On Undirected Two-commodity Integral Flow, Disjoint Paths and Strict Terminal Connection Problems

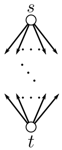
Alexsander A. de Melo[1] Celina M. H. de Figueiredo[1] Uéverton S. Souza[2]

[1]Federal University of Rio de Janeiro, Brazil
[2]Federal Fluminense University, Brazil

September 30, 2020

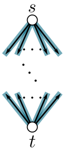
Seminário de Combinatória do IME-UFF

- MAXIMUM FLOW
  - **Goal:** send the maximum possible amount of flow from the *source* s into the *sink* t;

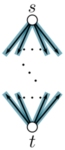


Introduction

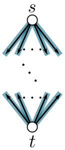
- Maximum flow
  - **Goal:** send the maximum possible amount of flow from the *source* s into the *sink* t;



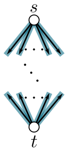
- MAXIMUM FLOW
  - **Goal:** send the maximum possible amount of flow from the *source* s into the *sink* t;
  - **Constraints:** Edge capacity and Flow conservation.



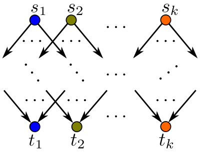
- MAXIMUM FLOW (SINGLE-COMMODITY FLOW)
  - **Goal:** send the maximum possible amount of flow from the *source* s into the *sink* t;
  - **Constraints:** Edge capacity and Flow conservation.



- MAXIMUM FLOW (SINGLE-COMMODITY FLOW)
  - **Goal:** send the maximum possible amount of flow from the *source* s into the *sink* t;
  - **Constraints:** Edge capacity and Flow conservation.

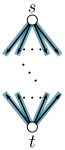


- MULTICOMMODITY FLOW
  - Multiple commodities: $\{s_i, t_i\}$;
  - Each commodity $\{s_i, t_i\}$ has a different demand d_i;

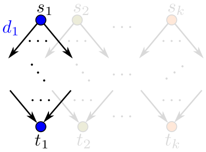


Introduction

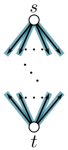
- MAXIMUM FLOW (SINGLE-COMMODITY FLOW)
  - **Goal:** send the maximum possible amount of flow from the *source* s into the *sink* t;
  - **Constraints:** Edge capacity and Flow conservation.



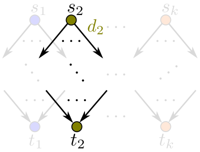
- MULTICOMMODITY FLOW
  - Multiple commodities: $\{s_i, t_i\}$;
  - Each commodity $\{s_i, t_i\}$ has a different demand d_i;



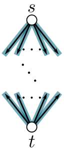
- MAXIMUM FLOW (SINGLE-COMMODITY FLOW)
  - **Goal:** send the maximum possible amount of flow from the *source* s into the *sink* t;
  - **Constraints:** Edge capacity and Flow conservation.



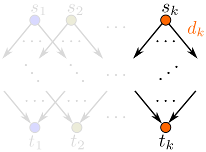
- MULTICOMMODITY FLOW
  - Multiple commodities: $\{s_i, t_i\}$;
  - Each commodity $\{s_i, t_i\}$ has a different demand d_i;



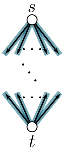
- MAXIMUM FLOW (SINGLE-COMMODITY FLOW)
  - **Goal:** send the maximum possible amount of flow from the *source* s into the *sink* t;
  - **Constraints:** Edge capacity and Flow conservation.



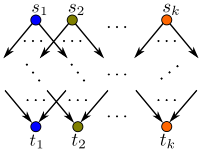
- MULTICOMMODITY FLOW
  - Multiple commodities: $\{s_i, t_i\}$;
  - Each commodity $\{s_i, t_i\}$ has a different demand d_i;



- MAXIMUM FLOW (SINGLE-COMMODITY FLOW)
  - **Goal:** send the maximum possible amount of flow from the *source* s into the *sink* t;
  - **Constraints:** Edge capacity and Flow conservation.



- MULTICOMMODITY FLOW
  - Multiple commodities: $\{s_i, t_i\}$;
  - Each commodity $\{s_i, t_i\}$ has a different demand d_i;
  - **Goal:** for each commodity $\{s_i, t_i\}$, send at least d_i unities of flow from the *source* s_i into the *sink* t_i;
  - **Constraints:** Edge capacity and Flow conservation; The capacities of the edges are shared among the flow of each commodity.



Complexity of Multicommodity Flow

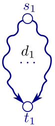
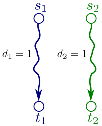
- MAXIMUM FLOW is polynomial-time solvable.
- By using linear programming, MULTICOMMODITY FLOW can be solved in polynomial-time if the flows are **real-valued** functions.
- On the other hand, Karp (1975) proved that MULTICOMMODITY FLOW is NP-complete if the flows must be **integral-valued** functions.

Complexity of Multicommodity Flow

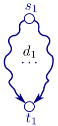
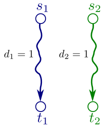
- MAXIMUM FLOW is polynomial-time solvable.
- By using linear programming, MULTICOMMODITY FLOW can be solved in polynomial-time if the flows are **real-valued** functions.
- On the other hand, Karp (1975) proved that MULTICOMMODITY FLOW is NP-complete if the flows must be **integral-valued** functions.
- Even, Itai and Shamir (1976) proved that **TWO-COMMODITY INTEGRAL FLOW** is NP-complete, even if the **demand** d_1 is **unitary**.

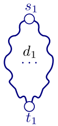
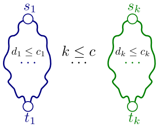
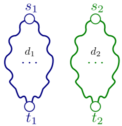
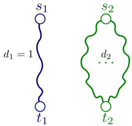
Complexity of Multicommodity Flow

- MAXIMUM FLOW is polynomial-time solvable.
- By using linear programming, MULTICOMMODITY FLOW can be solved in polynomial-time if the flows are **real-valued** functions.
- On the other hand, Karp (1975) proved that MULTICOMMODITY FLOW is NP-complete if the flows must be **integral-valued** functions.
- Even, Itai and Shamir (1976) proved that **TWO**-COMMODITY **INTEGRAL** FLOW is NP-complete, even if the **demand** d_1 is **unitary**.
- Fortune, Hopcroft and Wyllie (1980) proved that **TWO**-COMMODITY **INTEGRAL** FLOW is NP-complete, even if **both demands** d_1 and d_2 are **unitary**.

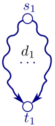
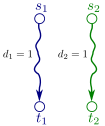
<table>
<tr><th colspan="2">Directed case</th></tr>
<tr><td></td><td></td></tr>
<tr><td>$k = 1$: Poly
Even if d_1 is arbitrary
(Edmonds and Karp, 1972)</td><td>Fixed $k \geq 2$: NP-c
Even if $d_i = 1\ \forall i$
(Fortune et al., 1980)</td></tr>
</table>

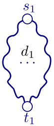
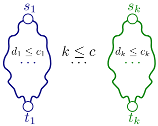
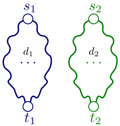
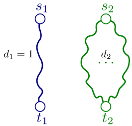
Complexity of Multicommodity Integral Flow

Directed case	
	
$k = 1$: Poly Even if d_1 is arbitrary (Edmonds and Karp, 1972)	Fixed $k \geq 2$: NP-c Even if $d_i = 1$ $\forall i$ (Fortune et al., 1980)

Undirected case			
			
$k = 1$: Poly Even if d_1 is arbitrary (Edmonds and Karp, 1972)	Fixed $k \geq 1$: Poly If d_i is fixed $\forall i$ (Roberson and Seymour, 1995)	Fixed $k \geq 2$: NP-c d_1 and d_2 are both arbitrarily large (Even et al., 1976)	**Fixed $k \geq 2$: Open** if d_i is fixed $\forall i \neq k$ and only d_k is arbitrarily large

Complexity of Multicommodity Integral Flow

Directed case	
	
$k = 1$: Poly Even if d_1 is arbitrary (Edmonds and Karp, 1972)	Fixed $k \geq 2$: NP-c Even if $d_i = 1$ $\forall i$ (Fortune et al., 1980)

Undirected case			
			
$k = 1$: Poly Even if d_1 is arbitrary (Edmonds and Karp, 1972)	Fixed $k \geq 1$: Poly If d_i is fixed $\forall i$ (Roberson and Seymour, 1995)	Fixed $k \geq 2$: NP-c d_1 and d_2 are both arbitrarily large (Even et al., 1976)	**Fixed $k \geq 2$: NP-c** Even if $d_i = 1$ $\forall i \neq k$ and only d_k is arbitrarily large

Simple undirected two-commodity integral flow

Simple undirected two-commodity integral flow (Simple U2CIF)

Input: An undirected graph G, two *commodities* $\{s_1, t_1\}$ and $\{s_2, t_2\}$, where s_1, t_1, s_2 and t_2 are vertices of G, and two *demands* $d_1, d_2 \in \mathbb{Z}^+$.

Question: Are there two flow functions $f_1, f_2 \colon \{\overrightarrow{uv}, \overrightarrow{vu} \mid uv \in E(G)\} \to \mathbb{Z}_0^+$ such that

1. for each $i \in \{1, 2\}$ and each edge $uv \in E(G)$,
$$f_i(\overrightarrow{uv}) = 0 \text{ or } f_i(\overrightarrow{vu}) = 0;$$
2. for each $i \in \{1, 2\}$ and each vertex $v \in V(G) \setminus \{s_i, t_i\}$, the flow function f_i is *conserved* at v, *i.e.*
$$\sum_{u \in N_G(v)} f_i(\overrightarrow{uv}) = \sum_{u \in N_G(v)} f_i(\overrightarrow{vu});$$
3. for each $i \in \{1, 2\}$, the net flow from s_i is at least d_i, i.e.
$$\sum_{v \in N_G(s_i)} (f_i(\overrightarrow{s_i v}) - f_i(\overrightarrow{v s_i})) \geq d_i;$$
4. for each edge $uv \in E(G)$, the total flow through uv is at most 1, i.e.
$$\max\{f_1(\overrightarrow{uv}), f_1(\overrightarrow{vu})\} + \max\{f_2(\overrightarrow{uv}), f_2(\overrightarrow{vu})\} \leq 1?$$

NP-completeness of SIMPLE U2CIF with a unitary demand

- SIMPLE U2CIF remains **NP-complete** when the demand of one commodity is unitary.

NP-completeness of SIMPLE U2CIF with a unitary demand

- SIMPLE U2CIF remains **NP-complete** when the demand of one commodity is unitary.
- Polynomial-time reduction from 3-SAT:

$$I = (X, \mathcal{C}) \longmapsto g(I) = (G, \{s_1, t_1\}, \{s_2, t_2\}, d_1, d_2)$$

NP-completeness of SIMPLE U2CIF with a unitary demand

- SIMPLE U2CIF remains **NP-complete** when the demand of one commodity is unitary.
- Polynomial-time reduction from 3-SAT:

$$I = (X, \mathcal{C}) \longmapsto g(I) = (G, \{s_1, t_1\}, \{s_2, t_2\}, d_1, d_2)$$

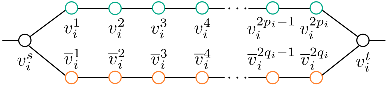
- Define $d_1 = 1$ and $d_2 = 5m$, where $m = |\mathcal{C}|$.

NP-completeness of Simple U2CIF with a unitary demand

- Simple U2CIF remains **NP-complete** when the demand of one commodity is unitary.
- Polynomial-time reduction from 3-SAT:

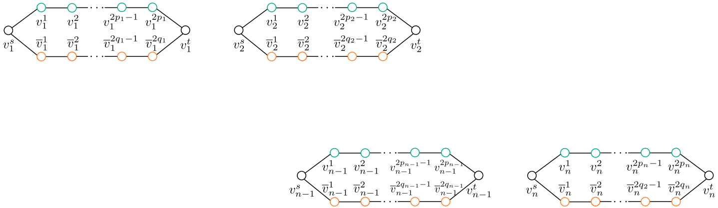
$$I = (X, \mathcal{C}) \longmapsto g(I) = (G, \{s_1, t_1\}, \{s_2, t_2\}, d_1, d_2)$$

- Define $d_1 = 1$ and $d_2 = 5m$, where $m = |\mathcal{C}|$.
- For each variable $x_i \in X$, create the gadget H_i:

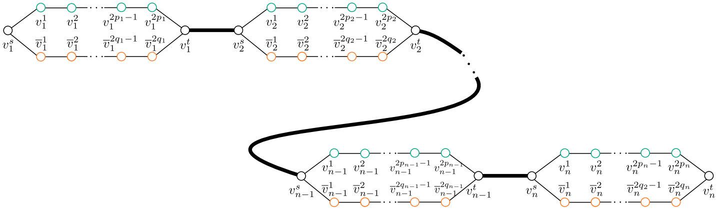


p_i: number of occurrences of x_i q_i: number of occurrences of $\overline{x}_i$

- Connect the variable gadgets in series:

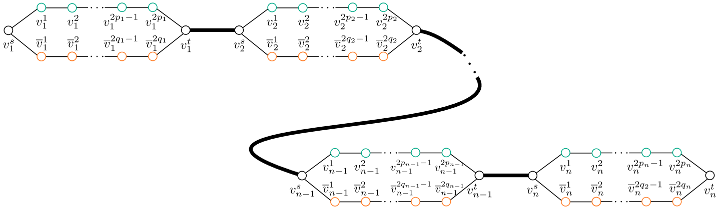


- Connect the variable gadgets in series:

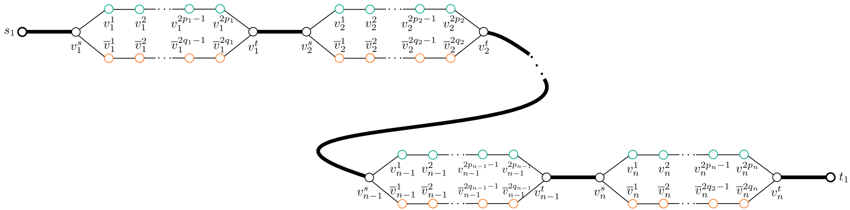


NP-completeness of SIMPLE U2CIF with a Unitary Demand

- Connect the variable gadgets in series:
- Add the edges $s_1 v_1^s$ and $v_n^t t_1$.

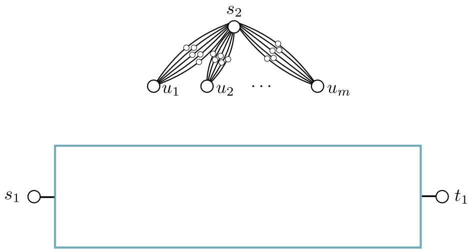


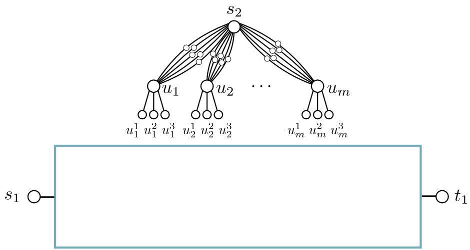
- Connect the variable gadgets in series:
- Add the edges $s_1 v_1^s$ and $v_n^t t_1$.



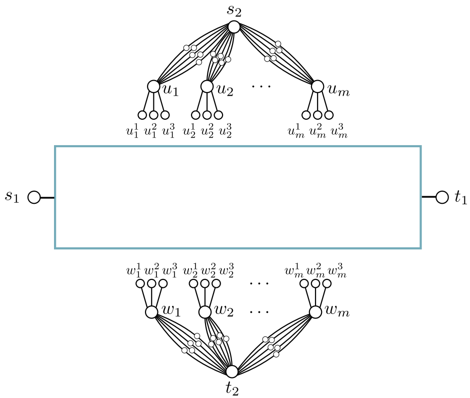
NP-completeness of Simple U2CIF with a unitary demand

NP-completeness of SIMPLE U2CIF with a unitary demand

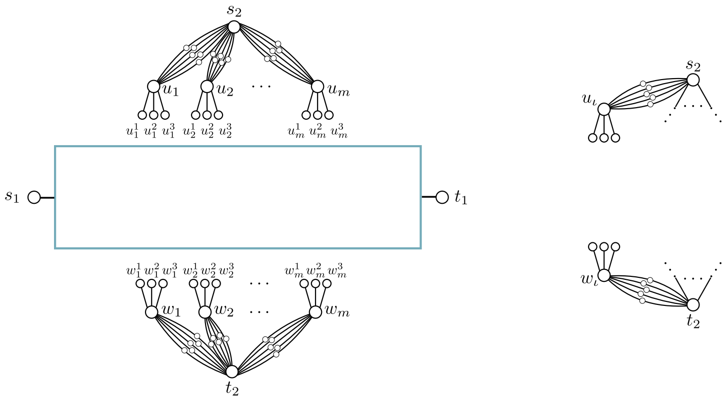



s_2
u_1
u_2
$\cdots$
u_m
u_1^1 u_1^2 u_1^3 u_2^1 u_2^2 u_2^3 u_m^1 u_m^2 u_m^3
s_1
t_1

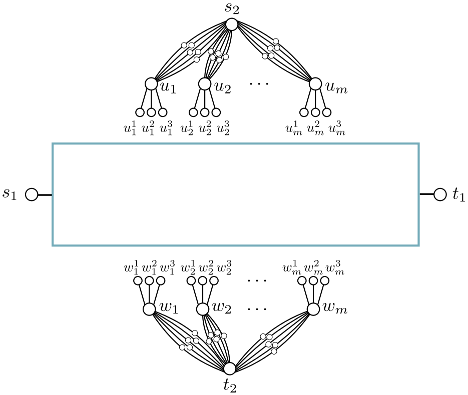
NP-completeness of SIMPLE U2CIF with a unitary demand

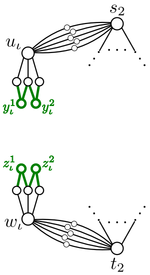


NP-completeness of SIMPLE U2CIF with a unitary demand

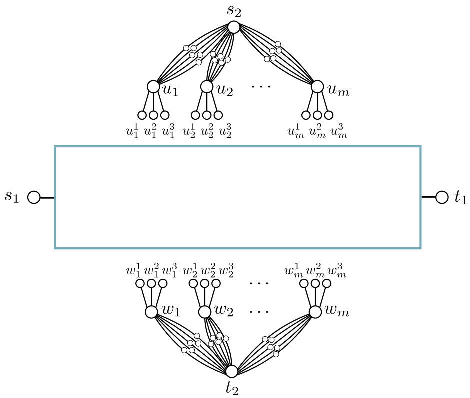


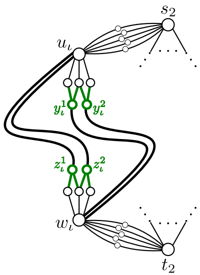
NP-completeness of SIMPLE U2CIF with a unitary demand



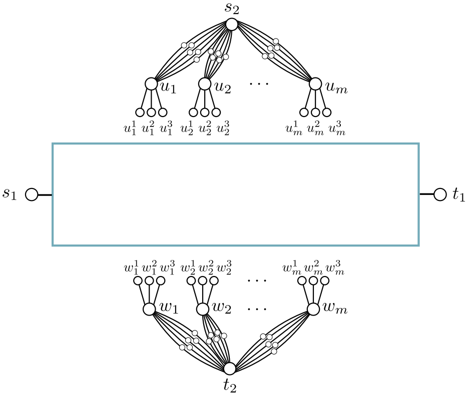


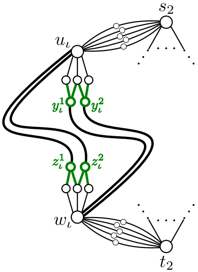
NP-completeness of SIMPLE U2CIF with a unitary demand





NP-completeness of Simple U2CIF with a unitary demand

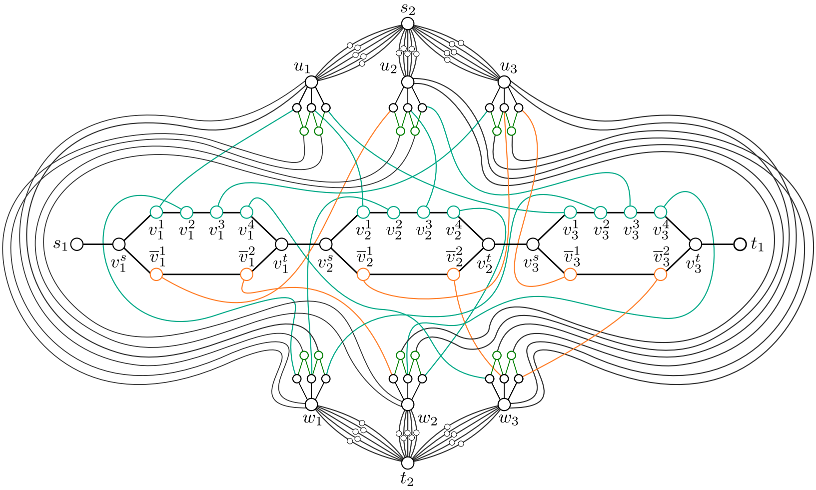




- Add the edges $u_\iota^j v_i^{2\ell-1}$ and $v_i^{2\ell} w_\iota^j$ if the j-th literal in C_ι corresponds to the ℓ-th occurrence of the **positive** literal x_i;
- Add the edges $u_\iota^j \overline{v}_i^{2\ell-1}$ and $\overline{v}_i^{2\ell} w_\iota^j$ if the j-th literal in C_ι corresponds to the ℓ-th occurrence of the **negative** literal $\overline{x}_i$.

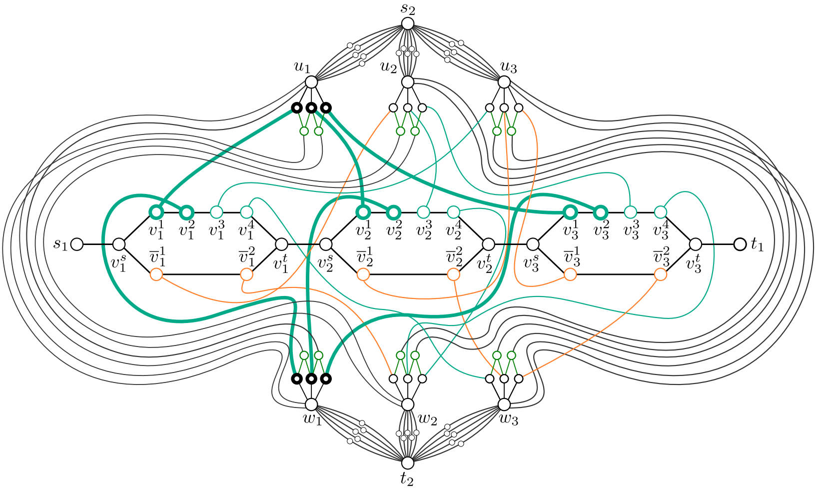
NP-completeness of SIMPLE U2CIF with a Unitary Demand

$$I = (x_1 \vee x_2 \vee x_3) \wedge (\overline{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee \overline{x}_3)$$

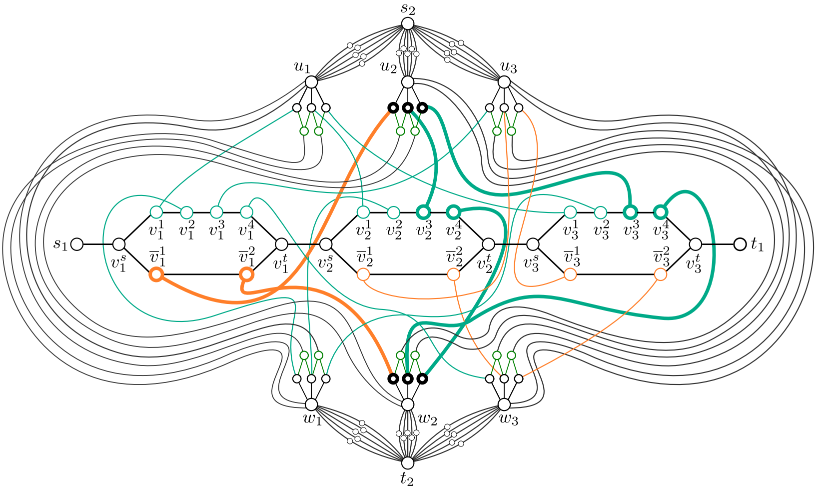


NP-completeness of SIMPLE U2CIF with a Unitary Demand

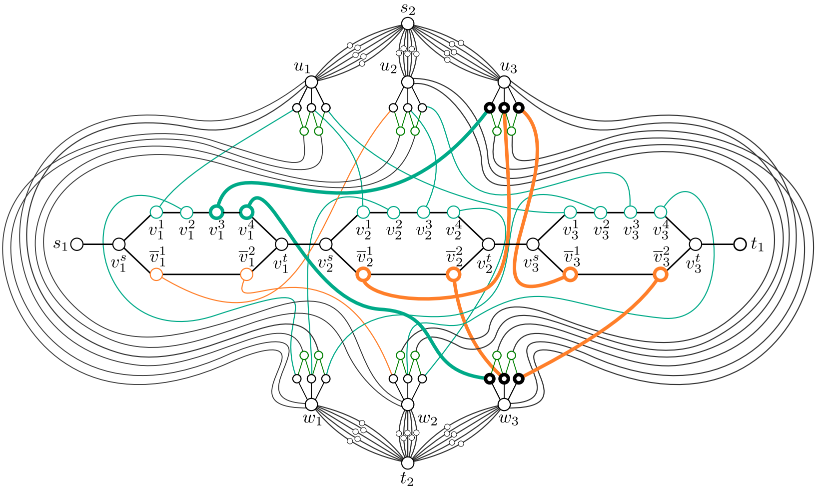
$$I = (\boldsymbol{x_1} \vee \boldsymbol{x_2} \vee \boldsymbol{x_3}) \wedge (\overline{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee \overline{x}_3)$$



$$I = (x_1 \vee x_2 \vee x_3) \wedge (\boldsymbol{\overline{x}_1} \vee \boldsymbol{x_2} \vee \boldsymbol{x_3}) \wedge (x_1 \vee \overline{x_2} \vee \overline{x}_3)$$



$$I = (x_1 \lor x_2 \lor x_3) \land (\overline{x}_1 \lor x_2 \lor x_3) \land (\boldsymbol{x_1} \lor \overline{\boldsymbol{x_2}} \lor \overline{\boldsymbol{x}}_{\boldsymbol{3}})$$



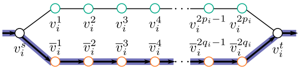
NP-completeness of SIMPLE U2CIF with a Unitary Demand

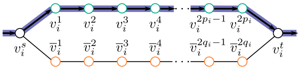
Lemma

If $g(I)$ is a YES *instance of* SIMPLE U2CIF*, then the first commodity flow only uses edges whose endpoints belong to* $\{s_1, t_1\} \cup V(H_1) \cup \cdots \cup V(H_n)$.

Lemma

If $g(I)$ is a YES *instance of* SIMPLE U2CIF*, then the first commodity flow only uses edges whose endpoints belong to $\{s_1, t_1\} \cup V(H_1) \cup \cdots \cup V(H_n)$.*

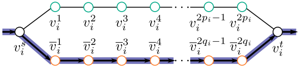


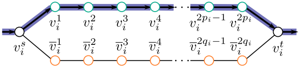


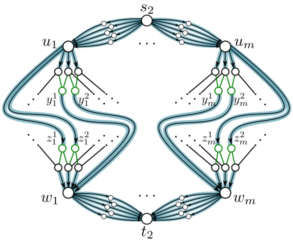
NP-completeness of SIMPLE U2CIF with a Unitary Demand

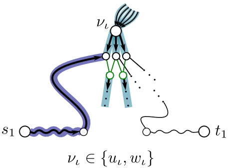
Lemma

If $g(I)$ is a YES *instance of* SIMPLE U2CIF*, then the first commodity flow only uses edges whose endpoints belong to* $\{s_1, t_1\} \cup V(H_1) \cup \cdots \cup V(H_n)$.





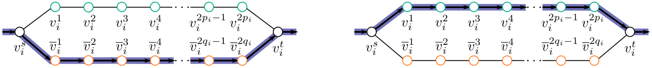




NP-completeness of SIMPLE U2CIF with a Unitary Demand

Lemma

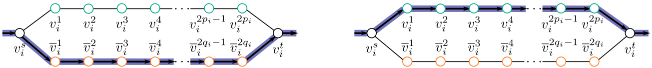
If $g(I)$ is a YES *instance of* SIMPLE U2CIF*, then the first commodity flow only uses edges whose endpoints belong to* $\{s_1, t_1\} \cup V(H_1) \cup \cdots \cup V(H_n)$.



NP-completeness of SIMPLE U2CIF with a Unitary Demand

Lemma

If $g(I)$ is a YES *instance of* SIMPLE U2CIF*, then the first commodity flow only uses edges whose endpoints belong to* $\{s_1, t_1\} \cup V(H_1) \cup \cdots \cup V(H_n)$.



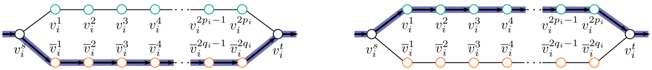
Lemma

$g(I)$ is Yes instance of SIMPLE U2CIF *if and only if I is a* YES *instance of 3-SAT.*

NP-completeness of Simple U2CIF with a Unitary Demand

Lemma

If $g(I)$ is a Yes *instance of* Simple U2CIF*, then the first commodity flow only uses edges whose endpoints belong to $\{s_1, t_1\} \cup V(H_1) \cup \cdots \cup V(H_n)$.*



Lemma

$g(I)$ is Yes instance of Simple U2CIF *if and only if I is a* Yes *instance of 3-SAT.*

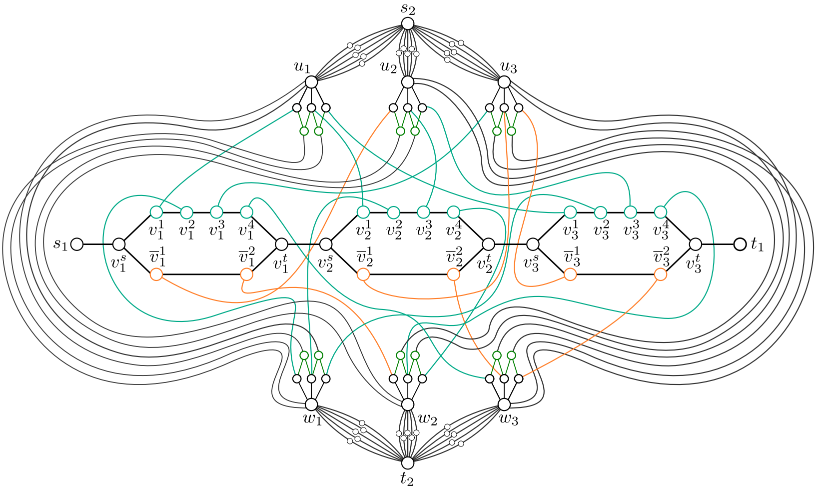
Theorem

Simple U2CIF *is NP-complete even if the* **demand** *of one commodity is* **unitary**.

An example

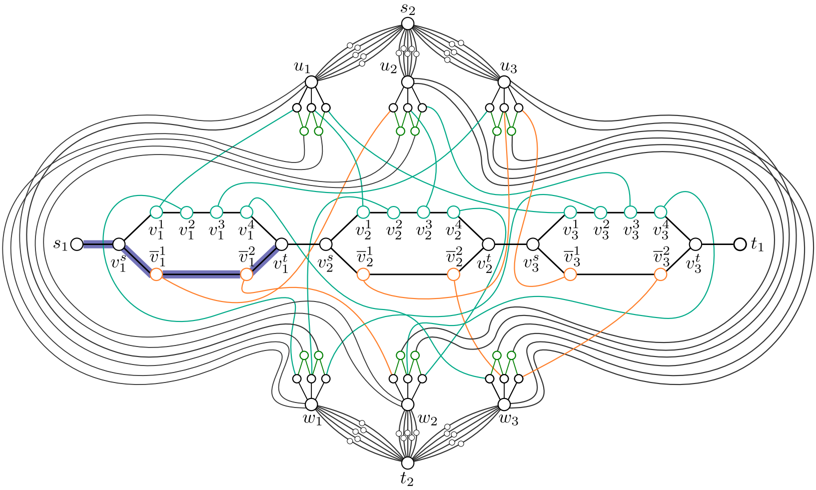
$$I = (x_1 \vee x_2 \vee x_3) \wedge (\overline{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee \overline{x}_3)$$

$\alpha(x_1) = \textit{true} \quad \alpha(x_2) = \textit{true} \quad \alpha(x_3) = \textit{false}$



$$I = (x_1 \vee x_2 \vee x_3) \wedge (\overline{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee \overline{x}_3)$$

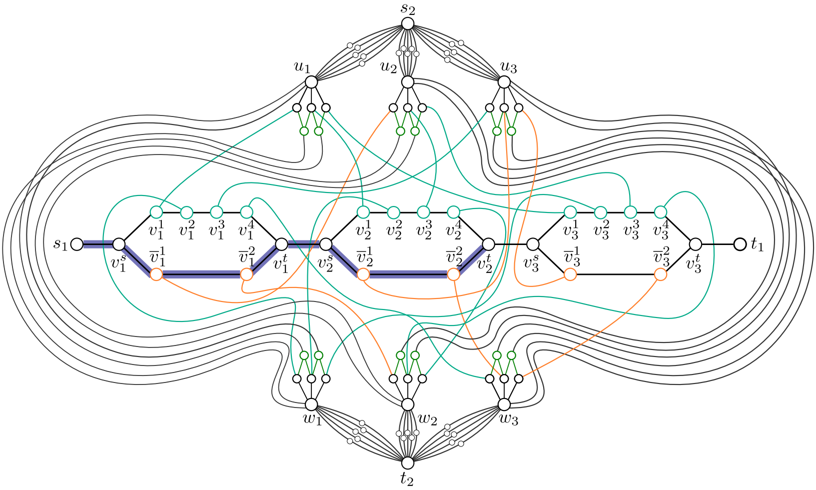
$\alpha(x_1) = \mathit{true}$ $\alpha(x_2) = \mathit{true}$ $\alpha(x_3) = \mathit{false}$



An example

$$I = (x_1 \vee x_2 \vee x_3) \wedge (\overline{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee \overline{x}_3)$$

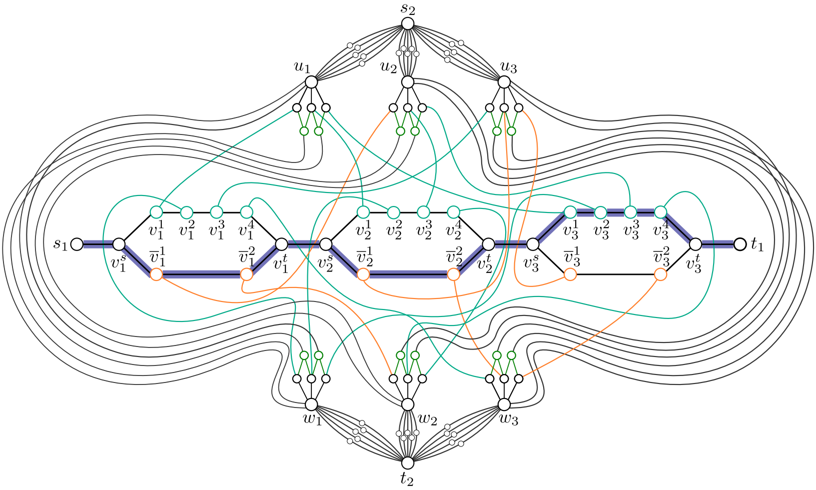
$\alpha(x_1) = \mathit{true}$ $\alpha(x_2) = \mathit{true}$ $\alpha(x_3) = \mathit{false}$



An example

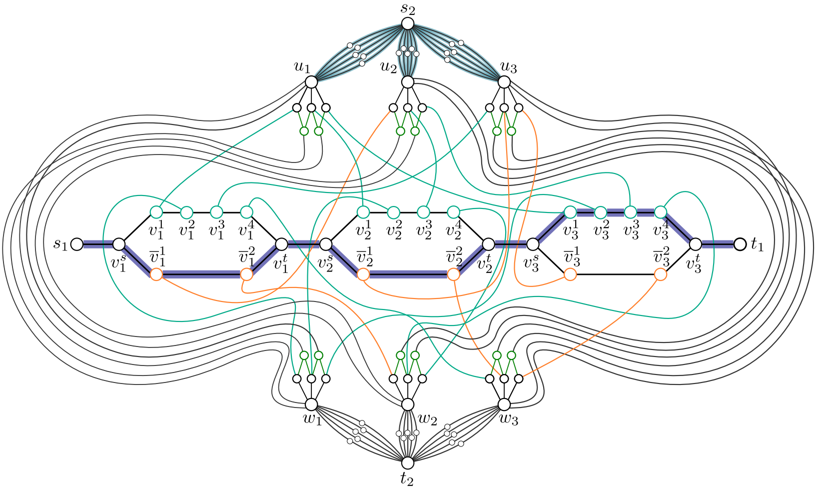
$$I = (x_1 \vee x_2 \vee x_3) \wedge (\overline{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee \overline{x}_3)$$

$\alpha(x_1) = true \quad \alpha(x_2) = true \quad \alpha(x_3) = false$



$$I = (x_1 \vee x_2 \vee x_3) \wedge (\overline{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee \overline{x}_3)$$

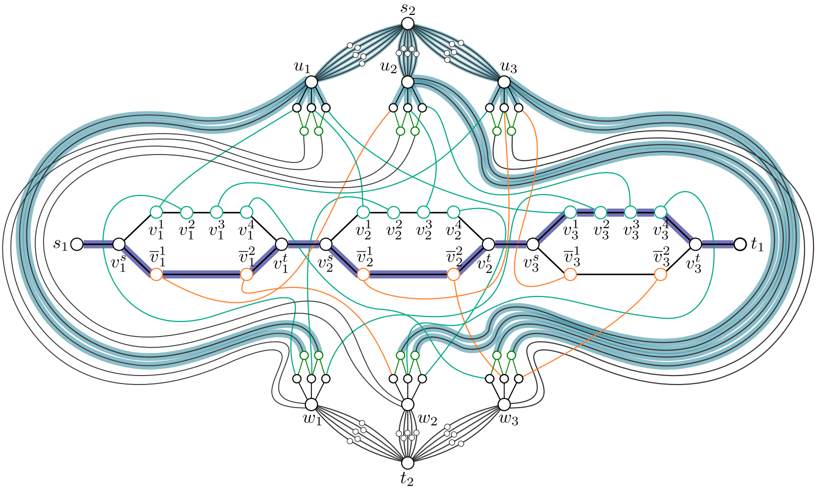
$\alpha(x_1) = \mathit{true} \quad \alpha(x_2) = \mathit{true} \quad \alpha(x_3) = \mathit{false}$



An example

$$I = (x_1 \vee x_2 \vee x_3) \wedge (\overline{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee \overline{x}_3)$$

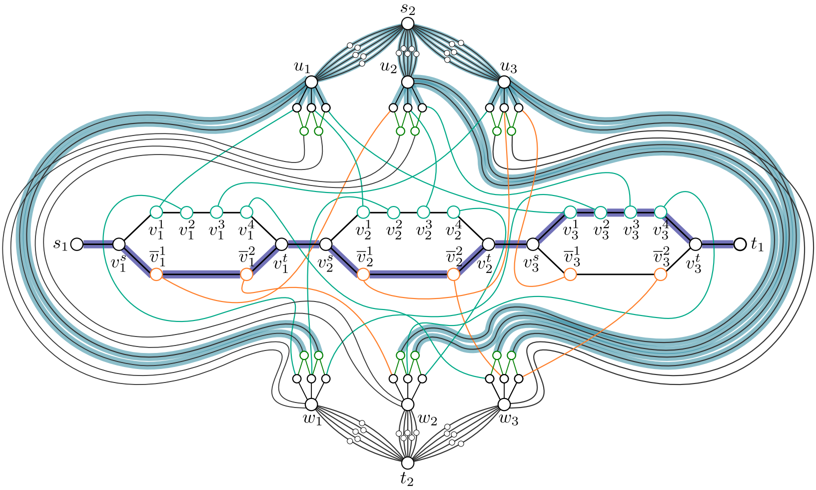
$\alpha(x_1) = \textit{true}$ $\quad \alpha(x_2) = \textit{true}$ $\quad \alpha(x_3) = \textit{false}$



An example

$$I = (\boldsymbol{x_1} \vee x_2 \vee x_3) \wedge (\overline{x}_1 \vee \boldsymbol{x_2} \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee \boldsymbol{\overline{x}_3})$$

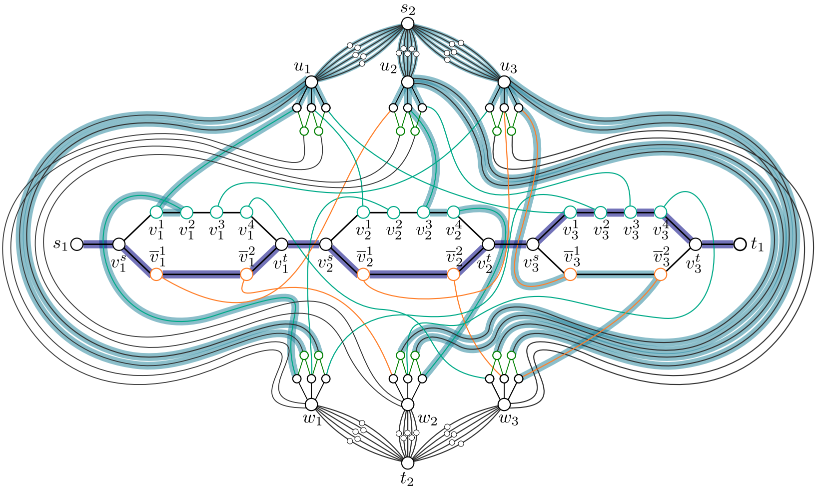
$\alpha(x_1) = \mathit{true}$ $\alpha(x_2) = \mathit{true}$ $\alpha(x_3) = \mathit{false}$



An example

$$I = (\boldsymbol{x_1} \vee x_2 \vee x_3) \wedge (\overline{x}_1 \vee \boldsymbol{x_2} \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee \overline{\boldsymbol{x}}_{\boldsymbol{3}})$$

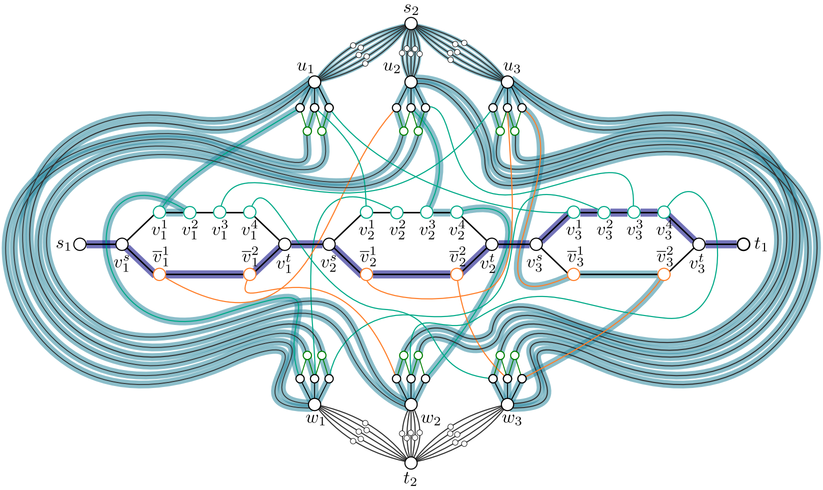
$\alpha(x_1) = true \quad \alpha(x_2) = true \quad \alpha(x_3) = false$



An example

$$I = (\boldsymbol{x_1} \vee x_2 \vee x_3) \wedge (\overline{x}_1 \vee \boldsymbol{x_2} \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee \boldsymbol{\overline{x}_3})$$

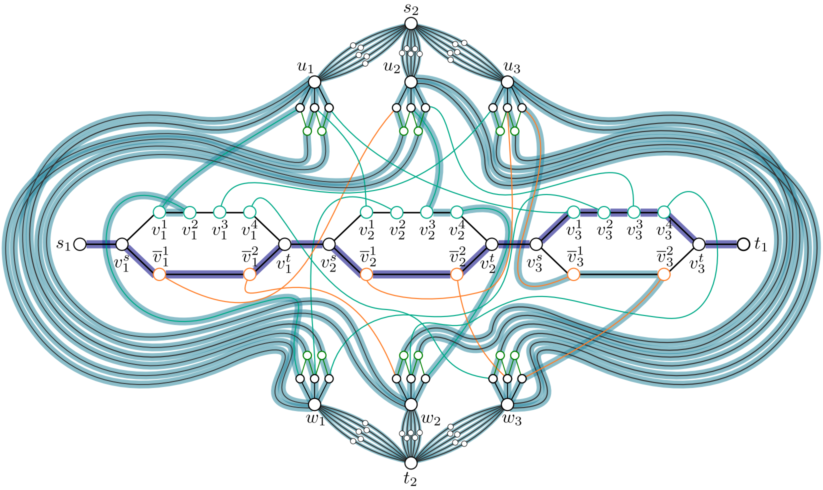
$\alpha(x_1) = true \quad \alpha(x_2) = true \quad \alpha(x_3) = false$



An example

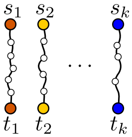
$$I = (\boldsymbol{x_1} \vee x_2 \vee x_3) \wedge (\overline{x}_1 \vee \boldsymbol{x_2} \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee \boldsymbol{\overline{x}_3})$$

$\alpha(x_1) = true$ $\alpha(x_2) = true$ $\alpha(x_3) = false$



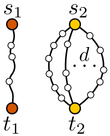
Disjoint Paths

- SIMPLE MULTICOMMODITY INTEGRAL FLOW and *k*-EDGE-DISJOINT PATHS are **polynomially equivalent**.

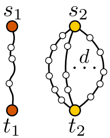


Disjoint Paths

- SIMPLE MULTICOMMODITY INTEGRAL FLOW and k-EDGE-DISJOINT PATHS are **polynomially equivalent**.
- SIMPLE U2CIF with a unitary demand **coincides** with $1 + d$-EDGE-DISJOINT PATHS.



- SIMPLE MULTICOMMODITY INTEGRAL FLOW and *k*-EDGE-DISJOINT PATHS are **polynomially equivalent**.
- SIMPLE U2CIF with a unitary demand **coincides** with $1 + d$-EDGE-DISJOINT PATHS.

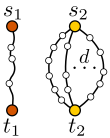


Corollary

$1 + d$-EDGE-DISJOINT PATHS *is NP-complete.*

Disjoint Paths

- SIMPLE MULTICOMMODITY INTEGRAL FLOW and k-EDGE-DISJOINT PATHS are **polynomially equivalent**.
- SIMPLE U2CIF with a unitary demand **coincides** with $1 + d$-EDGE-DISJOINT PATHS.



Corollary

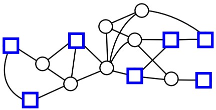
$1 + d$-EDGE-DISJOINT PATHS *is NP-complete.*

- By taking the **line graph**, $1 + d$-**VERTEX**-DISJOINT PATHS is also NP-complete.

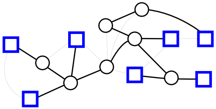
Strict Terminal Connection Problem

Strict Terminal Connection Problem

A *strict connection tree* of G for $W \subseteq V(G)$ is a tree subgraph T of G such that $\text{leaves}(T) = W$.



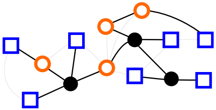
A *strict connection tree* of G for $W \subseteq V(G)$ is a tree subgraph T of G such that $\text{leaves}(T) = W$.



Strict Terminal Connection Problem

A *strict connection tree* of G for $W \subseteq V(G)$ is a tree subgraph T of G such that $\text{leaves}(T) = W$.

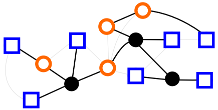
- $W \rightarrow$ *terminals*;
- $V(T) \setminus W$ with degree $= 2$ in $T \rightarrow$ *linkers*;
- $V(T) \setminus W$ with degree ≥ 3 in $T \rightarrow$ ***routers***.



Strict Terminal Connection Problem

A *strict connection tree* of G for $W \subseteq V(G)$ is a tree subgraph T of G such that $\text{leaves}(T) = W$.

- $W \rightarrow$ *terminals*;
- $V(T) \setminus W$ with degree $= 2$ in $T \rightarrow$ *linkers*;
- $V(T) \setminus W$ with degree ≥ 3 in $T \rightarrow$ ***routers***.



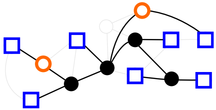
STRICT TERMINAL CONNECTION problem (S-TCP)
Input: G, $W \subseteq V(G)$ and $\ell, r \in \mathbb{Z}_0^+$
Question: Is there a strict connection tree T of G for W s.t. $|\mathrm{L}(T)| \leq \ell$ and $|\mathrm{R}(T)| \leq r$?

Dourado, M. C., Oliveira, R. A., Protti, F., and Souza, U. S.
Conexão de terminais com número restrito de roteadores e elos
Proccedings of XLVI Simpósio Brasileiro de Pesquisa Operacional, 2014, pp. 2965–2976.

Strict Terminal Connection Problem

A *strict connection tree* of G for $W \subseteq V(G)$ is a tree subgraph T of G such that $\text{leaves}(T) = W$.

- $W \rightarrow$ *terminals*;
- $V(T) \setminus W$ with degree $= 2$ in $T \rightarrow$ *linkers*;
- $V(T) \setminus W$ with degree ≥ 3 in $T \rightarrow$ ***routers***.



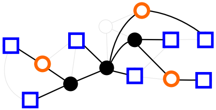
STRICT TERMINAL CONNECTION problem (S-TCP)
Input: G, $W \subseteq V(G)$ and $\ell, r \in \mathbb{Z}_0^+$
Question: Is there a strict connection tree T of G for W s.t. $|\text{L}(T)| \leq \ell$ and $|\text{R}(T)| \leq r$?

Dourado, M. C., Oliveira, R. A., Protti, F., and Souza, U. S.
Conexão de terminais com número restrito de roteadores e elos
Proccedings of XLVI Simpósio Brasileiro de Pesquisa Operacional, 2014, pp. 2965–2976.

Strict Terminal Connection Problem

A *strict connection tree* of G for $W \subseteq V(G)$ is a tree subgraph T of G such that leaves$(T) = W$.

- $W \rightarrow$ *terminals*;
- $V(T) \setminus W$ with degree $= 2$ in $T \rightarrow$ *linkers*;
- $V(T) \setminus W$ with degree ≥ 3 in $T \rightarrow$ ***routers***.



STRICT TERMINAL CONNECTION problem (S-TCP)
Input: G, $W \subseteq V(G)$ and $\ell, r \in \mathbb{Z}_0^+$
Question: Is there a strict connection tree T of G for W s.t. $|\mathrm{L}(T)| \leq \ell$ and $|\mathrm{R}(T)| \leq r$?

Dourado, M. C., Oliveira, R. A., Protti, F., and Souza, U. S.
Conexão de terminais com número restrito de roteadores e elos
Proccedings of XLVI Simpósio Brasileiro de Pesquisa Operacional, 2014, pp. 2965–2976.

Complexity of S-TCP

- Solvable in time $n^{\mathcal{O}(\ell+r)}$ (Dourado et al., 2014).
- $W[2]$-hard when parameterized by r even if $\ell \geq 0$ is constant (Melo et al., 2020).
- NP-complete even if $\ell \geq 0$ is constant and $\Delta(G) = 4$ (Melo et al., 2020).
- Solvable in time $2^{\mathcal{O}(\ell \log n)}$ when $\Delta(G) = 3$ but assuming ETH there is no $2^{o(\ell+n)}$-time algorithm even if $\Delta(G) = 3$ (Melo et al., 2020).
- FPT when parameterized by $\ell, r, \Delta(G)$ but No-poly Kernel

  (Dourado et al., 2014; Melo et al., 2020).
- Polynomial-time solvable when $r \in \{0, 1\}$ (Melo et al., 2017)

  Turing reduction to MIN-SUM st-VDP.

Complexity of S-TCP

- Solvable in time $n^{\mathcal{O}(\ell+r)}$ (Dourado et al., 2014).
- $W[2]$-hard when parameterized by r even if $\ell \geq 0$ is constant (Melo et al., 2020).
- NP-complete even if $\ell \geq 0$ is constant and $\Delta(G) = 4$ (Melo et al., 2020).
- Solvable in time $2^{\mathcal{O}(\ell \log n)}$ when $\Delta(G) = 3$ but assuming ETH there is no $2^{o(\ell+n)}$-time algorithm even if $\Delta(G) = 3$ (Melo et al., 2020).
- FPT when parameterized by $\ell, r, \Delta(G)$ but No-poly Kernel

  (Dourado et al., 2014; Melo et al., 2020).
- Polynomial-time solvable when $r \in \{0, 1\}$ (Melo et al., 2017)

  Turing reduction to MIN-SUM st-VDP.

Open problem

Is there an $n^{\mathcal{O}(r)}$-time algorithm for S-TCP?

S-TCP fixed $r \geq 2$: Combination of two problems

PROBLEM I. Connecting the terminals to the routers.

PROBLEM II. Connecting the routers to one another.

S-TCP fixed $r \geq 2$: Combination of two problems

PROBLEM I. Connecting the terminals to the routers.

- Turing reducible to MIN-SUM st-VDP.
- Polynomial-time solvable even for r arbitrarily large.

PROBLEM II. Connecting the routers to one another.

S-TCP fixed $r \geq 2$: Combination of two problems

PROBLEM I. Connecting the terminals to the routers.

- Turing reducible to MIN-SUM st-VDP.
- Polynomial-time solvable even for r arbitrarily large.

PROBLEM II. Connecting the routers to one another.

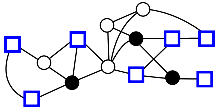
- Polynomial-time reducible to S-TCP.
- For $r \leq 3$, polynomial-time solvable by a Turing reduction to MIN-SUM st-VDP.
- For fixed $r \geq 4$, the complexity is unsettled.
- Polynomial-time reducible to SHORTEST K-CYCLE, whose complexity for fixed $|K|$ is a long-standing open question.

Some variants of S-TCP

- Constrained Router Set
- Constrained Terminal Partition
- Constrained Router Topology
- Connected Router Subgraph

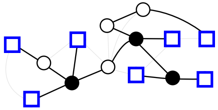
Some variants of S-TCP

- **CONSTRAINED ROUTER SET**
- CONSTRAINED TERMINAL PARTITION
- CONSTRAINED ROUTER TOPOLOGY
- CONNECTED ROUTER SUBGRAPH



Some variants of S-TCP

- **Constrained Router Set**
- Constrained Terminal Partition
- Constrained Router Topology
- Connected Router Subgraph



Some variants of S-TCP

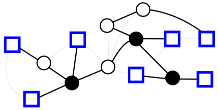
- **CONSTRAINED ROUTER SET**

  Turing reduction from S-TCP.

- CONSTRAINED TERMINAL PARTITION

- CONSTRAINED ROUTER TOPOLOGY

- CONNECTED ROUTER SUBGRAPH

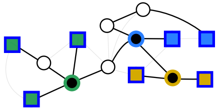


Some variants of S-TCP

- Constrained Router Set

  Turing reduction from S-TCP.

- **Constrained Terminal Partition**



- Constrained Router Topology

- Connected Router Subgraph

Some variants of S-TCP

- CONSTRAINED ROUTER SET

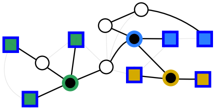
  Turing reduction from S-TCP.

- **CONSTRAINED TERMINAL PARTITION**

  NP-complete for each $r \geq 2$

  Polynomial-time reduction from $1 + d$-VDP.

- CONSTRAINED ROUTER TOPOLOGY



- CONNECTED ROUTER SUBGRAPH

Some variants of S-TCP

- CONSTRAINED ROUTER SET

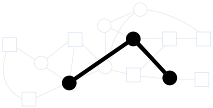
  Turing reduction from S-TCP.

- CONSTRAINED TERMINAL PARTITION

  NP-complete for each $r \geq 2$

  Polynomial-time reduction from $1 + d$-VDP.

- **CONSTRAINED ROUTER TOPOLOGY**



- CONNECTED ROUTER SUBGRAPH

Some variants of S-TCP

- Constrained Router Set

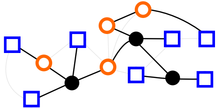
  Turing reduction from S-TCP.

- Constrained Terminal Partition

  NP-complete for each $r \geq 2$

  Polynomial-time reduction from $1 + d$-VDP.

- **Constrained Router Topology**



- Connected Router Subgraph

- CONSTRAINED ROUTER SET

  Turing reduction from S-TCP.

- CONSTRAINED TERMINAL PARTITION

  NP-complete for each $r \geq 2$

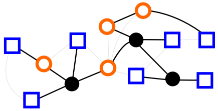
  Polynomial-time reduction from $1 + d$-VDP.

- **CONSTRAINED ROUTER TOPOLOGY**

  NP-complete for each $r \geq 3$

  Polynomial-time reduction from $1 + d$-VDP.

- CONNECTED ROUTER SUBGRAPH



Some variants of S-TCP

- CONSTRAINED ROUTER SET

  Turing reduction from S-TCP.

- CONSTRAINED TERMINAL PARTITION

  NP-complete for each $r \geq 2$

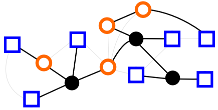
  Polynomial-time reduction from $1 + d$-VDP.

- CONSTRAINED ROUTER TOPOLOGY

  NP-complete for each $r \geq 3$

  Polynomial-time reduction from $1 + d$-VDP.

- **CONNECTED ROUTER SUBGRAPH**



Some variants of S-TCP

- Constrained Router Set

  Turing reduction from S-TCP.

- Constrained Terminal Partition

  NP-complete for each $r \geq 2$

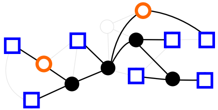
  Polynomial-time reduction from $1 + d$-VDP.

- Constrained Router Topology

  NP-complete for each $r \geq 3$

  Polynomial-time reduction from $1 + d$-VDP.

- **Connected Router Subgraph**



Some variants of S-TCP

- CONSTRAINED ROUTER SET

  Turing reduction from S-TCP.

- CONSTRAINED TERMINAL PARTITION

  NP-complete for each $r \geq 2$

  Polynomial-time reduction from $1 + d$-VDP.

- CONSTRAINED ROUTER TOPOLOGY

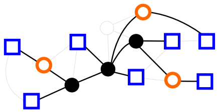
  NP-complete for each $r \geq 3$

  Polynomial-time reduction from $1 + d$-VDP.

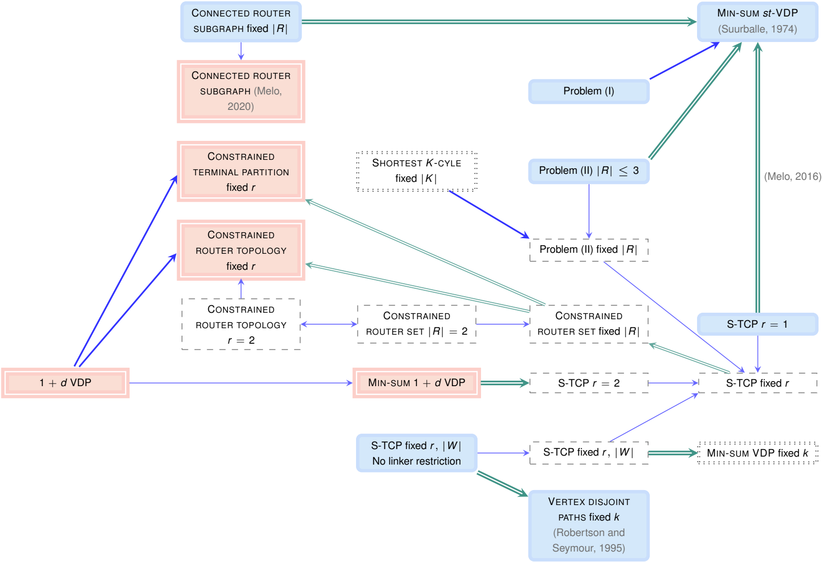
- **CONNECTED ROUTER SUBGRAPH**

  Polynomial-time solvable for constant $r \geq 2$

  Turing reduction from MIN-SUM st-VDP.



Relationship: Disjoint paths and S-TCP



Open problems

- Are $1 + d$-EDGE-DISJOINT PATHS and $1 + d$-VERTEX-DISJOINT PATHS on planar graphs polynomial-time solvable?

Naves, G. and Sebő, A.
Multiflow feasibility: An annotated tableau
Research Trends in Combinatorial Optimization, 2009, 261–283.

Open problems

- Are $1+d$-EDGE-DISJOINT PATHS and $1+d$-VERTEX-DISJOINT PATHS on planar graphs polynomial-time solvable?

  Naves, G. and Sebő, A.
  Multiflow feasibility: An annotated tableau
  Research Trends in Combinatorial Optimization, 2009, 261–283.

- Is MIN-SUM VERTEX-DISJOINT PATHS polynomial-time solvable for fixed $k \geq 2$?

Open problems

- Are $1 + d$-EDGE-DISJOINT PATHS and $1 + d$-VERTEX-DISJOINT PATHS on planar graphs polynomial-time solvable?

  Naves, G. and Sebő, A.
  Multiflow feasibility: An annotated tableau
  Research Trends in Combinatorial Optimization, 2009, 261–283.

- Is MIN-SUM VERTEX-DISJOINT PATHS polynomial-time solvable for fixed $k \geq 2$?
- Is S-TCP parameterized by r in XP?
- Is S-TCP parameterized by $|W|$ in FPT (or in XP)?

Thank you for your attention!

(https://doi.org/10.1002/net.21976)

Alexsander A. de Melo
aamelo@cos.ufrj.br