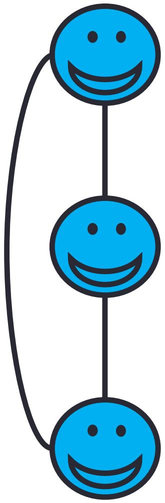
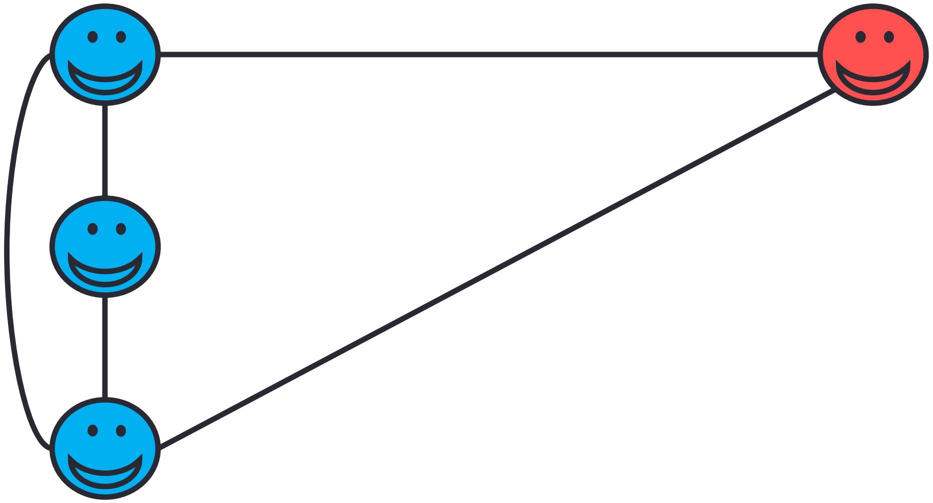
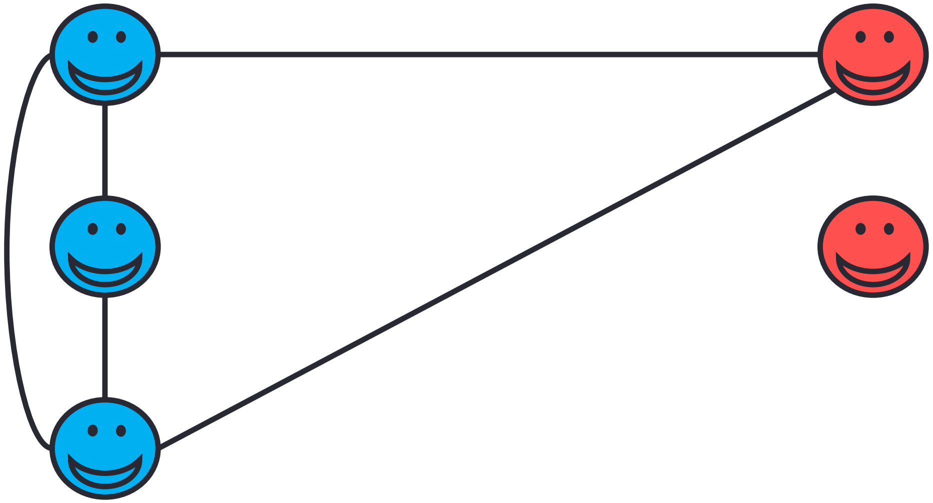
EDITING GRAPHS TO SATISFY DIVERSITY REQUIREMENTS

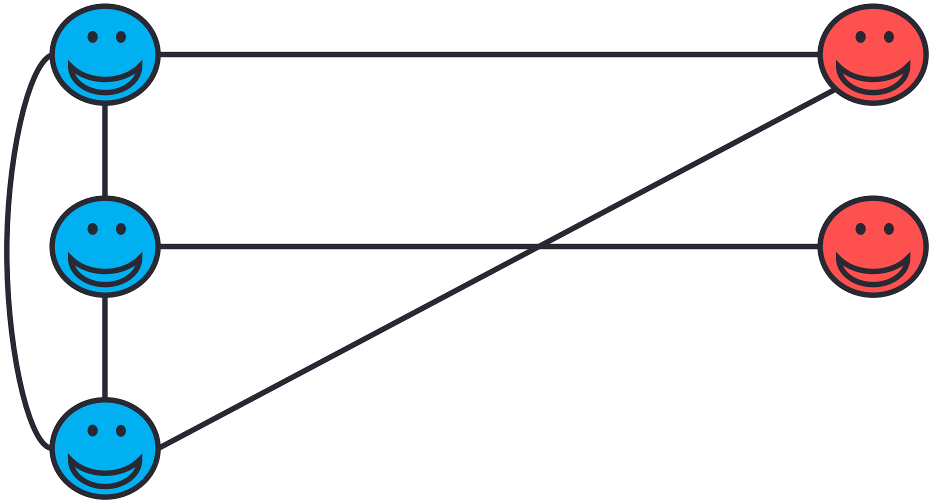
Huda Chuangpishit **Manuel Lafond** **Lata Narayanan**

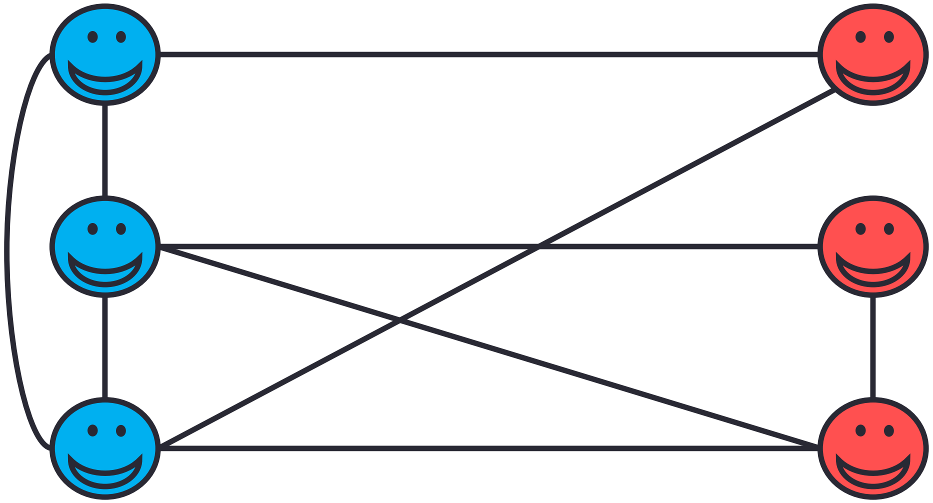
Ryerson University
Université de Sherbrooke
Concordia University



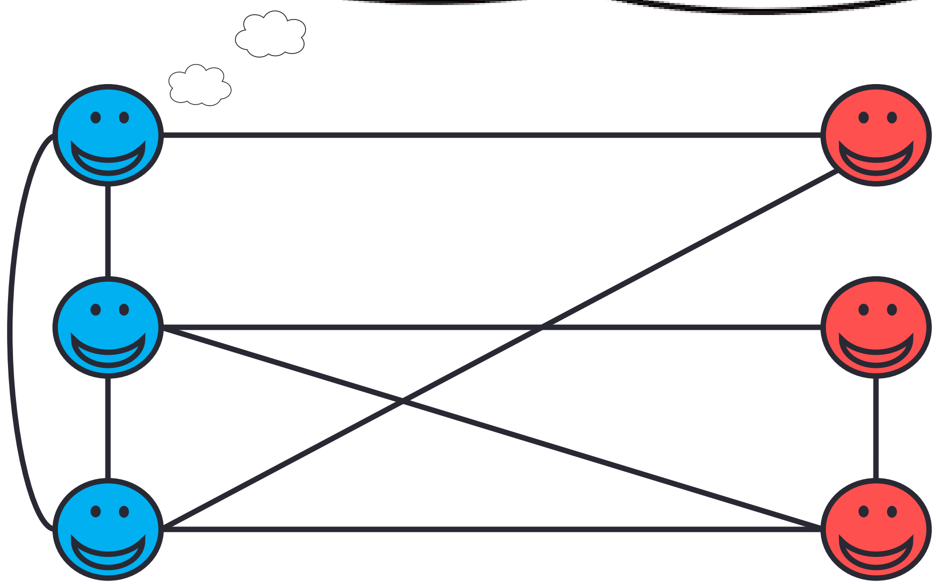




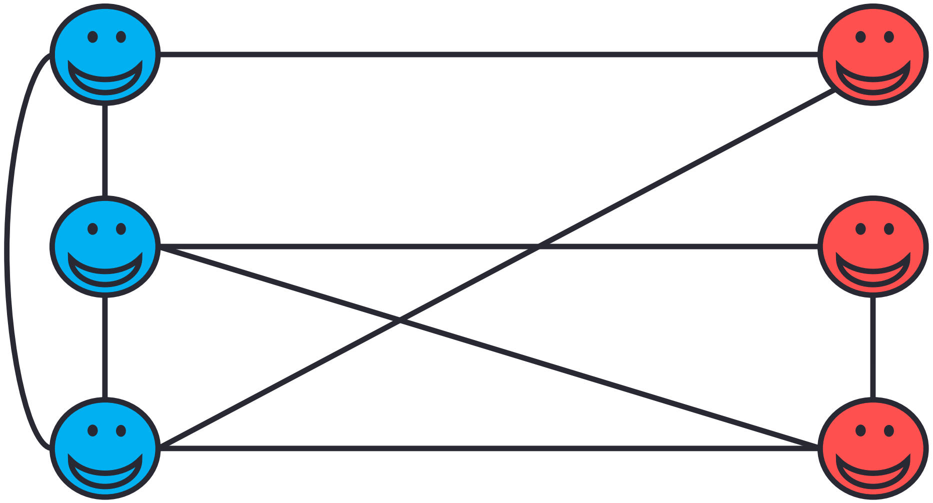




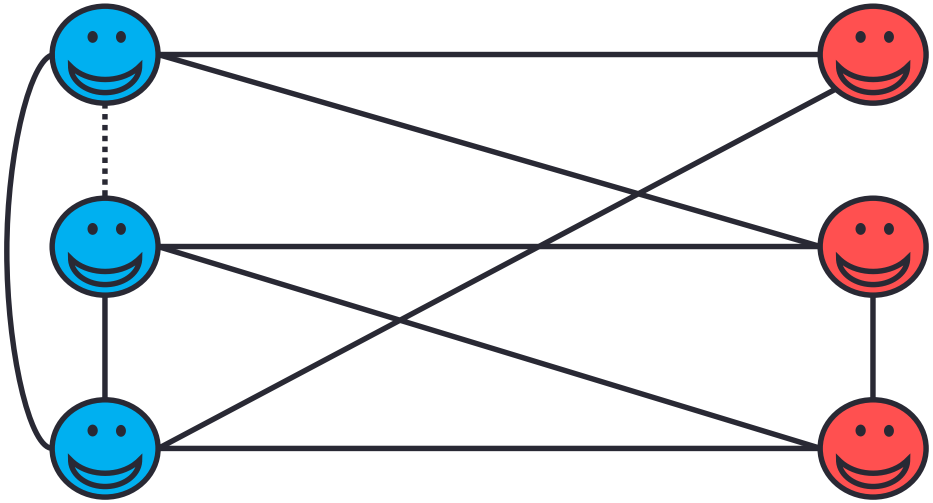
I’d really like to have **2** red friends!
I could learn from them.
But I only have time to maintain **3** friendshisps.



Wants:
≥ 2 red friends
≤ 3 friends total



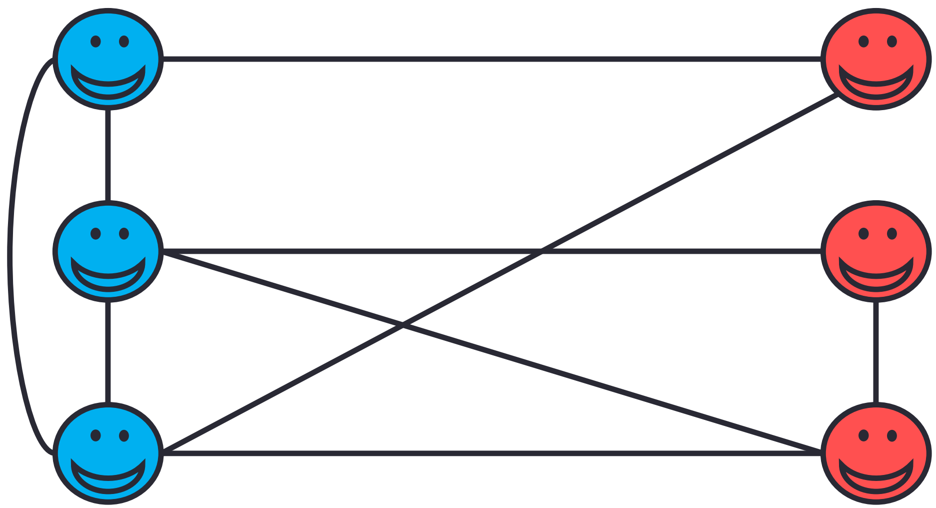
Wants:
≥ 2 red friends
≤ 3 friends total



≥ 2 red friends
≤ 3 friends total

≥ 1 red friend
≤ 3 friends total

≥ 2 red friends
≤ 3 friends total



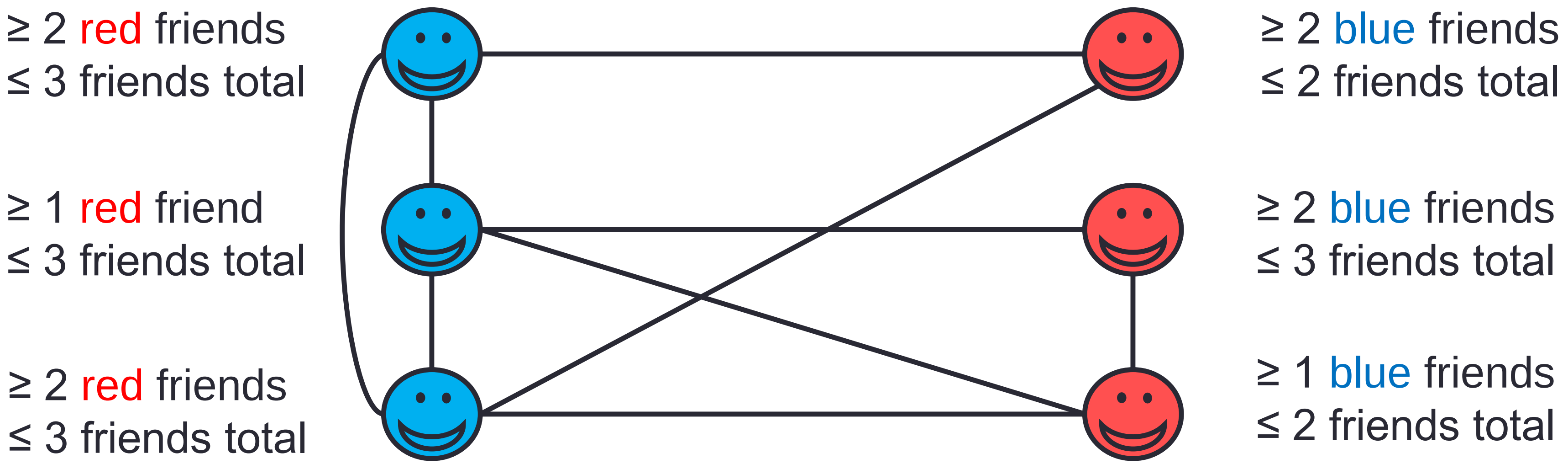
≥ 2 blue friends
≤ 2 friends total

≥ 2 blue friends
≤ 3 friends total

≥ 1 blue friends
≤ 2 friends total

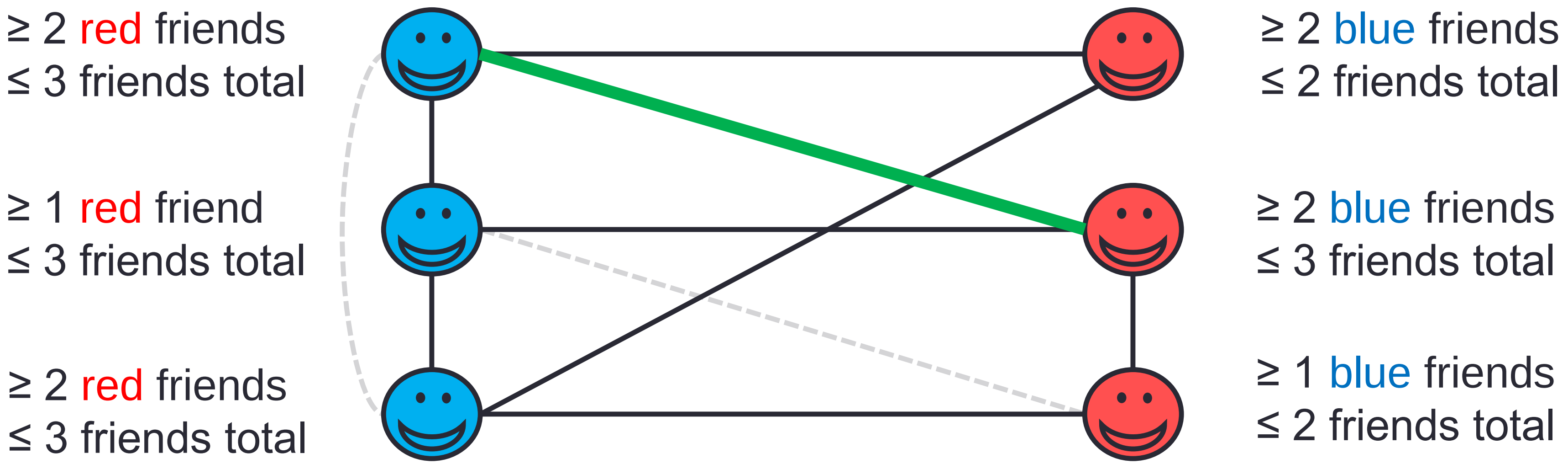
MIN-EDIT-COST PROBLEM

Modify a minimum number of edges so that everyone is satisfied.



MIN-EDIT-COST PROBLEM

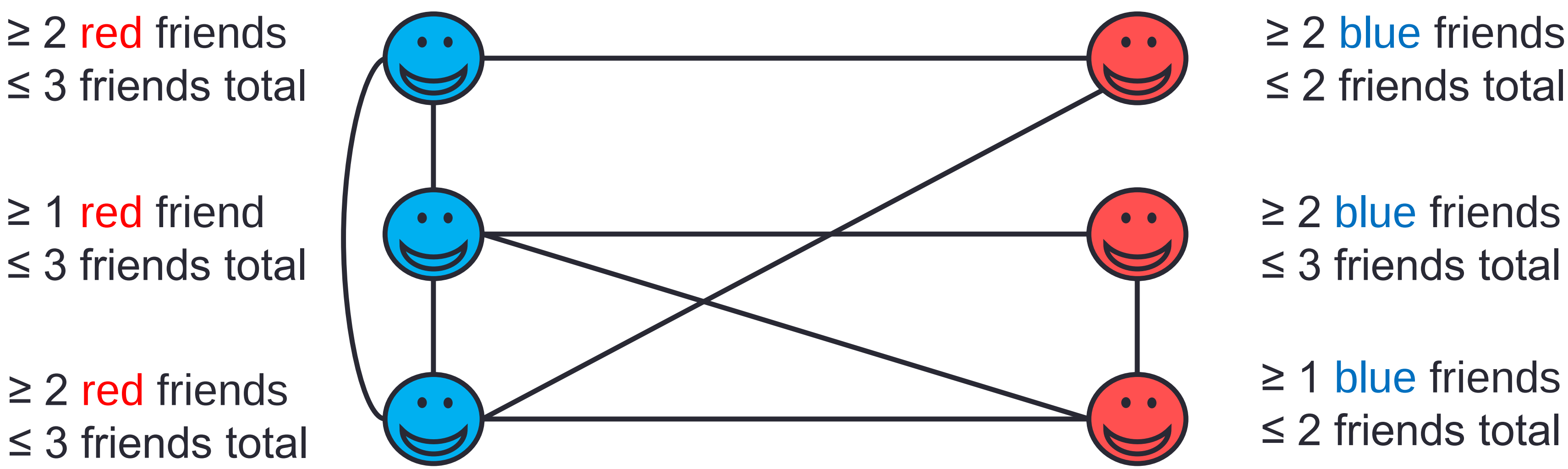
Modify a minimum number of edges so that everyone is satisfied.



1 edge insertion + 2 edge deletions

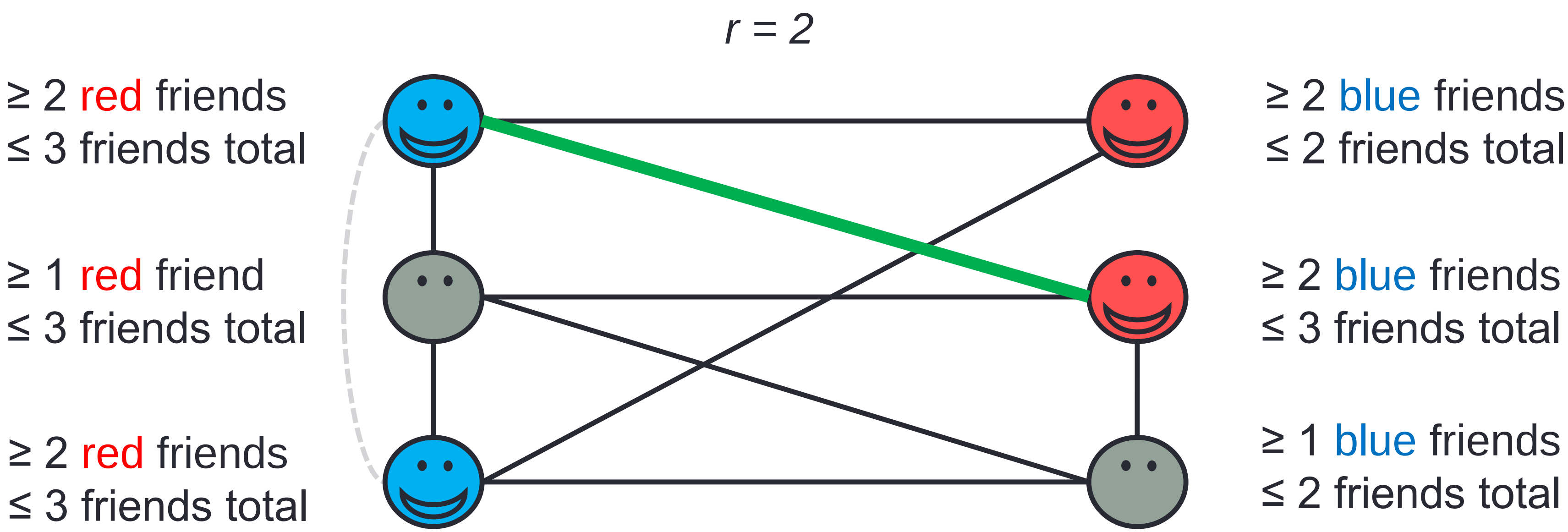
MAX-SATISFIED-NODES PROBLEM

Edit ***r*** edges to maximize the number of satisfied people (*r* is given).



MAX-SATISFIED-NODES PROBLEM

Edit ***r*** edges to maximize the number of satisfied people (*r* is given).



Motivation: make multicultural workgroups

More formally

- Given
  - A graph $G = (V, E)$ and a coloring function $c : V \rightarrow [k]$
  - For each $v \in V$ and each $i \in [k]$, a degree lower bond $d_i(v)$
  - For each $v \in V$, a total degree upper bound $\beta(v)$
- A node v is ***satisfied*** if
  - for each $i \in [k]$, v has at least $d_i(v)$ neighbors of color i
  - v has at most $\beta(v)$ neighbors

More formally

- Given
  - A graph $G = (V, E)$ and a coloring function $c : V \rightarrow [k]$
  - For each $v \in V$ and each $i \in [k]$, a degree lower bond $d_i(v)$
  - For each $v \in V$, a total degree upper bound $\beta(v)$
- A node *v* is ***satisfied*** if
  - for each $i \in [k]$, *v* has at least $d_i(v)$ neighbors of color *i*
  - *v* has at most $\beta(v)$ neighbors

- **MIN-EDIT-COST**: insert/remove a minimum number of edges so that every $v \in V$ is satisfied.
- **MAX-SATISFIED-NODES**: insert/remove at most *r* edges to satisfy a maximum number of people.

Related work

- Many graph editing problems: minimum modifications to …
  - **belong to graph class**
    - e.g. bipartite [Yannakakis, 1981], cograph [Liu & al., 2011], outerplanar, …
  - obtain a **regular** graph [Cornuéjols, 1988] (polynomial-time!)
    - Or kind-of regular (anonymization) [Liu & Terzi, 2008]
  - obtain a **specific graph** (graph edit distance) [Neuhaus-Bunke, 2005]
  - obtain a **given degree sequence** [Golovach & Mertzios, 2012]
  - …

Related work

- To make each vertex v of degree of required degree $r(v)$

Related work

- To make each vertex v of degree of required degree $r(v)$
  - Polynomial time [Mathieson & Szeider, 2012]

Related work

- To make each vertex *v* of degree of required degree *r(v)*
  - Polynomial time [Mathieson & Szeider, 2012]
  - Variant: possibles degrees for v is a set *R(v)* of integers.
  - NP-hard in general.
  - In P if only deletions allowed and *R(v)* is an interval (cf [Korte & Vygen, 2008]).

Related work

- To make each vertex *v* of degree of required degree *r(v)*
  - Polynomial time [Mathieson & Szeider, 2012]
  - Variant: possibles degrees for v is a set *R(v)* of integers.
  - NP-hard in general.
  - In P if only deletions allowed and *R(v)* is an interval (cf [Korte & Vygen, 2008]).
  - Insertions + deletions + *R(v)* is an interval = UNKNOWN

Related work

- To make each vertex *v* of degree of required degree *r(v)*
  - Polynomial time [Mathieson & Szeider, 2012]
  - Variant: possibles degrees for v is a set *R(v)* of integers.
  - NP-hard in general.
  - In P if only deletions allowed and *R(v)* is an interval (cf [Korte & Vygen, 2008]).
  - Insertions + deletions + *R(v)* is an interval = **polytime**

Related work

- To make each vertex *v* of degree of required degree *r(v)*
  - Polynomial time [Mathieson & Szeider, 2012]
  - Variant: possibles degrees for v is a set *R(v)* of integers.
  - NP-hard in general.
  - In P if only deletions allowed and *R(v)* is an interval (cf [Korte & Vygen, 2008]).
  - Insertions + deletions + *R(v)* is an interval = **polytime**
- Degree editing problems on colored graphs = ???

In the paper

- **MIN-EDIT-COST**: insert/remove a minimum number of edges so that every $v \in V$ is satisfied.
  - Can be solved in time $O(n^5 \log n)$
  - Two colors case in time $O(n^3 \log n \log \log n)$

In the paper

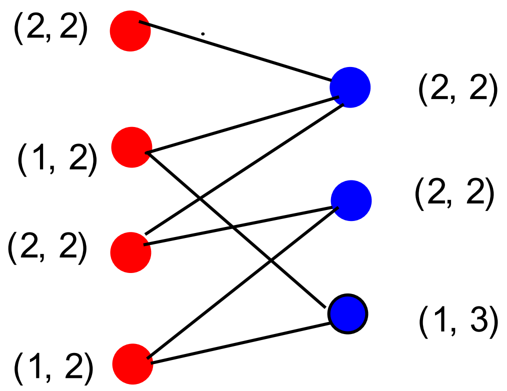
- **MIN-EDIT-COST**: insert/remove a minimum number of edges so that every $v \in V$ is satisfied.
  - Can be solved in time $O(n^5 \log n)$
  - Two colors case in time $O(n^3 \log n \log \log n)$

- **MAX-SATISFIED-NODES**: insert/remove at most *r* edges to satisfy a maximum number of people.
  - W[1]-hard for parameter $r + l$ (l = number of people to satisfy)
  - ½ approximation with no degree upper bounds
  - 1/(9k) approximation with degree upper bounds

Min-Edit-Cost (bipartite case)

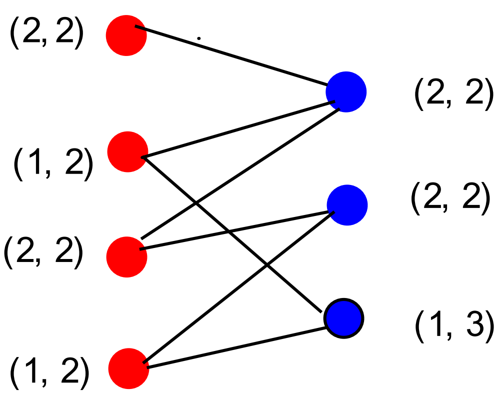
- Reduction to Min-Cost Flow
- Given: a directed graph with source/sink *s* and *t*, and in which each arc *e* has
  - A cost c_e
  - A capacity u_e
  - A lower bound l_e
- Find: a weight w_e assignment on arcs s.t.
  - each vertex has total weight-in = total weight-out (a flow)
  - $l_e \leq w_e \leq u_e$
  - The sum of costs $c_e w_e$ is minimized

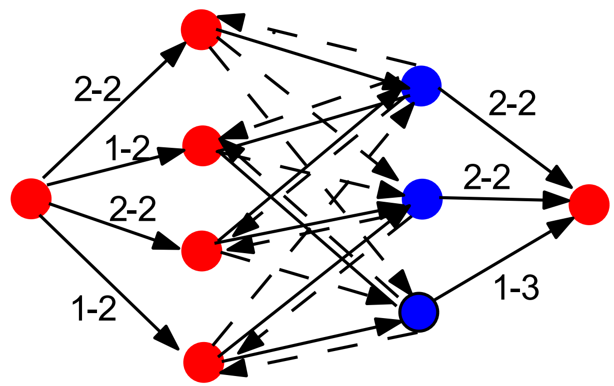
Min-Edit-Cost (bipartite case)



Our instance
(a,b) means
(want *a* more,
max-degree *b*)

Min-Edit-Cost (bipartite case)

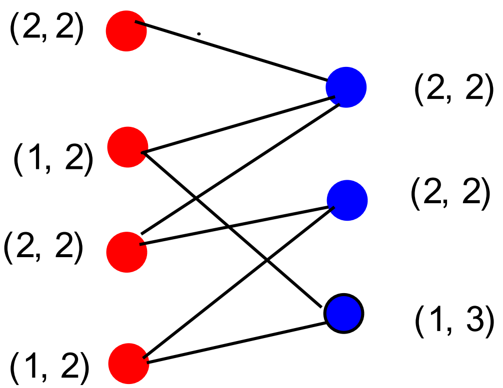




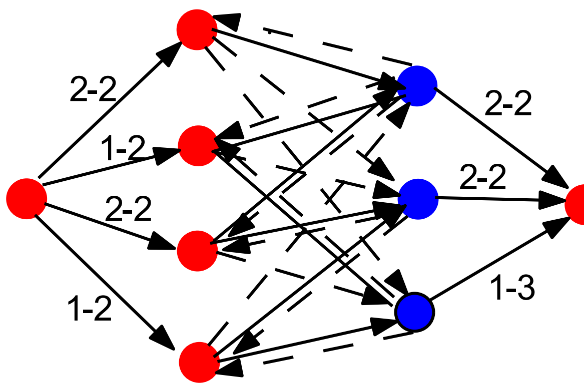
Our instance
(a,b) means
(want *a* more,
max-degree *b*)

- Reduction to Min-Cost Flow
- *a-b* means $l_e = a$, $u_e = b$
- Solid middle edges have
  $l_e = u_e = 1$ and $c_e = 0$
- Dashed edges have
  $l_e = 0$, $u_e = 1$ and $c_e = 1$

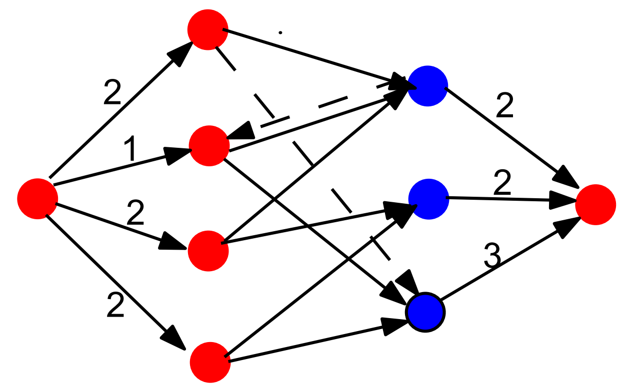
Min-Edit-Cost (bipartite case)



Our instance
(a,b) means
(want *a* more,
max-degree *b*)



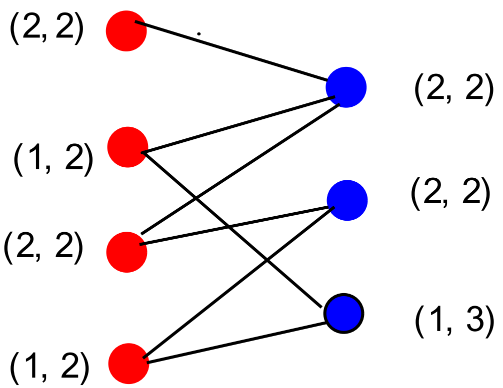
- Reduction to Min-Cost Flow
- *a-b* means $l_e = a$, $u_e = b$
- Solid middle edges have $l_e = u_e = 1$ and $c_e = 0$
- Dashed edges have $l_e = 0$, $u_e = 1$ and $c_e = 1$



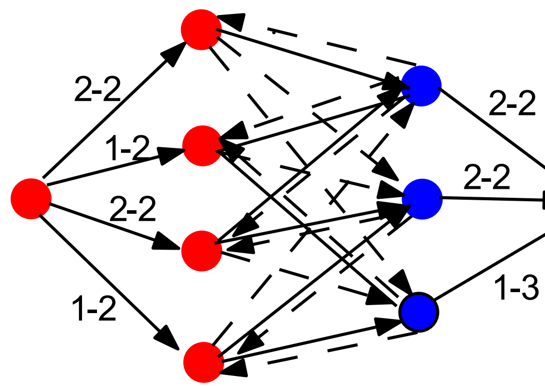
Solution of cost 2.
Using a dashed backwards edge = deleting the edge
Using a dashed forward edge = inserting the edge

Min-Edit-Cost (bipartite case)

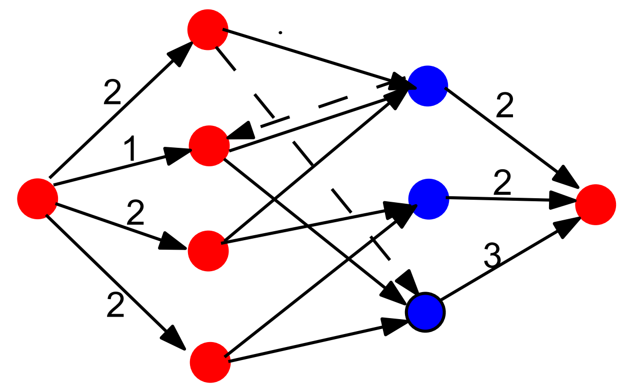
Takes time $O(n^3 \log n \log \log n)$ *to solve [Ahuja & al, 1992]*



Our instance
(a,b) means
(want *a* more,
max-degree *b*)



- Reduction to Min-Cost Flow
- *a-b* means $l_e = a$, $u_e = b$
- Solid middle edges have $l_e = u_e = 1$ and $c_e = 0$
- Dashed edges have $l_e = 0$, $u_e = 1$ and $c_e = 1$



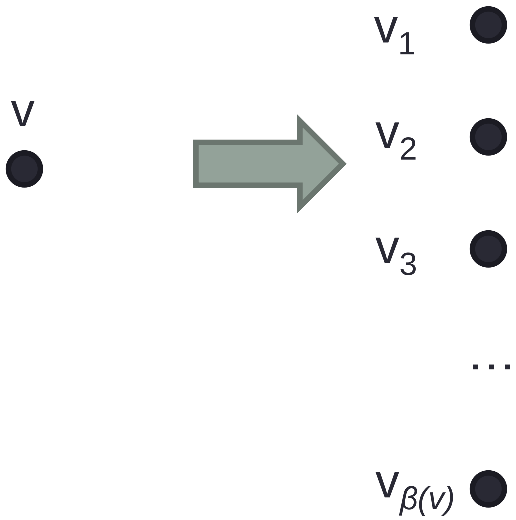
Solution of cost 2.
Using a dashed backwards edge = deleting the edge
Using a dashed forward edge = inserting the edge

Min-Edit-Cost (general case)

- **Idea**: weighted perfect matching reduction
- Transform *G* into *H* such that *G* has **edit cost** *c* iff *H* has a **perfect matching** of cost *c.* (weights of H edges are 0-1)

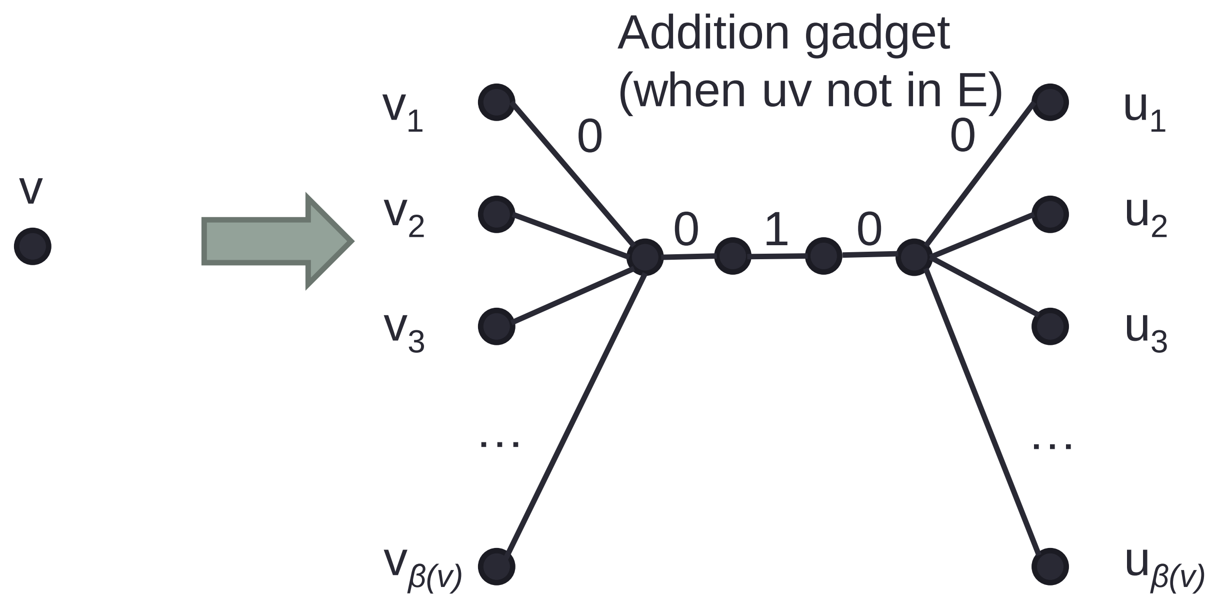
Min-Edit-Cost (general case)

- Each vertex *v* has at most *β(v)* neighbors.
- Make *β(v)* copies of *v*, each representing a potential neighbor in a solution.



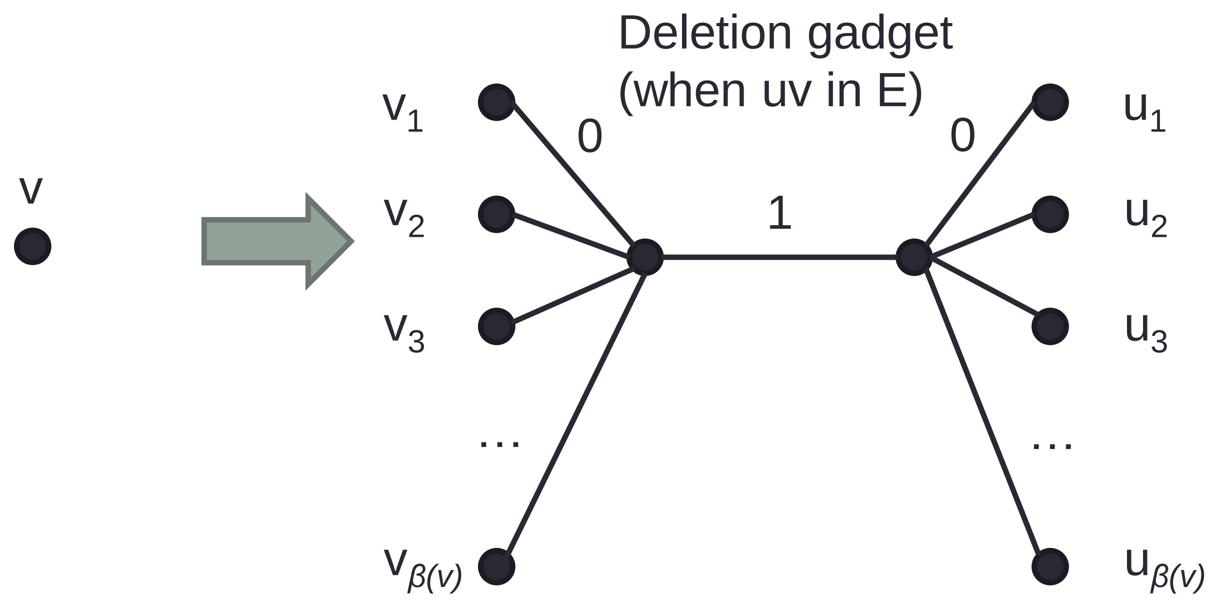
Min-Edit-Cost (general case)

- Each vertex *v* has at most *β(v)* neighbors.
- Make *β(v)* copies of *v*, each representing a potential neighbor in a solution.



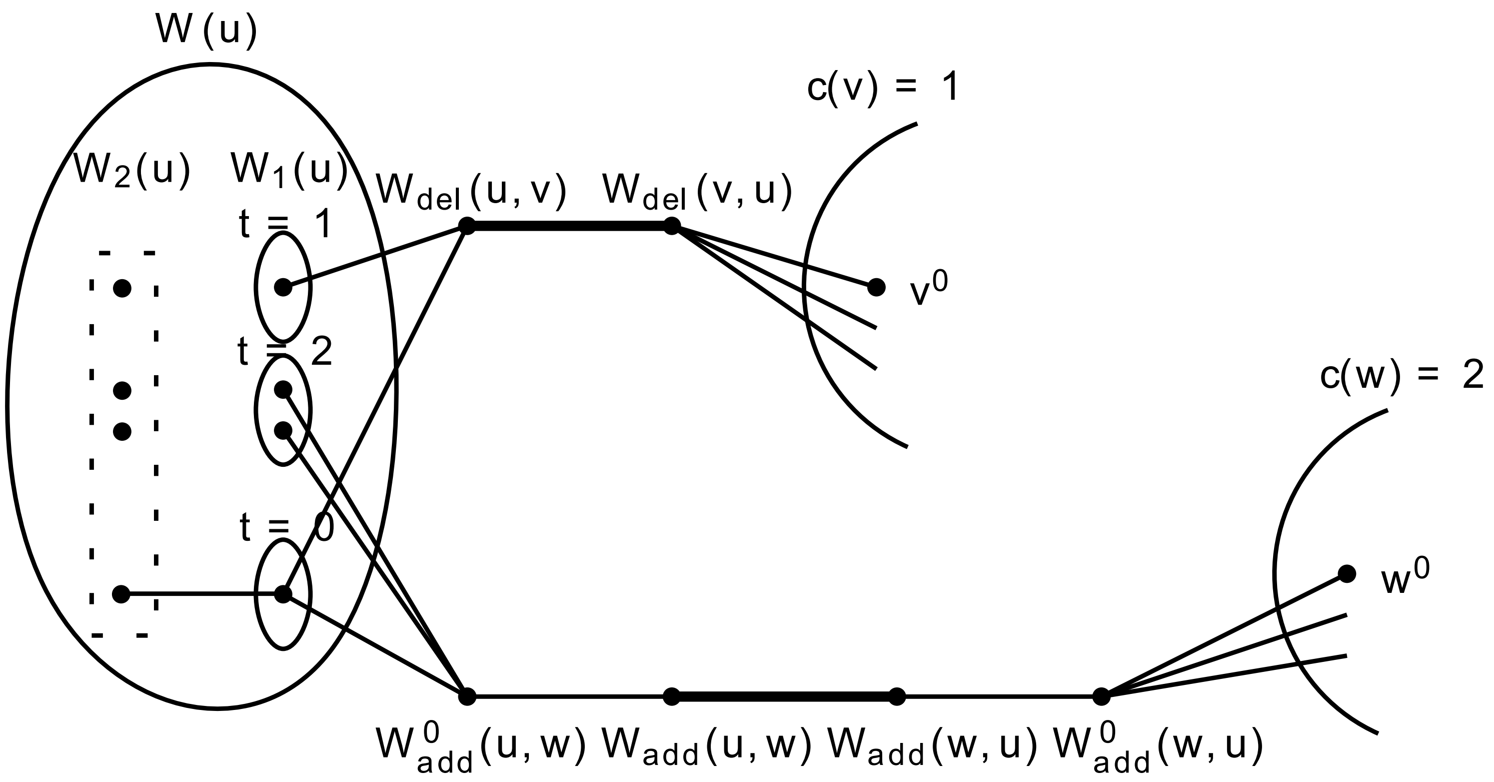
Min-Edit-Cost (general case)

- Each vertex *v* has at most *β(v)* neighbors.
- Make *β(v)* copies of *v*, each representing a potential neighbor in a solution.



Min-Edit-Cost (general case)

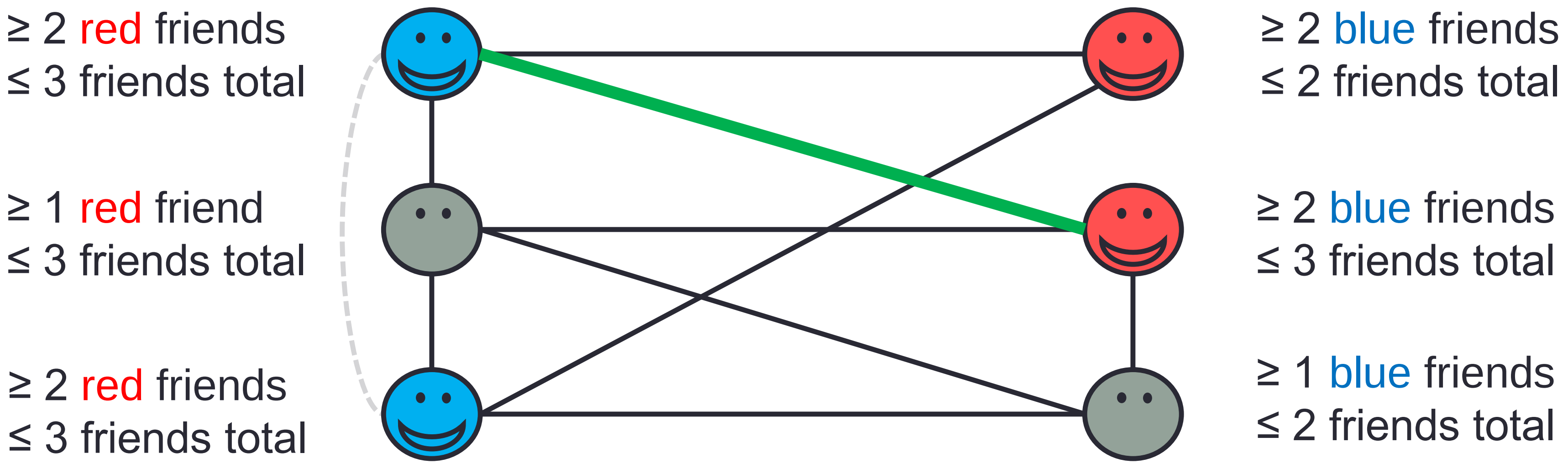
- Add gadgets between relevant colors.



MAX-SATISFIED-NODES PROBLEM

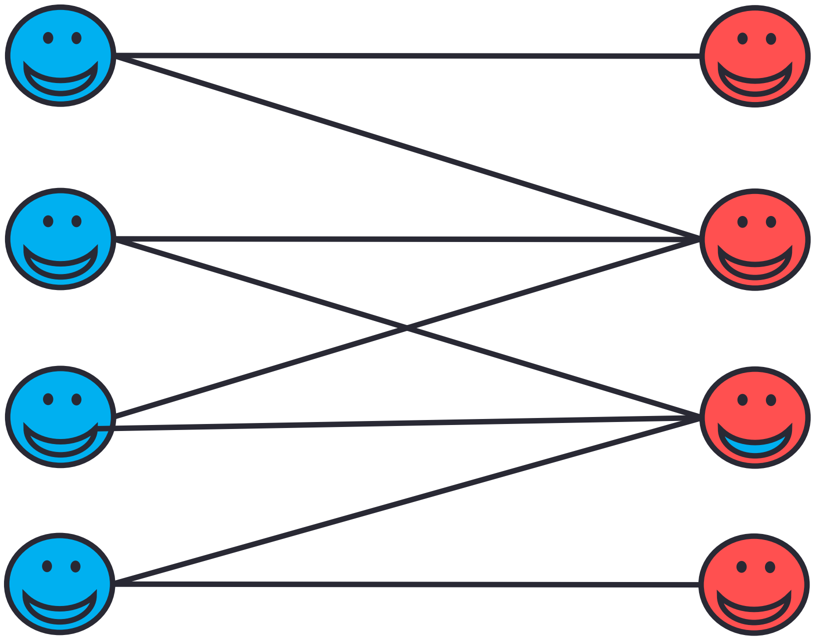
Edit ***r*** edges to maximize the number of satisfied people (*r* is given).

r = 2



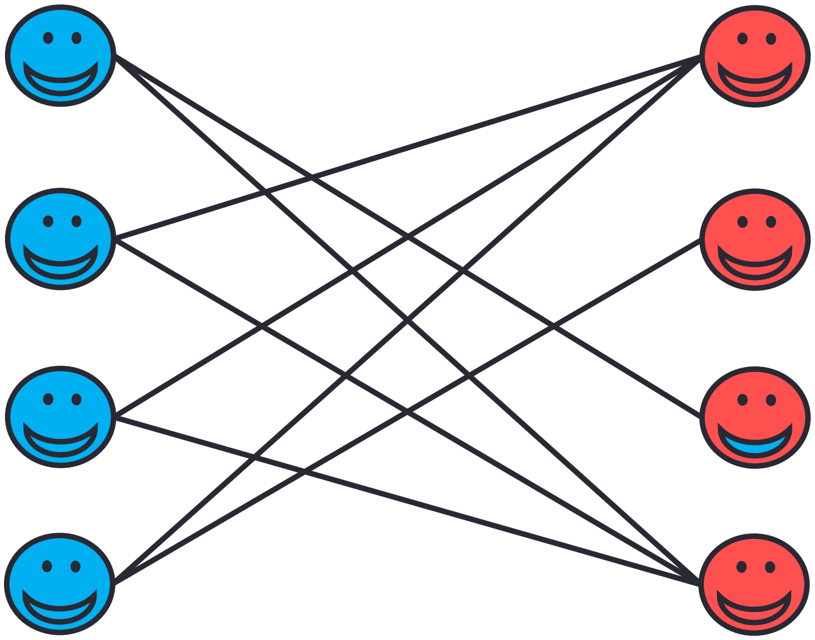
MAX-SATISFIED-NODES

- W[1]-hardness from Balanced BiClique [Lin, SODA2015]
  - Given bipartite graph *G = (A U B, E)* and integer *q*, is there a complete bipartite graph with *q* nodes on each side.



MAX-SATISFIED-NODES

- W[1]-hardness from Balanced BiClique [Lin, SODA2015]
  - Given bipartite graph *G = (A U B, E)* and integer *q*, is there a complete bipartite graph with *q* nodes on each side.



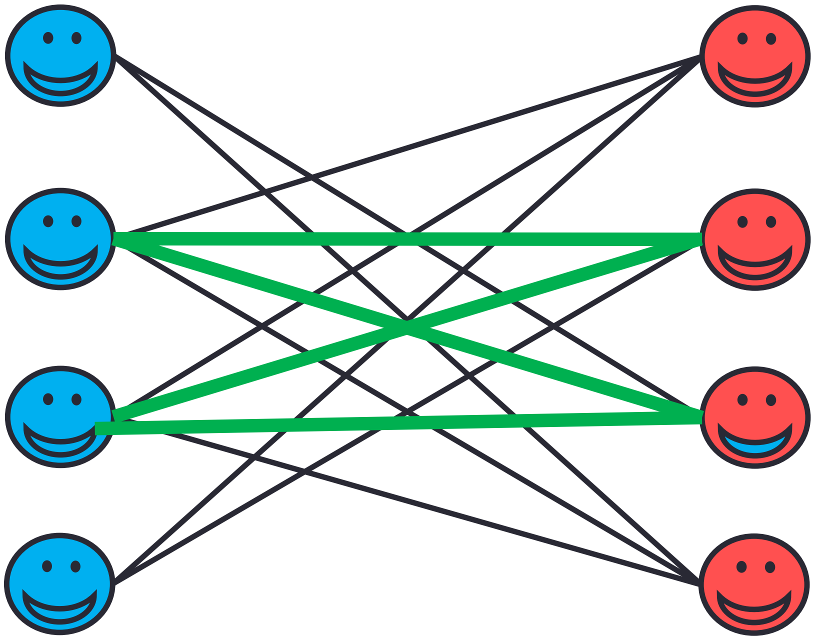
Take complement

Make each node want *q* more neighbors of other color.

Can we add q^2 edges and satisfy $2q$ guys?

MAX-SATISFIED-NODES

- W[1]-hardness from Balanced BiClique [Lin, SODA2015]
  - Given bipartite graph *G = (A U B, E)* and integer *q*, is there a complete bipartite graph with *q* nodes on each side.



Take complement

Make each node want *q* more neighbors of other color.

Can we add q^2 edges and satisfy *2q* guys?

YES iff G has a *2q-biclique.*

½ approx with no degree upper bounds

½ approx with no degree upper bounds

- *req(v)* = # of edges to edit if we **only** want to satisfy *v*
  - Easy to compute
- Node *v* is satisfied iff *req(v) = 0*

½ approx with no degree upper bounds

- $req(v)$ = # of edges to edit if we **only** want to satisfy v
  - Easy to compute
- Node v is satisfied iff $req(v) = 0$

- Order $V(G) = \{v_1, v_2, \ldots, v_n\}$ s.t. $req(v_i) \leq req(v_{i+1})$

- Editing edge $v_i v_j$ can, at best, lower $req(v_i)$ and $req(v_j)$ by 1

- We are allowed to edit at most r edges =>

  If $\sum_{i=1}^{p+1} req(v_i) > 2r$ then we can't satisfy more than p nodes.

½ approx with no degree upper bounds

- If $\sum_{i=1}^{p+1} req(v_i) > 2r$ then we can't satisfy more than *p* nodes.
- Choose smallest *p* that verifies the above inequality.

- With no degree upper bounds, We are always able to satisfy the nodes $v_1, v_2, \ldots, v_{p/2}$
  - Just add $req(v_1)$ neighbors to v_1 of the appropriate colors until satisfaction.
  - Repeat with $v_2, \ldots, v_{p/2}$
  - Requires at most $\sum_{i=1}^{p/2} req(v_i) \leq r$ modifications.

½ approx *with* degree upper bounds

- If $\sum_{i=1}^{p+1} req(v_i) > 2r$ then we can't satisfy more than *p* nodes.
- Choose smallest *p* that verifies the above inequality.

- With no degree upper bounds, We are always able to satisfy the nodes $v_1, v_2, \ldots, v_{p/2}$
  - Just add $req(v_1)$ neighbors to v_1 of the appropriate colors until satisfaction.
  - Repeat with $v_2, \ldots, v_{p/2}$
  - Requires at most $\sum_{i=1}^{p/2} req(v_i) \leq r$ modifications.

- Doesn't work if we have upper bounds on degrees.

½ approx *with* degree upper bounds

- If $\sum_{i=1}^{p+1} req(v_i) > 2r$ then we can't satisfy more than *p* nodes.
- Choose smallest *p* that verifies the above inequality.

- If we have upper bounds, we'll only care about satisfying vertices in a single color class.
  - Choose color class containing the most vertices from $v_1, \ldots, v_p$
  - Satisfy ¼ of them (see paper for details)
  - Yields a *1/floor(8k) ~ 1/(9k)* approx

Some perspectives

- Better approximation? 1/(9k) is probably not best possible
- Other parameters for FPT algorithms?
  - e.g. # of *un*satisfied nodes, or structural graph parameters
- Better algorithms for MinEditCost.
- Other nice **colored** graph editing problems?

Some perspectives

- Better approximation? 1/(9k) is probably not best possible
- Other parameters for FPT algorithms?
  - e.g. # of *un*satisfied nodes, or structural graph parameters
- Better algorithms for MinEditCost.
- Other nice **colored** graph editing problems?
- That's it, thanks!