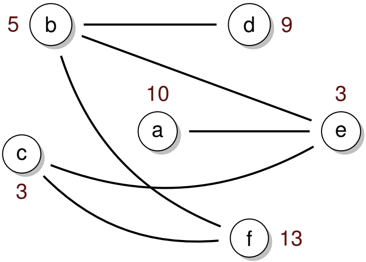
Conflict Directed Clause Learning for the Maximum Weighted Clique Problem

Emmanuel Hebrard[1] **and George Katsirelos**[2]

[1]LAAS-CNRS, Université de Toulouse [2]MIAT, INRA

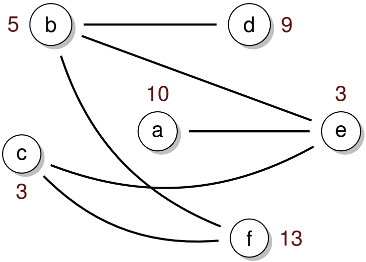
The Maximum Weight Clique Problem

- Given a graph *G* with weight function *w*



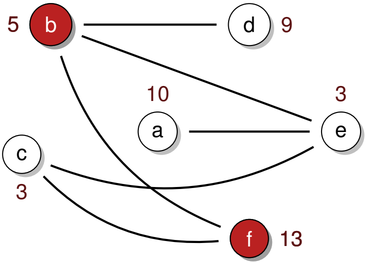
The Maximum Weight Clique Problem

- Given a graph G with weight function w
- Find a clique – set of vertices all pairwise adjacent – of maximum weight



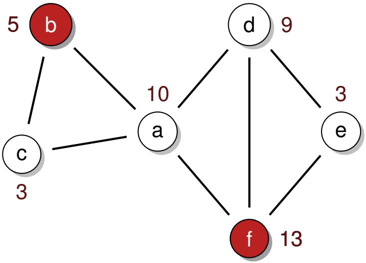
The Maximum Weight Clique Problem

- Given a graph G with weight function w
- Find a clique – set of vertices all pairwise adjacent – of maximum weight



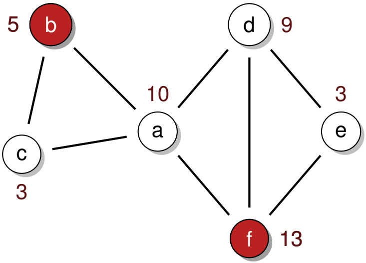
The Maximum Weight Clique Problem

- Given a graph *G* with weight function *w*
- Find a clique – set of vertices all pairwise adjacent – of maximum weight
- $\Longleftrightarrow$ Find a set of vertices without edge (Independent Set – *IS*) of maximum weight (on the complement)



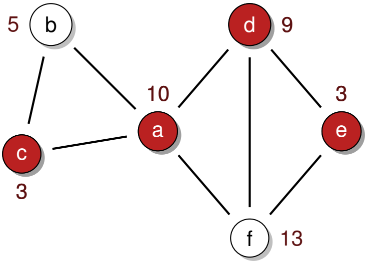
The Maximum Weight Clique Problem

- Given a graph G with weight function w
- Find a clique – set of vertices all pairwise adjacent – of maximum weight
- $\iff$ Find a set of vertices without edge (Independent Set – *IS*) of maximum weight (on the complement)
- $\iff$ Find a set of vertices touching every edge (Vertex Cover – *VC*) minimum weight (on the complement)



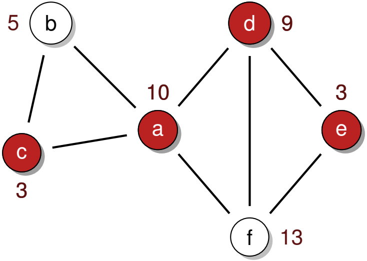
The Maximum Weight Clique Problem

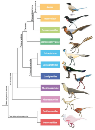
- Given a graph *G* with weight function *w*
- Find a clique – set of vertices all pairwise adjacent – of maximum weight
- $\iff$ Find a set of vertices without edge (Independent Set – *IS*) of maximum weight (on the complement)
- $\iff$ Find a set of vertices touching every edge (Vertex Cover – *VC*) minimum weight (on the complement)



The Maximum Weight Clique Problem

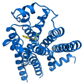
- Given a graph *G* with weight function *w*
- Find a clique – set of vertices all pairwise adjacent – of maximum weight
- $\iff$ Find a set of vertices without edge (Independent Set – *IS*) of maximum weight (on the complement)
- $\iff$ Find a set of vertices touching every edge (Vertex Cover – *VC*) minimum weight (on the complement)





Perfect philogeny

Error-correcting codes



Subgraph isomorphism

Branch & Bound (CP) vs SAT Encoding

Branch & Bound (CP)

SAT Encoding

Branch & Bound (CP) vs SAT Encoding

Branch & Bound (CP)

- Dedicated upper bound (coloring)

SAT Encoding

- Weak upper bound (UP)

Branch & Bound (CP) vs SAT Encoding

Branch & Bound (CP)

- Dedicated upper bound (coloring)
- Dedicated data structure (bitsets)

SAT Encoding

- Weak upper bound (UP)
- No data structure

Branch & Bound (CP) vs SAT Encoding

Branch & Bound (CP)	SAT Encoding
Dedicated upper bound (coloring)	Weak upper bound (UP)
Dedicated data structure (bitsets)	No data structure
Dedicated strategies (degree, color)	VSIDS

Branch & Bound (CP) vs SAT Encoding

Branch & Bound (CP)

- Dedicated upper bound (coloring)
- Dedicated data structure (bitsets)
- Dedicated strategies (degree, color)
- Backtracking

SAT Encoding

- Weak upper bound (UP)
- No data structure
- VSIDS
- Conflict-Driven Clause Learning

Branch & Bound (CP) vs SAT Encoding

Branch & Bound (CP)

- Dedicated upper bound (coloring)
- Dedicated data structure (bitsets)
- Dedicated strategies (degree, color)
- Backtracking

SAT Encoding

- Weak upper bound (UP)
- No data structure
- VSIDS
- Conflict-Driven Clause Learning

We want the best of both worlds!

Generalised Nogoods [Katsirelos and Bacchus 05][Ohrimenko, Stuckey and Codish 07]

- Branch and propagate as a CP solver

Generalised Nogoods [Katsirelos and Bacchus 05][Ohrimenko, Stuckey and Codish 07]

- Branch and propagate as a CP solver
- Store every deduction made during propagation as an explanation clause

$$(p_1 \land p_2 \land \ldots \land p_k) \implies c$$

with p_i's and c literals of the form $x\{=, \neq\}v$

Generalised Nogoods [Katsirelos and Bacchus 05][Ohrimenko, Stuckey and Codish 07]

- Branch and propagate as a CP solver
- Store every deduction made during propagation as an explanation clause

$$(p_1 \land p_2 \land \ldots \land p_k) \implies c$$

with p_i's and c literals of the form $x\{=, \neq\}v$

- When failing, compute a conflict using the explanation graph

Generalised Nogoods [Katsirelos and Bacchus 05][Ohrimenko, Stuckey and Codish 07]

- Branch and propagate as a CP solver
- Store every deduction made during propagation as an explanation clause

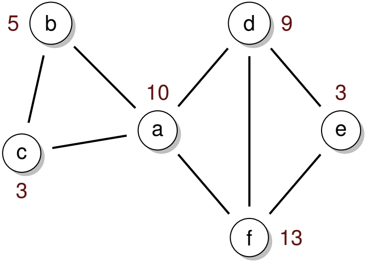
$$(p_1 \wedge p_2 \wedge \ldots \wedge p_k) \implies c$$

  with p_i's and c literals of the form $x\{=, \neq\}v$

- When failing, compute a conflict using the explanation graph
- A global constraint "X is an independent set of weight larger than k"
  - Compute an upper bound
  - Prune w.r.t. this upper bound
  - Need to compute explanations!

Branching

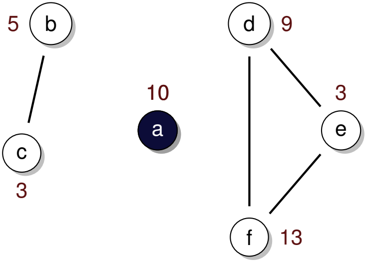
For any vertex v:



Branching

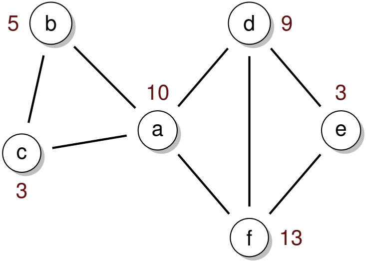
For any vertex v:

- **either** v is in VC ($x_v = \texttt{true}$)



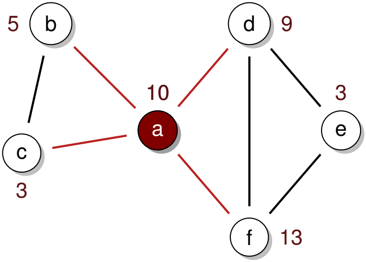
For any vertex v:

- **either** v is in VC ($x_v = \texttt{true}$)



For any vertex v:

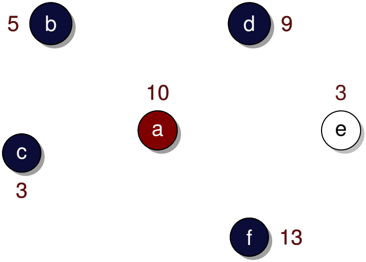
- **either** v is in VC ($x_v = \texttt{true}$)
- **or** v is in IS ($x_v = \texttt{false}$)



Branching

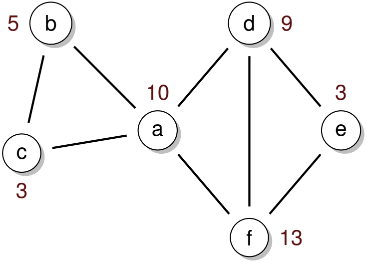
For any vertex v:

- **either** v is in VC ($x_v = \texttt{true}$)
- **or** v is in IS ($x_v = \texttt{false}$)



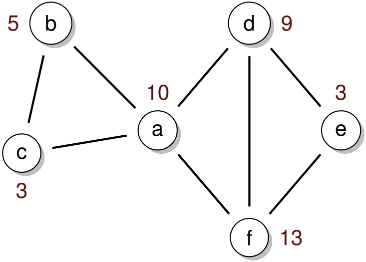
Upper bound

- Unweighted case: *coloring*



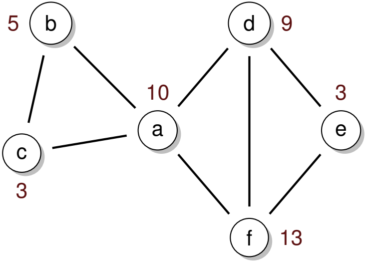
Upper bound

- Unweighted case: *coloring*
- A clique of size k requires k colors



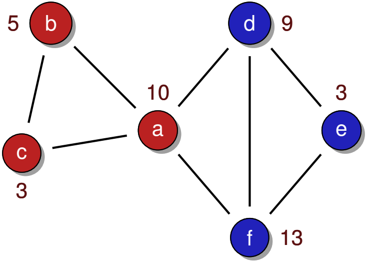
Upper bound

- Unweighted case: *coloring*
- A clique of size k requires k colors
- The chromatic number $\chi(G)$ of G is an upper bound on its clique number $\omega(G)$



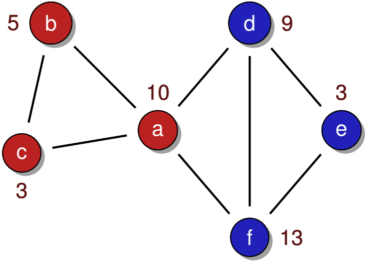
Upper bound

- Unweighted case: *coloring*
- A clique of size k requires k colors
- The chromatic number $\chi(G)$ of G is an upper bound on its clique number $\omega(G)$
- We work on the complement, so non-neighbors cannot share the same color $\Rightarrow$ clique cover



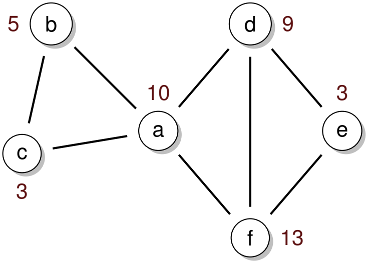
Upper bound

- Unweighted case: clique cover
- An independent set of size k requires k cliques
- The chromatic number $\chi(\overline{G})$ of the complement of G is an upper bound on G's independence number $\alpha(G)$
- We work on the complement, so non-neighbors cannot share the same color $\Rightarrow$ clique cover



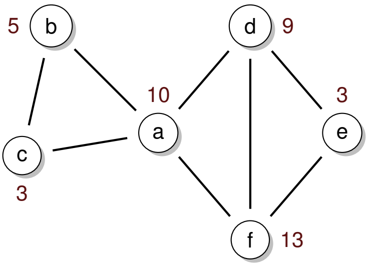
Upper bound

- Weighted case: *clique multi cover* [Babel 94]



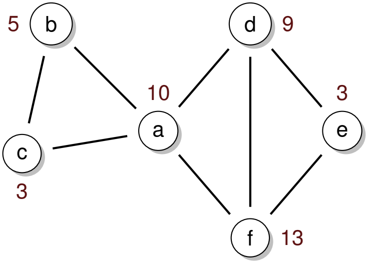
Upper bound

- Weighted case: *clique multi cover* [Babel 94]
- A vertex of weight w must be covered by w cliques



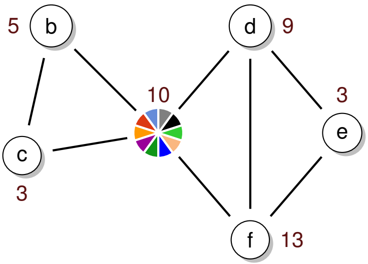
Upper bound

- Weighted case: *clique multi cover* [Babel 94]
- A vertex of weight w must be covered by w cliques
- The total number of clique is an upper bound on weight of the heaviest independent set



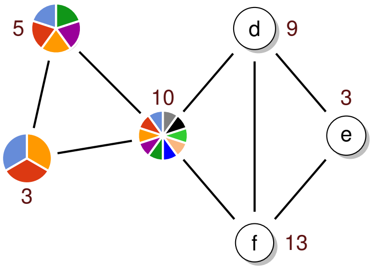
Upper bound

- Weighted case: *clique multi cover* [Babel 94]
- A vertex of weight w must be covered by w cliques
- The total number of clique is an upper bound on weight of the heaviest independent set
- Babel's complexity depends on the number of colors ($|E|k$)



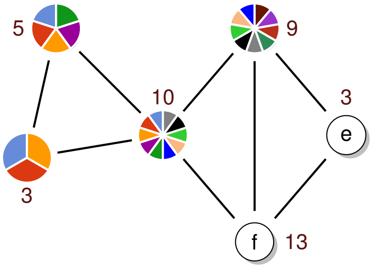
Upper bound

- Weighted case: *clique multi cover* [Babel 94]
- A vertex of weight w must be covered by w cliques
- The total number of clique is an upper bound on weight of the heaviest independent set
- Babel's complexity depends on the number of colors ($|E|k$)



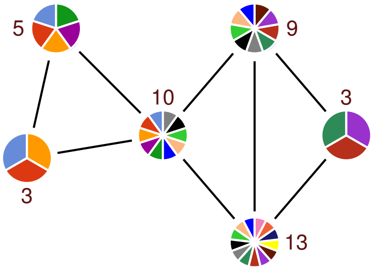
Upper bound

- Weighted case: *clique multi cover* [Babel 94]
- A vertex of weight w must be covered by w cliques
- The total number of clique is an upper bound on weight of the heaviest independent set
- Babel's complexity depends on the number of colors ($|E|k$)



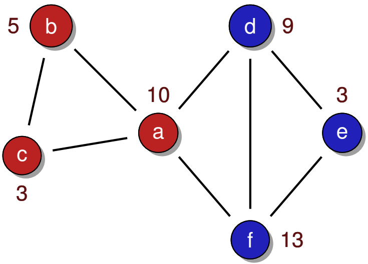
Upper bound

- Weighted case: *clique multi cover* [Babel 94]
- A vertex of weight w must be covered by w cliques
- The total number of clique is an upper bound on weight of the heaviest independent set
- Babel's complexity depends on the number of colors ($|E|k$)



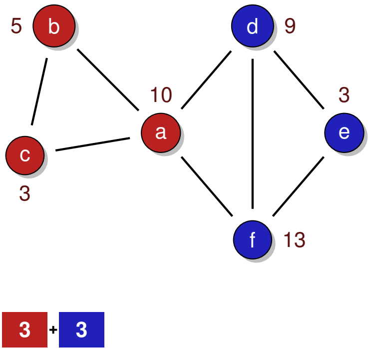
Upper bound

- Weighted case: *clique multi cover* [Babel 94]
- A vertex of weight w must be covered by w cliques
- The total number of clique is an upper bound on weight of the heaviest independent set
- Babel's complexity depends on the number of colors ($|E|k$)
  - method similar to [Tavares 16] ($|V|^3$)



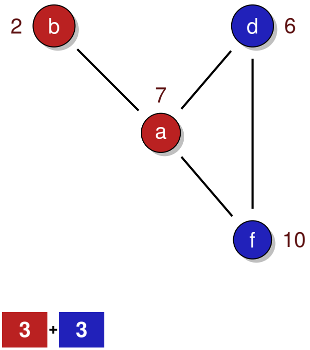
Upper bound

- Weighted case: *clique multi cover* [Babel 94]
- A vertex of weight w must be covered by w cliques
- The total number of clique is an upper bound on weight of the heaviest independent set
- Babel's complexity depends on the number of colors ($|E|k$)
  - method similar to [Tavares 16] ($|V|^3$)



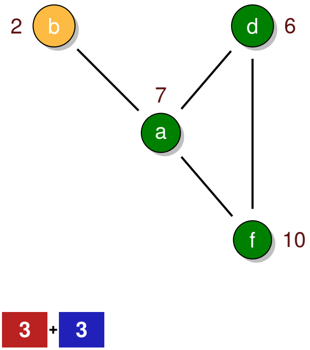
Upper bound

- Weighted case: *clique multi cover* [Babel 94]
- A vertex of weight w must be covered by w cliques
- The total number of clique is an upper bound on weight of the heaviest independent set
- Babel's complexity depends on the number of colors ($|E|k$)
  - method similar to [Tavares 16] ($|V|^3$)



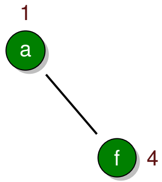
Upper bound

- Weighted case: *clique multi cover* [Babel 94]
- A vertex of weight w must be covered by w cliques
- The total number of clique is an upper bound on weight of the heaviest independent set
- Babel's complexity depends on the number of colors ($|E|k$)
  - method similar to [Tavares 16] ($|V|^3$)



Upper bound

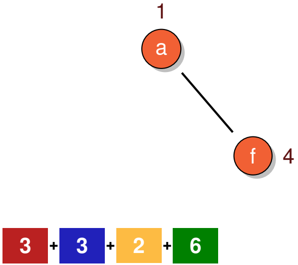
- Weighted case: *clique multi cover* [Babel 94]
- A vertex of weight w must be covered by w cliques
- The total number of clique is an upper bound on weight of the heaviest independent set
- Babel's complexity depends on the number of colors ($|E|k$)
  - method similar to [Tavares 16] ($|V|^3$)



3 + 3 + 2 + 6

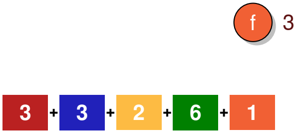
Upper bound

- Weighted case: *clique multi cover* [Babel 94]
- A vertex of weight w must be covered by w cliques
- The total number of clique is an upper bound on weight of the heaviest independent set
- Babel's complexity depends on the number of colors ($|E|k$)
  - method similar to [Tavares 16] ($|V|^3$)



Upper bound

- Weighted case: *clique multi cover* [Babel 94]
- A vertex of weight w must be covered by w cliques
- The total number of clique is an upper bound on weight of the heaviest independent set
- Babel's complexity depends on the number of colors ($|E|k$)
  - method similar to [Tavares 16] ($|V|^3$)



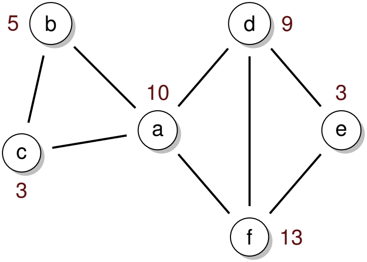
Upper bound

- Weighted case: *clique multi cover* [Babel 94]
- A vertex of weight w must be covered by w cliques
- The total number of clique is an upper bound on weight of the heaviest independent set
- Babel's complexity depends on the number of colors ($|E|k$)
  - method similar to [Tavares 16] ($|V|^3$)

$$3+3+2+6+1+3=18$$

Pruning

- [Babel 94]'s pruning rule: marginal cost of adding a vertex to the *IS*
- At most one vertex per clique in the *IS*

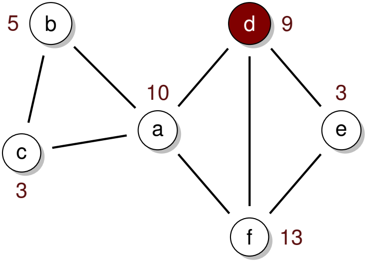


$\{a, b, c\}\{d, e, f\}$ $\{b\}$ $\{a, d, f\}$ $\{a, f\}$ $\{f\}$

3 + 3 + 2 + 6 + 1 + 3 = 18

Pruning

- [Babel 94]'s pruning rule: marginal cost of adding a vertex to the *IS*
- At most one vertex per clique in the *IS*
- E.g. if $d \in IS$ then a, e, f cannot be in the *IS*, then:

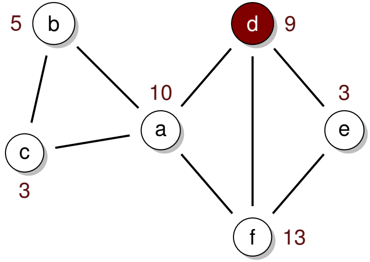


$\{a, b, c\}\{d, e, f\}$ $\{b\}$ $\{a, d, f\}$ $\{a, f\}$ $\{f\}$

3 + 3 + 2 + 6 + 1 + 3 = 18

Pruning

- [Babel 94]'s pruning rule: marginal cost of adding a vertex to the *IS*
- At most one vertex per clique in the *IS*
- E.g. if $d \in IS$ then a, e, f cannot be in the *IS*, then:
  - No vertices from cliques included in $N(d)$
  - Marginal cost of $d \in IS$ is 4

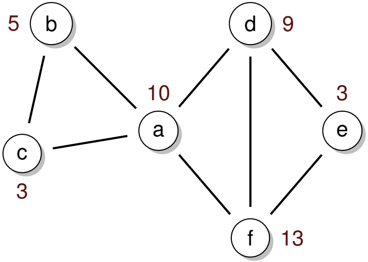


$\{a, b, c\}\{d, e, f\}$ $\{b\}$ $\{a, d, f\}$

3 + 3 + 2 + 6 + 1 + 3 = 14

Dominance

- “Dual” rule: marginal cost of $v \in VC$

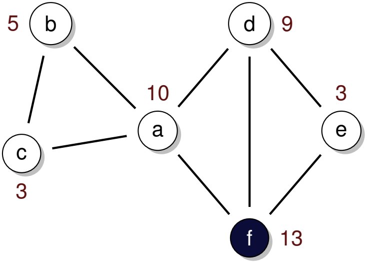


$\{a, b, c\}\{d, e, f\}$ $\{b\}$ $\{a, d, f\}$ $\{a, f\}$ $\{f\}$

3 + 3 + 2 + 6 + 1 + 3 = 18

Dominance

- "Dual" rule: marginal cost of $v \in VC$
- Lose $w(f) = 13$, gain at most $\sum_{u \in N(f)} w(u) = 22$

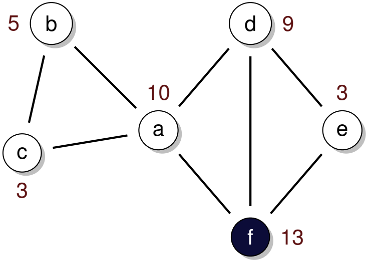


$\{a, b, c\}\{d, e, f\}$ $\{b\}$ $\{a, d, f\}$ $\{a, f\}$ $\{f\}$

3 + 3 + 2 + 6 + 1 + 3 = 18

Dominance

- "Dual" rule: marginal cost of $v \in VC$
- Lose $w(f) = 13$, gain at most $\sum_{u \in N(f)} w(u) = 22$
- At most one vertex per clique in the *IS*

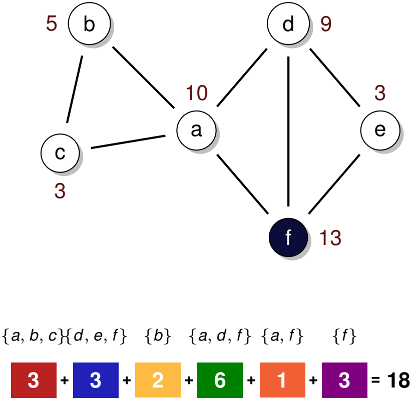


$\{a, b, c\}\{d, e, f\}$ $\{b\}$ $\{a, d, f\}$ $\{a, f\}$ $\{f\}$

3 + 3 + 2 + 6 + 1 + 3 = 18

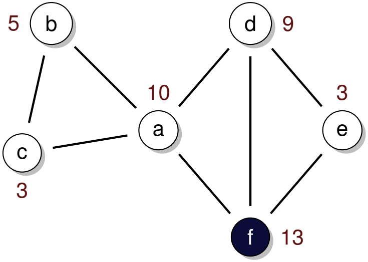
Dominance

- "Dual" rule: marginal cost of $v \in VC$
- Lose $w(f) = 13$, gain at most $\sum_{u \in N(f)} w(u) = 22$
- At most one vertex per clique in the IS
  - Gain at most 1 per neighbor clique



Dominance

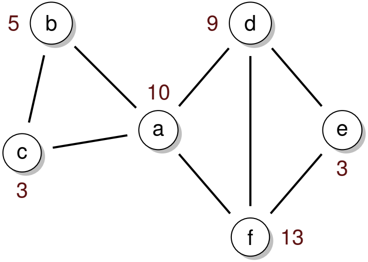
- "Dual" rule: marginal cost of $v \in VC$
- Lose $w(f) = 13$, gain at most $\sum_{u \in N(f)} w(u) = 22$
- At most one vertex per clique in the *IS*
  - Gain at most 1 per neighbor clique
  - Lose 13, win ≤ 13: might as well be in *IS*



$\{a, b, c\}\{d, e, f\}$ $\{b\}$ $\{a, d, f\}$ $\{a, f\}$ $\{f\}$

3 + 3 + 2 + 6 + 1 + 3 = 13

Explanation

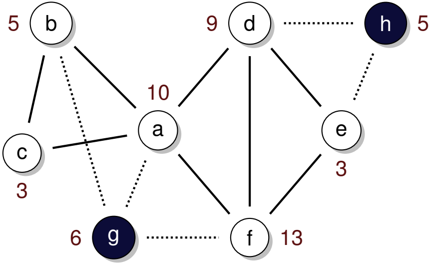


$\{a, b, c\}\{d, e, f\}$ $\{b\}$ $\{a, d, f\}$ $\{a, f\}$ $\{f\}$

3 + 3 + 2 + 6 + 1 + 3 = 18

Explanation

- Residual graph after two decisions:
  $g \in VC$ and $h \in VC$

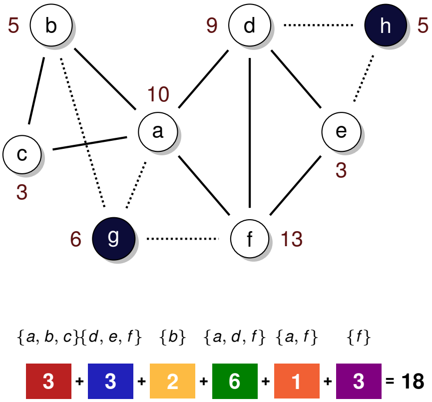


$\{a, b, c\}\{d, e, f\}$ $\{b\}$ $\{a, d, f\}$ $\{a, f\}$ $\{f\}$

$3 + 3 + 2 + 6 + 1 + 3 = 18$

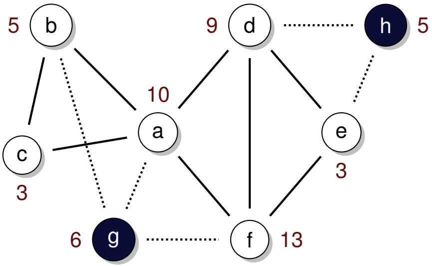
Explanation

- Residual graph after two decisions: $g \in VC$ and $h \in VC$
- We have found an *IS* of weight 21
  - Failure since $21 \geq 18$



Explanation

- Residual graph after two decisions: $g \in VC$ and $h \in VC$
- We have found an *IS* of weight 21
  - Failure since $21 \geq 18$
- Explanation: minimal clause that entails the current upper bound when falsified

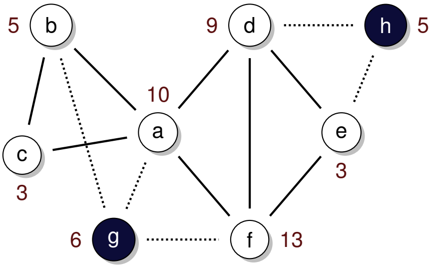


$\{a, b, c\}\{d, e, f\}$ $\{b\}$ $\{a, d, f\}$ $\{a, f\}$ $\{f\}$

3 + 3 + 2 + 6 + 1 + 3 = **18**

Explanation

- Residual graph after two decisions: $g \in VC$ and $h \in VC$
- We have found an *IS* of weight 21
  - Failure since $21 \geq 18$
- Explanation: minimal clause that entails the current upper bound when falsified
- Trivial explanation: $g \in IS$ or $h \in IS$

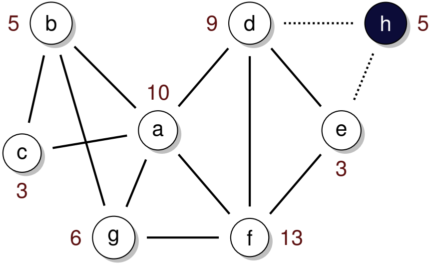


$\{a, b, c\}\{d, e, f\}$ $\{b\}$ $\{a, d, f\}$ $\{a, f\}$ $\{f\}$

3 + 3 + 2 + 6 + 1 + 3 = 18

Explanation

- Residual graph after two decisions: $g \in VC$ and $h \in VC$
- We have found an *IS* of weight 21
  - Failure since $21 \geq 18$
- Explanation: minimal clause that entails the current upper bound when falsified
- Trivial explanation: $g \in IS$ or $h \in IS$

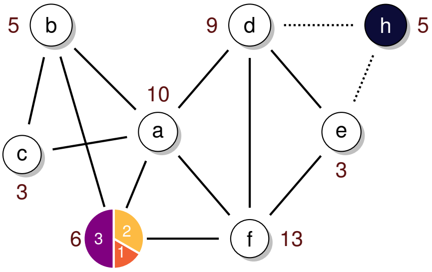


$\{a, b, c\}\{d, e, f\}$ $\{b\}$ $\{a, d, f\}$ $\{a, f\}$ $\{f\}$

3 + 3 + 2 + 6 + 1 + 3 = 18

Explanation

- Residual graph after two decisions: $g \in VC$ and $h \in VC$
- We have found an *IS* of weight 21
  - Failure since $21 \geq 18$
- Explanation: minimal clause that entails the current upper bound when falsified
- Trivial explanation: $g \in IS$ or $h \in IS$

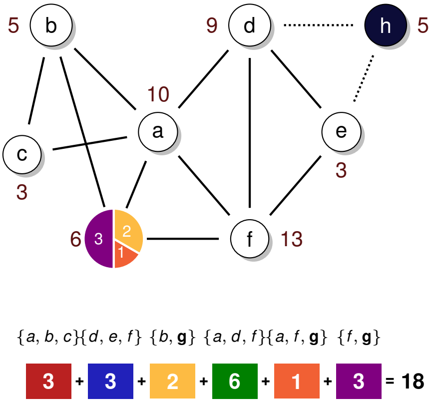


$\{a, b, c\}\{d, e, f\}$ $\{b, \mathbf{g}\}$ $\{a, d, f\}\{a, f, \mathbf{g}\}$ $\{f, \mathbf{g}\}$

3 + 3 + 2 + 6 + 1 + 3 = 18

Explanation

- Residual graph after two decisions: $g \in VC$ and $h \in VC$
- We have found an *IS* of weight 21
  - Failure since $21 \geq 18$
- Explanation: minimal clause that entails the current upper bound when falsified
- Trivial explanation: $g \in IS$ or $h \in IS$
- Reduced explanation: $h \in IS$



Explanation

- The stack of decisions/deductions contains literals $[v \in VC]$ or $[v \in IS]$

Explanation

- The stack of decisions/deductions contains literals $[v \in VC]$ or $[v \in IS]$
- A literal $[v \in IS]$ is followed by the literals $[u \in VC]$ for $u \in N(v)$

Explanation

- The stack of decisions/deductions contains literals $[v \in VC]$ or $[v \in IS]$
- A literal $[v \in IS]$ is followed by the literals $[u \in VC]$ for $u \in N(v)$

Explanation Algorithm

- Explore the stack of decisions/deductions in reverse order
  - Given a literal $[v \in VC]$: try to remove it, and keep it otherwise

Explanation

- The stack of decisions/deductions contains literals $[v \in VC]$ or $[v \in IS]$
- A literal $[v \in IS]$ is followed by the literals $[u \in VC]$ for $u \in N(v)$

Explanation Algorithm

- Explore the stack of decisions/deductions in reverse order
  - Given a literal $[v \in VC]$: try to remove it, and keep it otherwise
  - Given a literal $[v \in VC]$: compute the cost of removing it

Explanation

- The stack of decisions/deductions contains literals $[v \in VC]$ or $[v \in IS]$
- A literal $[v \in IS]$ is followed by the literals $[u \in VC]$ for $u \in N(v)$

Explanation Algorithm

- Explore the stack of decisions/deductions in reverse order
  - Given a literal $[v \in VC]$: try to remove it, and keep it otherwise
  - Given a literal $[v \in VC]$: compute the cost of removing it
  - Given a literal $[v \in IS]$,

Explanation

- The stack of decisions/deductions contains literals $[v \in VC]$ or $[v \in IS]$
- A literal $[v \in IS]$ is followed by the literals $[u \in VC]$ for $u \in N(v)$

Explanation Algorithm

- Explore the stack of decisions/deductions in reverse order
  - Given a literal $[v \in VC]$: try to remove it, and keep it otherwise
  - Given a literal $[v \in VC]$: compute the cost of removing it
  - Given a literal $[v \in IS]$, if the cost of removing $[u \in VC]$ for $u \in N(v)$ is too high:
    - keep it, and remove $[u \in VC]$ for $u \in N(v)$

Explanation

- The stack of decisions/deductions contains literals $[v \in VC]$ or $[v \in IS]$
- A literal $[v \in IS]$ is followed by the literals $[u \in VC]$ for $u \in N(v)$

Explanation Algorithm

- Explore the stack of decisions/deductions in reverse order
  - Given a literal $[v \in VC]$: try to remove it, and keep it otherwise
  - Given a literal $[v \in VC]$: compute the cost of removing it
  - Given a literal $[v \in IS]$, if the cost of removing $[u \in VC]$ for $u \in N(v)$ is too high:
    - keep it, and remove $[u \in VC]$ for $u \in N(v)$
  - Otherwise:
    - remove $[v \in IS]$ and try to remove as many $[u \in VC]$ for $u \in N(v)$ as possible

Experimental evaluation: methods

- `mwclq` [Fang *et al.* 16]
- `wlmc` [Jiang *et al.* 17]
- `cliquer` [Ostergard 01]
- `OTClique` [Shimuzu *et al.* 17]
- `Tavares` [Tavares 16] (implementation [McCreesh *et al.* 17])

Experimental evaluation: benchmarks

Experimental evaluation: benchmarks

- `DIMACS` Maximum Clique
- `BHOSLIB` Maximum Independent Set

Experimental evaluation: benchmarks

- DIMACS Maximum Clique, $w(v_i) = (i \bmod 200) + 1$
- BHOSLIB Maximum Independent Set, $w(v_i) = (i \bmod 200) + 1$

LAAS CNRS

Experimental evaluation: benchmarks

- DIMACS Maximum Clique, $w(v_i) = (i \bmod 200) + 1$
- BHOSLIB Maximum Independent Set, $w(v_i) = (i \bmod 200) + 1$
- Structured benchmarks proposed by citationMcCreesh *et al.* 17
  - WDP Winner Determination Problem in combinatorial auctions
  - EC-CODE Design of error-correction codes
  - REF Optimisation of university evaluation
  - KIDNEY Maximizes the number/emergency of kidney exchanges

Experimental evaluation: results on classes

- Objective function: Geometric average weight

Experimental evaluation: results on classes

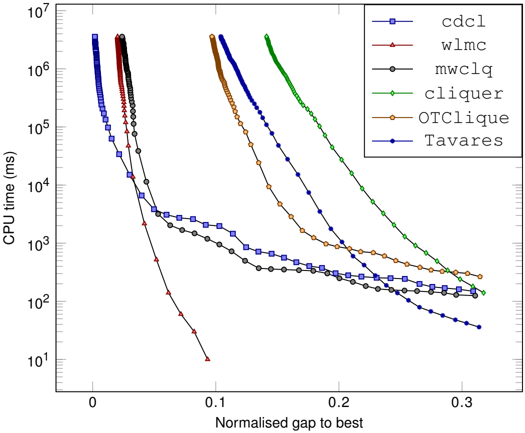
- Objective function: Geometric average weight

		cdcl	wlmc	mwclq	cliquer	OTClique	Tavares
		objective	objective	objective	objective	objective	objective
BHOSLIB	(40)	4672.66	3770.83	4598.76	835.05	1619.57	4277.46
WDP	(50)	84.95**M**	85.53**M**	85.53**M**	85.53**M**	85.53**M**	84.81**M**
EC-CODE	(15)	97.31	97.31	96.88	97.31	97.31	97.31
DIMACS	(160)	3277.55	3232.41	3252.04	2079.63	2496.57	3146.91
REF	(129)	129.82	128.11	128.61	105.06	117.88	129.24
KIDNEY	(188)	549.71**B**	549.41**B**	516.48**B**	537.69**B**	540.15**B**	544.41**B**

Experimental evaluation: global results

Mean normalised gap to the best solution average over every instance of:

- maximum weight u
- minimum weight l
- gap of weight $g(w) = \begin{cases} \frac{u-w}{u-l} & \text{if } u > l \\ 0 & \text{otherwise} \end{cases}$
  - 0 if best, 1 if worst



PhD Thesis on combinatorial oprimization / machine learning with Renault

- Based at LAAS (Toulouse), visits to Renault (Paris)
- Fundamental research / Industrial applications
  - Routing in workshop, Car sequencing, Project Scheduling, ?
- Open topic: CDCL, DNN, Monte-Carlo tree search,...
- Attractive Salary
- Flexible starting date (end of 2018 to late spring 2019)

Fig. 1 – Cantine du LAAS

Fig. 2 – QG Renault

Questions?