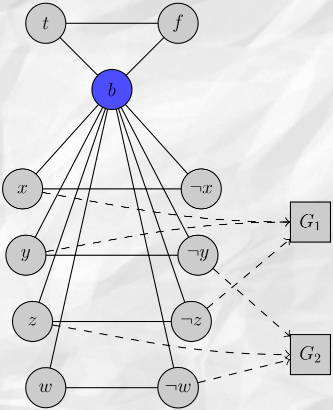
Completing G_ϕ

- $\phi = (x \vee y \vee \neg z) \wedge (\neg y \vee z \vee \neg w)$



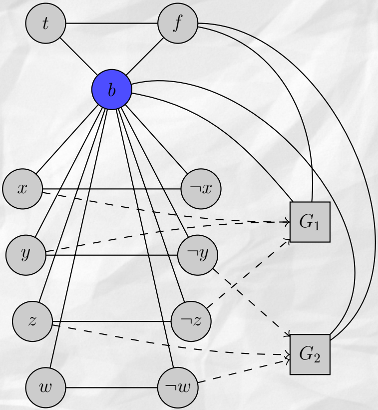
Completing G_ϕ

- $\phi = (x \vee y \vee \neg z) \wedge (\neg y \vee z \vee \neg w)$



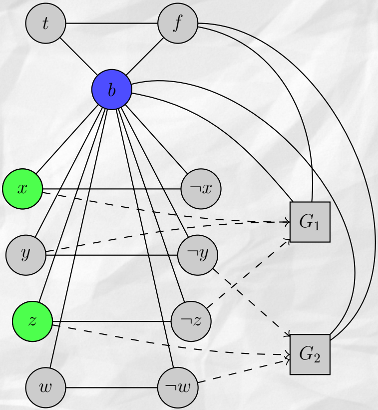
Completing G_ϕ

- $\phi = (x \vee y \vee \neg z) \wedge (\neg y \vee z \vee \neg w)$



Completing G_ϕ

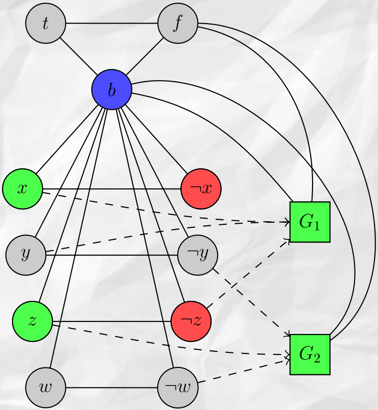
- $\phi = (x \vee y \vee \neg z) \wedge (\neg y \vee z \vee \neg w)$



- Let's satisfy ϕ...

Completing G_ϕ

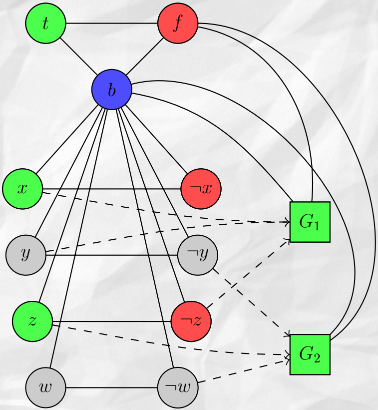
- $\phi = (x \vee y \vee \neg z) \wedge (\neg y \vee z \vee \neg w)$



- Let's satisfy ϕ...

Completing G_ϕ

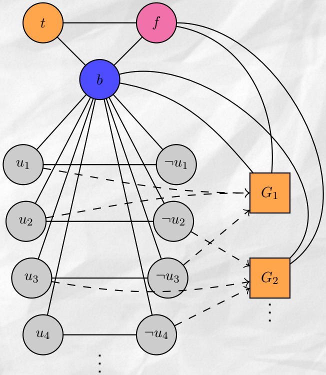
- $\phi = (x \vee y \vee \neg z) \wedge (\neg y \vee z \vee \neg w)$



- Let's satisfy ϕ... $\Rightarrow \boldsymbol{\chi(G_\phi) \leq 3}$

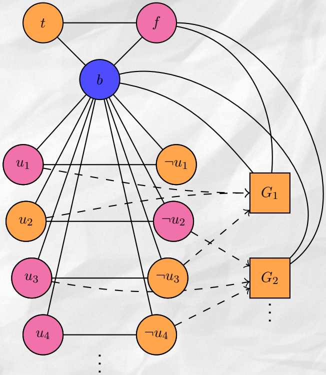
Checking the "if and only if"

- Now let G_ϕ be 3-colorable.



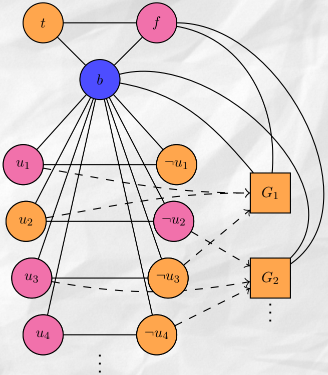
Checking the “if and only if”

- Now let G_ϕ be 3-colorable.
- And pay attention to the coloring of $u_i, \bar{u_i}$



Checking the "if and only if"

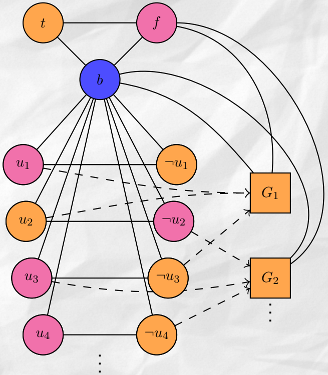
- Now let G_ϕ be 3-colorable.
- And pay attention to the coloring of $u_i, \bar{u_i}$



- Since the gadgets output orange, they must each have an orange input.

Checking the "if and only if"

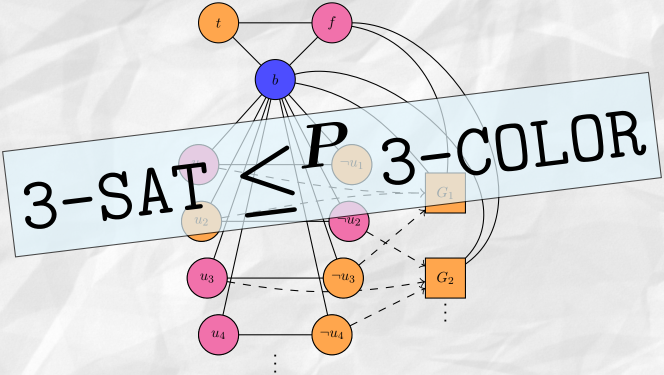
- Now let G_ϕ be 3-colorable.
- And pay attention to the coloring of $u_i, \bar{u_i}$



- Since the gadgets output orange, they must each have an orange input.
- So our true color is the orange, and an assignment that satisfies ϕ follows the orange u-nodes.

Checking the "if and only if"

- Now let G_ϕ be 3-colorable.
- And pay attention to the coloring of $u_i, \bar{u_i}$



- Since the gadgets output orange, they must each have an orange input.
- So our true color is the orange, and an assignment that satisfies ϕ follows the orange u-nodes.

Outline

0/1 Integer Programming

3-colorability

Hamilton Path (HP)

Traveling Salesman Problem (TSP)

Reducing 3-SAT to Hamilton Path

Definition 3.1 (Hamilton Path).

Input: graph G.
Output: decide whether G allows a path visiting all nodes excatly once.

- Hamilton Path $\in$ *NP*. We can guess $n-1$ edges and verify if they add up to a Hamilton Path.
- We need 3 gadgets for this problem..

Reducing 3-SAT to Hamilton Path

Definition 3.1 (Hamilton Path).

Input: graph G.
Output: decide whether G allows a path visiting all nodes excatly once.

- Hamilton Path $\in$ *NP*. We can guess $n-1$ edges and verify if they add up to a Hamilton Path.
- We need 3 gadgets for this problem..

Reducing `3-SAT` to `Hamilton Path`

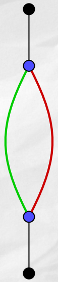
Definition 3.1 (`Hamilton Path`).

Input: graph G.
Output: decide whether G allows a path visiting all nodes excatly once.

- `Hamilton Path` $\in$ *NP*. We can guess $n-1$ edges and verify if they add up to a Hamilton Path.
- We need 3 gadgets for this problem..

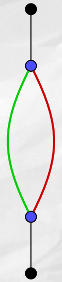
Gadgets (1/3)

The choice gadget - one per literal



Gadgets (1/3)

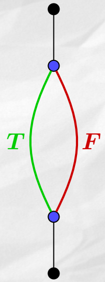
The choice gadget - one per literal



- Actually, the colored edges will become subgraphs that allow a path between the blue nodes.

Gadgets (1/3)

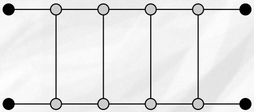
The choice gadget - one per literal



- Actually, the colored edges will become subgraphs that allow a path between the blue nodes.
- They sure translate to an evaluation "True" of "False" for the literal.

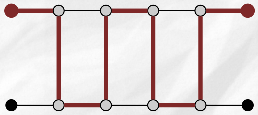
Gadgets (2/3)

The consistency gadget - an "xor" gate



Gadgets (2/3)

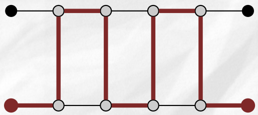
The consistency gadget - an "xor" gate



- A part o a Hamilton Path must either enter and exit this subgraph using **both top vertices**

Gadgets (2/3)

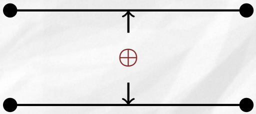
The consistency gadget - an "xor" gate



- A part o a Hamilton Path must either enter and exit this subgraph using **both top vertices or both bottom vertices.**

Gadgets (2/3)

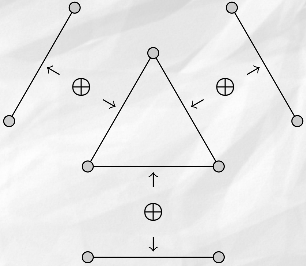
The consistency gadget - an “xor” gate



- A part o a Hamilton Path must either enter and exit this subgraph using **both top vertices or both bottom vertices.**
- That “exclusive or” functionality will be the hint for **gadget 3** to prove useful.

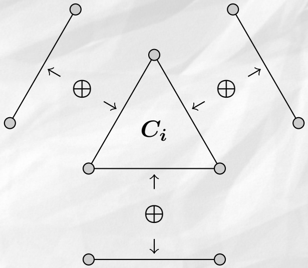
Gadgets (3/3)

The constraint gadget - one per clause



Gadgets (3/3)

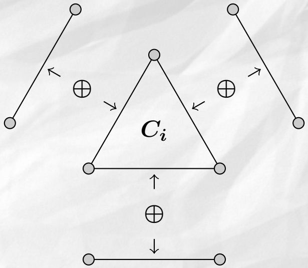
The constraint gadget - one per clause



- Let's take $C_i = (x_1 \vee x_2 \vee \neg x_3)$

Gadgets (3/3)

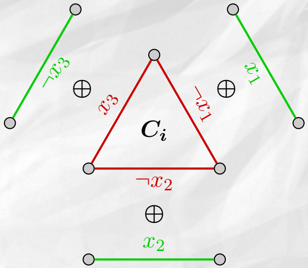
The constraint gadget - one per clause



- Let's take $C_i = (x_1 \vee x_2 \vee \neg x_3)$
- We must force that the "edges" (paths) of the triangle are traversed by a Hamilton Path **if and only if** the corresponding literal is false.

Gadgets (3/3)

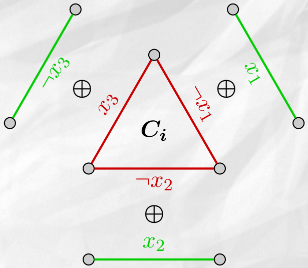
The constraint gadget - one per clause



- Let's take $C_i = (x_1 \vee x_2 \vee \neg x_3)$
- We must force that the "edges" (paths) of the triangle are traversed by a Hamilton Path **if and only if** the corresponding literal is false.

Gadgets (3/3)

The constraint gadget - one per clause



- Let's take $C_i = (x_1 \vee x_2 \vee \neg x_3)$
- We must force that the "edges" (paths) of the triangle are traversed by a Hamilton Path **if and only if** the corresponding literal is false.
- **Then the clause is True, or else there would be no Hamilton Path!**

Constructing the full $R(\phi)$

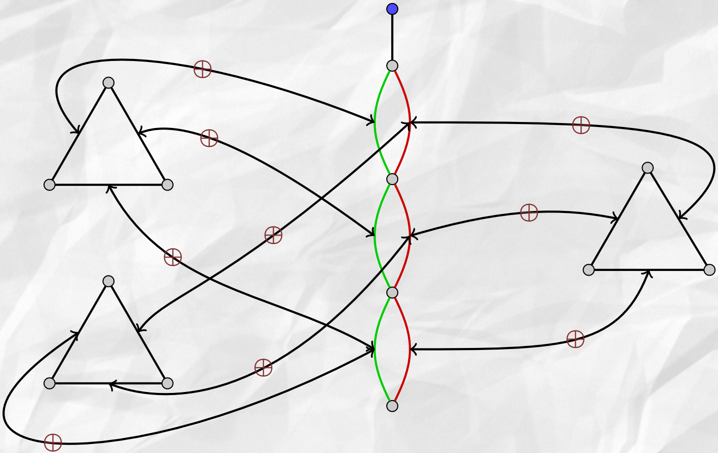
Let $\phi = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$

Constructing the full $R(\phi)$

Let $\phi = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$

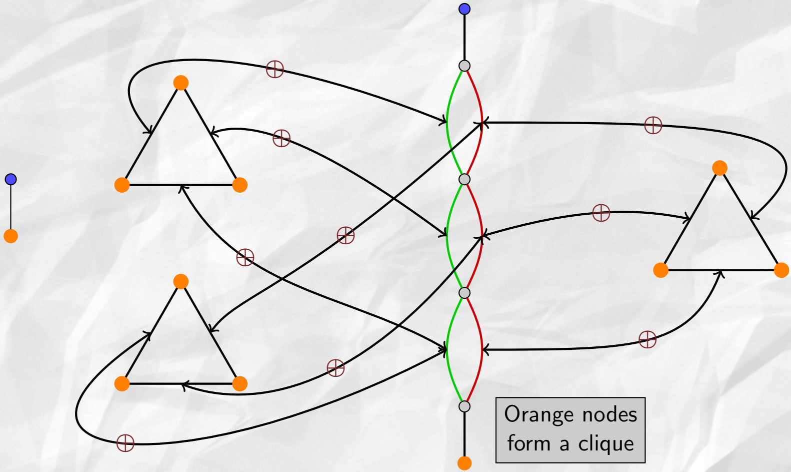
Constructing the full $R(\phi)$

Let $\phi = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$



Constructing the full $R(\phi)$

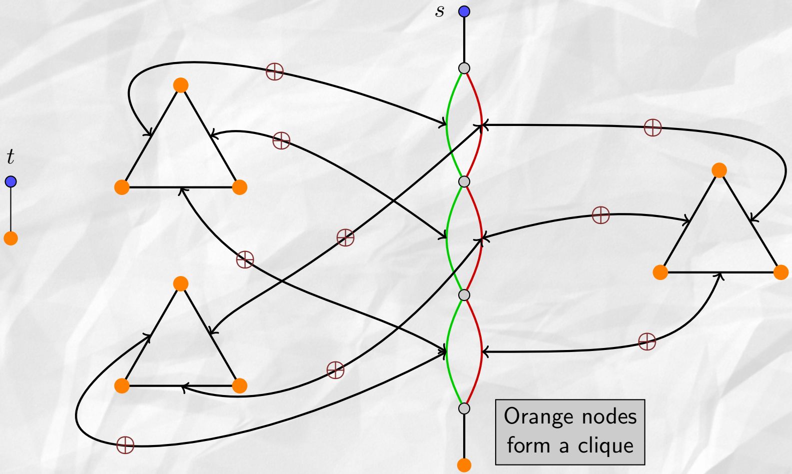
Let $\phi = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$



Orange nodes form a clique

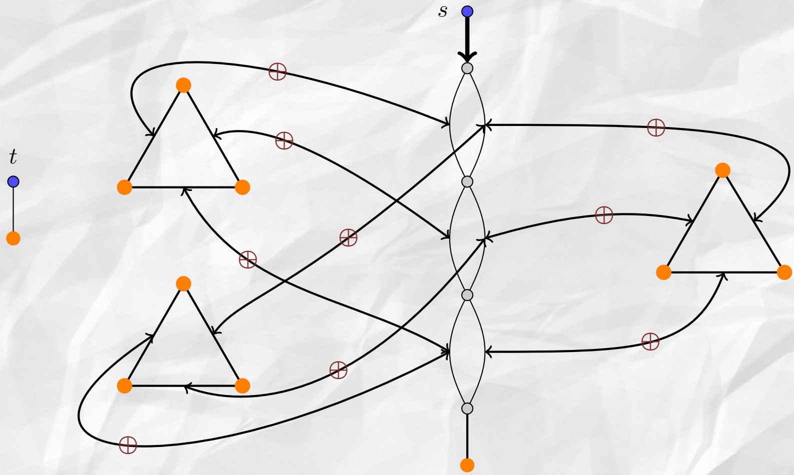
Constructing the full $R(\phi)$

Let $\phi = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$



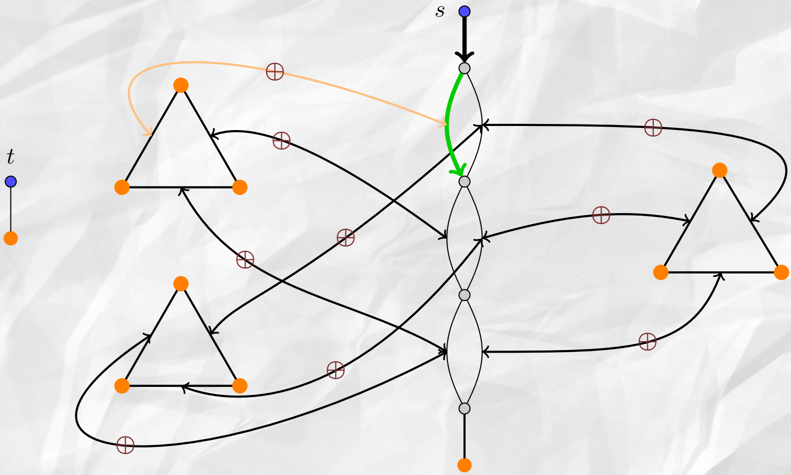
ϕ is satisfiable $\Rightarrow R(\phi)$ has a Hamilton Path

Let $\phi = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$



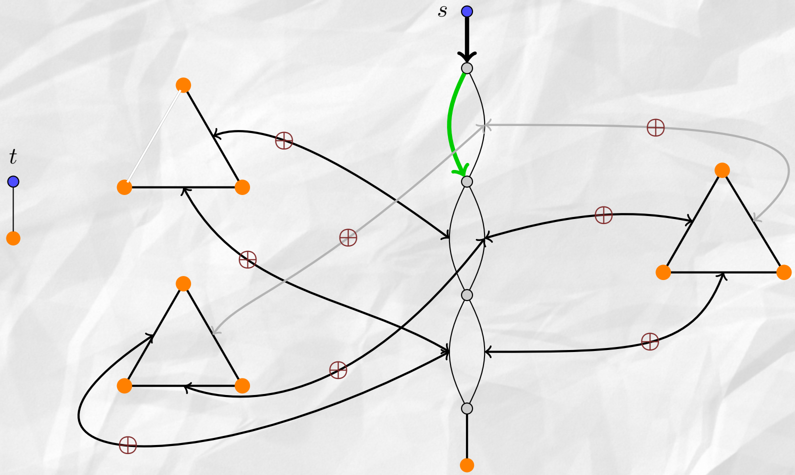
ϕ is satisfiable $\Rightarrow R(\phi)$ has a Hamilton Path

Let $\phi = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$



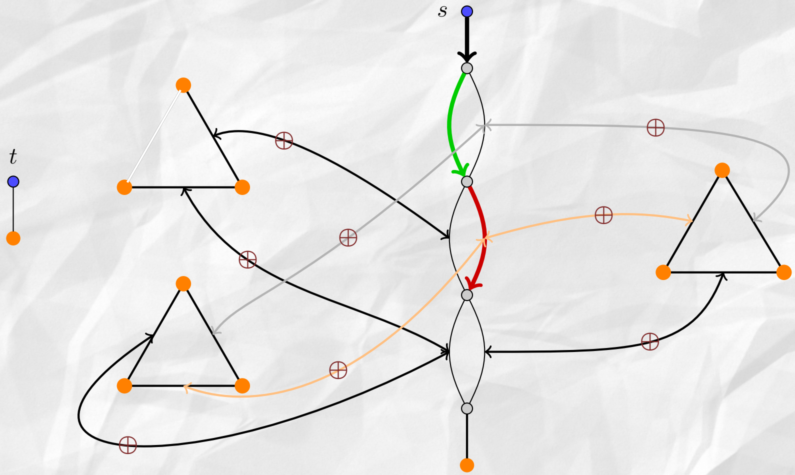
ϕ is satisfiable $\Rightarrow R(\phi)$ has a Hamilton Path

Let $\phi = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$



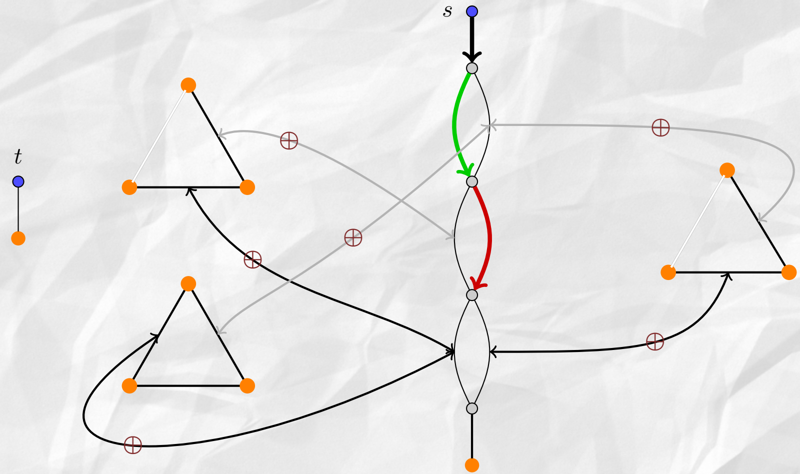
ϕ is satisfiable $\Rightarrow R(\phi)$ has a Hamilton Path

Let $\phi = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$



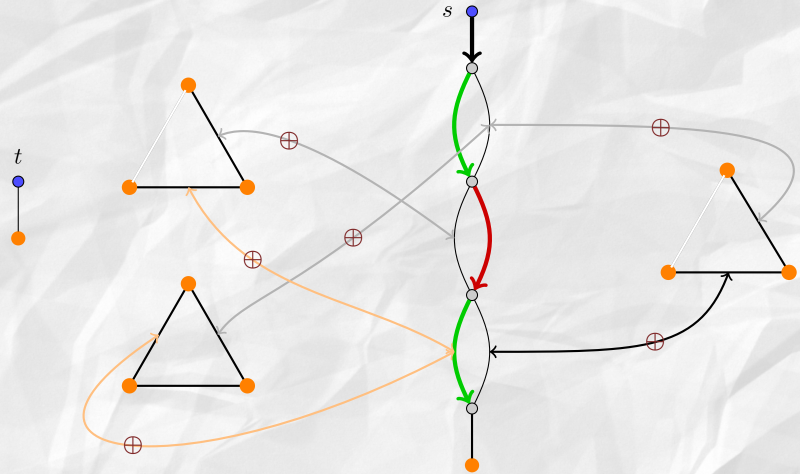
ϕ is satisfiable $\Rightarrow R(\phi)$ has a Hamilton Path

Let $\phi = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$



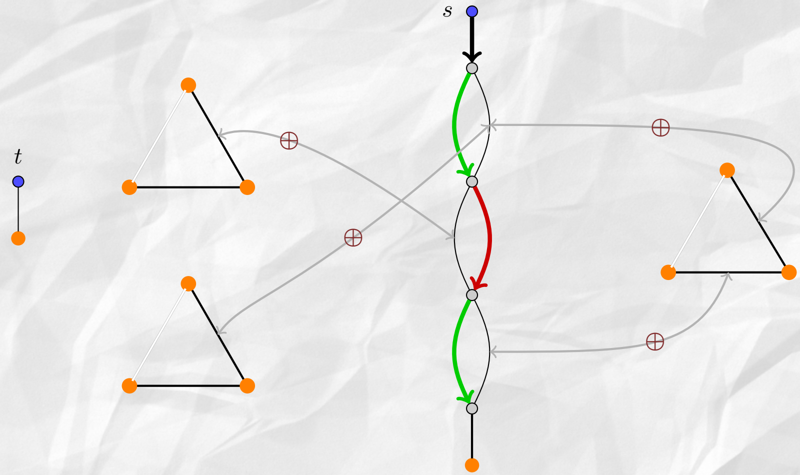
ϕ is satisfiable $\Rightarrow R(\phi)$ has a Hamilton Path

Let $\phi = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$



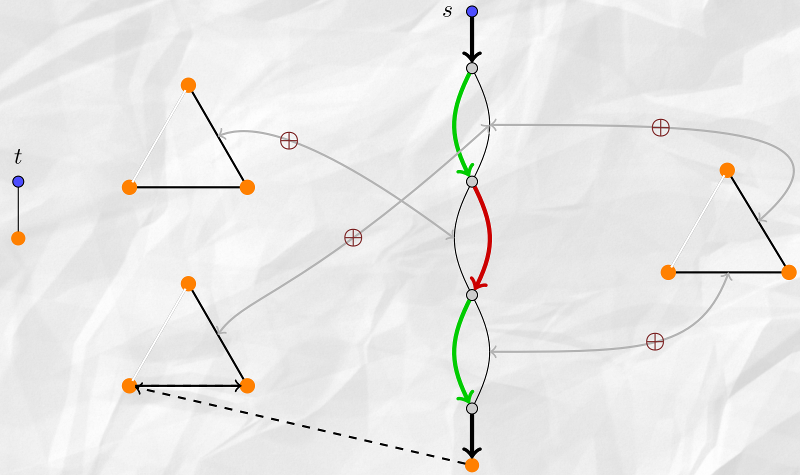
ϕ is satisfiable $\Rightarrow R(\phi)$ has a Hamilton Path

Let $\phi = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$



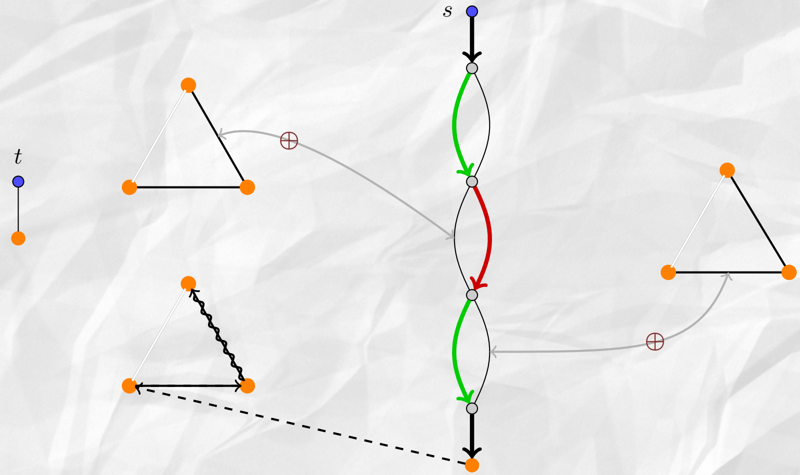
ϕ is satisfiable $\Rightarrow R(\phi)$ has a Hamilton Path

Let $\phi = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$



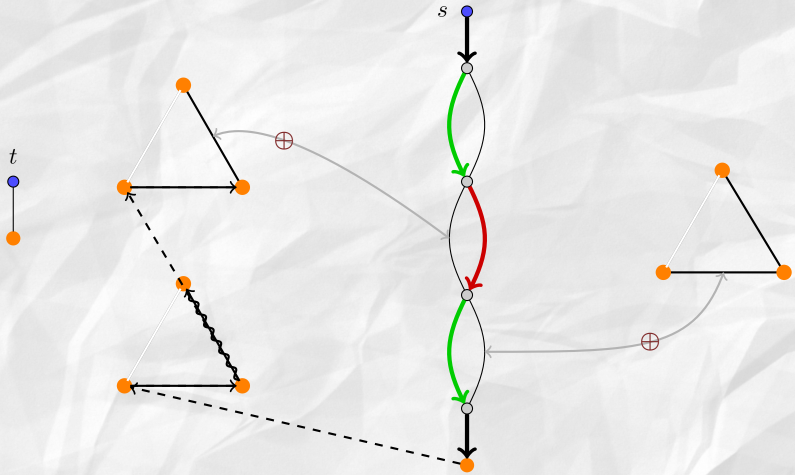
ϕ is satisfiable $\Rightarrow R(\phi)$ has a Hamilton Path

Let $\phi = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$



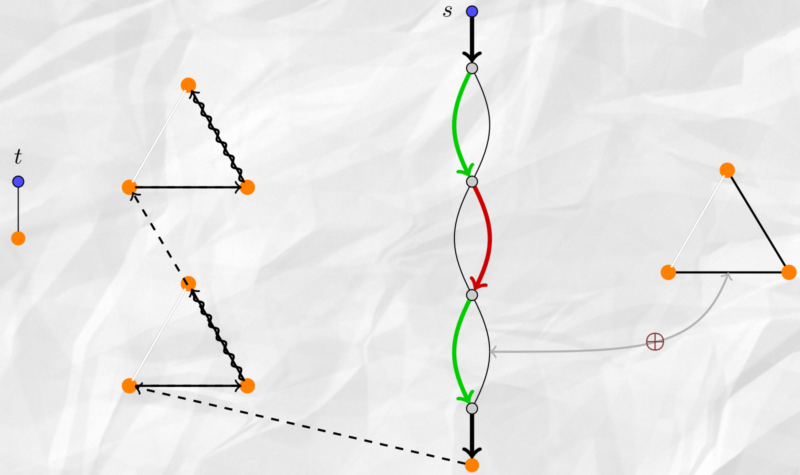
ϕ is satisfiable $\Rightarrow R(\phi)$ has a Hamilton Path

Let $\phi = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$



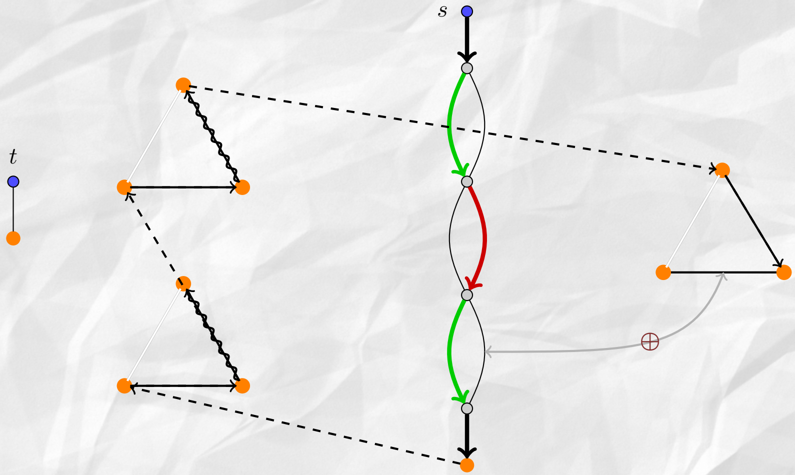
ϕ is satisfiable $\Rightarrow R(\phi)$ has a Hamilton Path

Let $\phi = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$



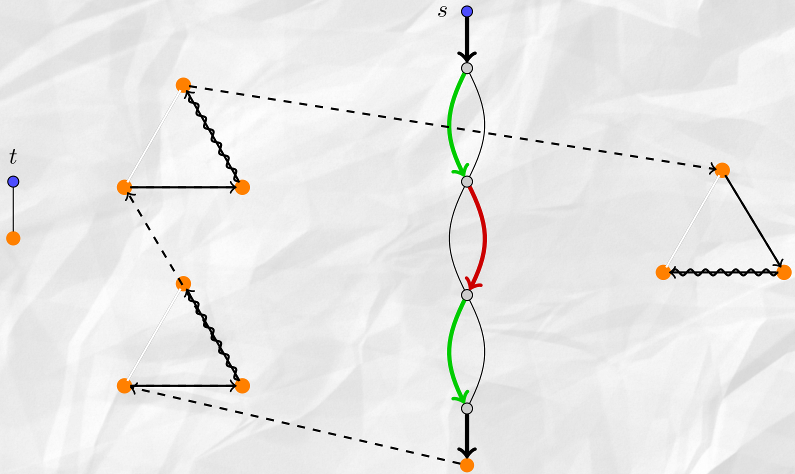
ϕ is satisfiable $\Rightarrow R(\phi)$ has a Hamilton Path

Let $\phi = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$



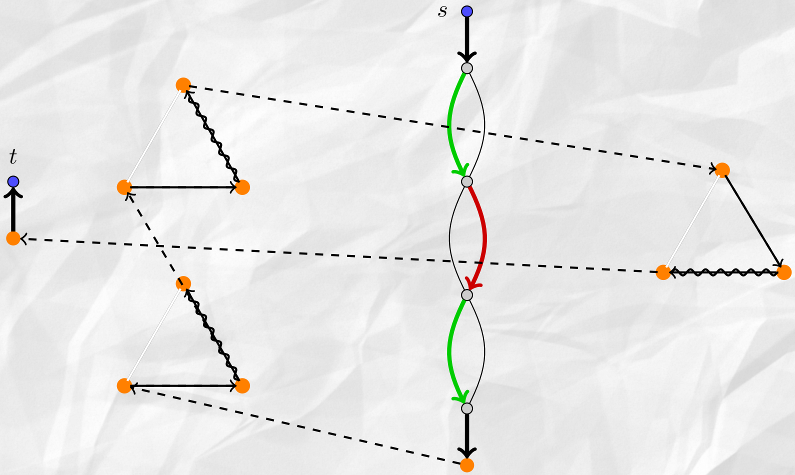
ϕ is satisfiable $\Rightarrow R(\phi)$ has a Hamilton Path

Let $\phi = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$



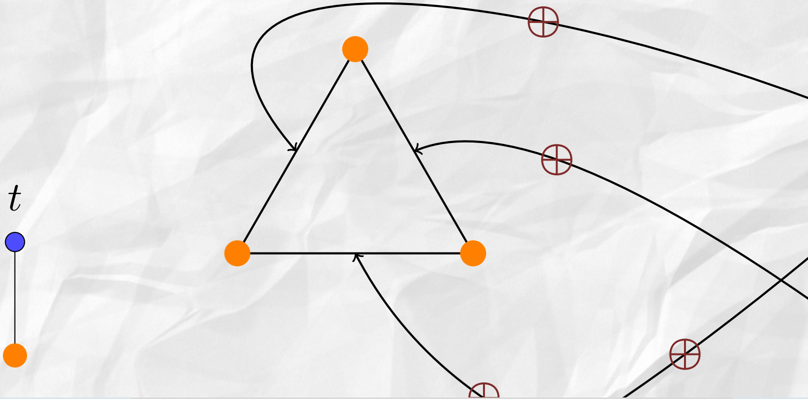
ϕ is satisfiable $\Rightarrow R(\phi)$ has a Hamilton Path

Let $\phi = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$



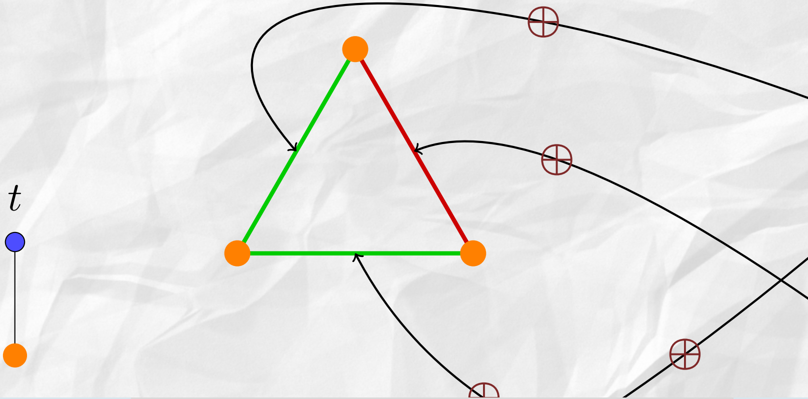
$R(\phi)$ has a Hamilton Path $\Rightarrow \phi$ is satisfiable

Remember the costraint gadget.



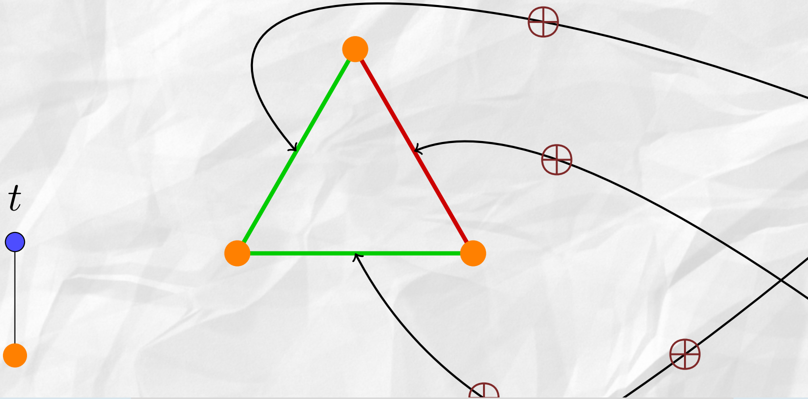
$R(\phi)$ has a Hamilton Path $\Rightarrow \phi$ is satisfiable

Remember the costraint gadget. If the red edge-"xor" path belongs to the Hamilton Path, then both green edges do not belong to the path.



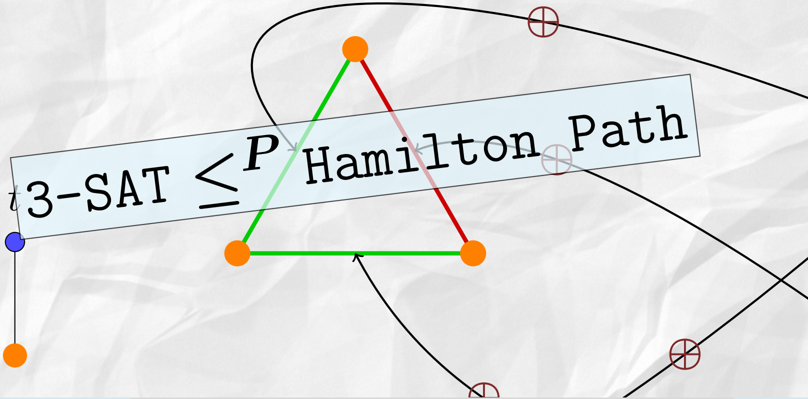
$R(\phi)$ has a Hamilton Path $\Rightarrow \phi$ is satisfiable

Remember the costraint gadget. If the red edge-"xor" path belongs to the Hamilton Path, then both green edges do not belong to the path. But this defines a truth assignment, where no clause gets all 3 literals false.



$R(\phi)$ has a Hamilton Path $\Rightarrow \phi$ is satisfiable

Remember the costraint gadget. If the red edge-"xor" path belongs to the Hamilton Path, then both green edges do not belong to the path. But this defines a truth assignment, where no clause gets all 3 literals false.



Outline

0/1 Integer Programming

3-colorability

Hamilton Path (HP)

Traveling Salesman Problem (TSP)

The Traveling Salesman Problem

Definition 4.1 (TSP).

Given a set of n cities and the distance between any two of them, find the shortest tour covering all cities.

Definition 4.2 (TSP (decision problem)).

Input: a complete graph G with weighted edges, budget (target cost) B
Output: is there a tour (cycle) visiting every vertex of G with total cost $\leq B$?

- Verify that TSP(D) belongs to class *NP*...
- We shall use Hamilton Path as ou known *NP*-complete problem.

The Traveling Salesman Problem

Definition 4.1 (TSP).

Given a set of n cities and the distance between any two of them, find the shortest tour covering all cities.

Definition 4.2 (TSP (decision problem)).

Input: a complete graph G with weighted edges, budget (target cost) B
Output: is there a tour (cycle) visiting every vertex of G with total cost $\leq B$?

- Verify that TSP(D) belongs to class *NP*...
- We shall use Hamilton Path as ou known *NP*-complete problem.

The Traveling Salesman Problem

Definition 4.1 (TSP).

Given a set of n cities and the distance between any two of them, find the shortest tour covering all cities.

Definition 4.2 (TSP (decision problem)).

Input: a complete graph G with weighted edges, budget (target cost) B
Output: is there a tour (cycle) visiting every vertex of G with total cost $\leq B$?

- Verify that `TSP(D)` belongs to class *NP*...
- We shall use `Hamilton Path` as ou known *NP*-complete problem.

The Traveling Salesman Problem

Definition 4.1 (TSP).

Given a set of n cities and the distance between any two of them, find the shortest tour covering all cities.

Definition 4.2 (TSP (decision problem)).

Input: a complete graph G with weighted edges, budget (target cost) B
Output: is there a tour (cycle) visiting every vertex of G with total cost $\leq B$?

- Verify that `TSP(D)` belongs to class *NP*...
- We shall use `Hamilton Path` as ou known *NP*-complete problem.

`Hamilton Path` $\leq^P$ `TSP(D)`

- Take any instance of `Hamilton Path` (i.e. any graph G with n vertices) and take a copy of it, $\bar{G}$.

Hamilton Path $\leq^P$ TSP(D)

- Take any instance of `Hamilton Path` (i.e. any graph G with n vertices) and take a copy of it, $\bar{G}$.
- Set all edges of $\bar{G}$ to have a weight equal to **1**.

Hamilton Path $\leq^P$ TSP(D)

- Take any instance of `Hamilton Path` (i.e. any graph G with n vertices) and take a copy of it, $\bar{G}$.
- Set all edges of $\bar{G}$ to have a weight equal to **1**.
- Insert all missing edges of $\bar{G}$ with weight **2**.

- Take any instance of `Hamilton Path` (i.e. any graph G with n vertices) and take a copy of it, $\bar{G}$.
- Set all edges of $\bar{G}$ to have a weight equal to **1**.
- Insert all missing edges of $\bar{G}$ with weight **2**.
- To finalize the instance of `TSP(D)`, take $B = n + 1$.

Hamilton Path $\leq^P$ TSP(D)

- Take any instance of `Hamilton Path` (i.e. any graph G with n vertices) and take a copy of it, $\bar{G}$.
- Set all edges of $\bar{G}$ to have a weight equal to **1**.
- Insert all missing edges of $\bar{G}$ with weight **2**.
- To finalize the instance of `TSP(D)`, take $B = n + 1$.

- G has a Hamilton Path $\Rightarrow \bar{G}$ has a tour of cost $\leq n + 1$...

`Hamilton Path` $\leq^P$ `TSP(D)`

- Take any instance of `Hamilton Path` (i.e. any graph G with n vertices) and take a copy of it, $\bar{G}$.
- Set all edges of $\bar{G}$ to have a weight equal to **1**.
- Insert all missing edges of $\bar{G}$ with weight **2**.
- To finalize the instance of `TSP(D)`, take $B = n + 1$.

- G has a Hamilton Path $\Rightarrow$ $\bar{G}$ has a tour of cost $\leq n + 1$...
- $\bar{G}$ has a tour of cost $\leq n + 1 \Rightarrow G$ has a Hamilton Path...

Hamilton Path $\leq^P$ TSP(D)

- Take any instance of Hamilton Path (i.e. any graph G with n vertices) and take a copy of it, $\bar{G}$.
- Set all edges of $\bar{G}$ to have a weight equal to **1**.
- Insert all missing edges of $\bar{G}$ with weight **2**.
- To finalize the instance of TSP(D), take $B = n + 1$.

Hamilton Path $\leq^P$ TSP(D)

- G has a Hamilton Path $\Rightarrow \bar{G}$ has a tour of cost $\leq n + 1$...
- $\bar{G}$ has a tour of cost $\leq n + 1 \Rightarrow G$ has a Hamilton Path...

Thank you!