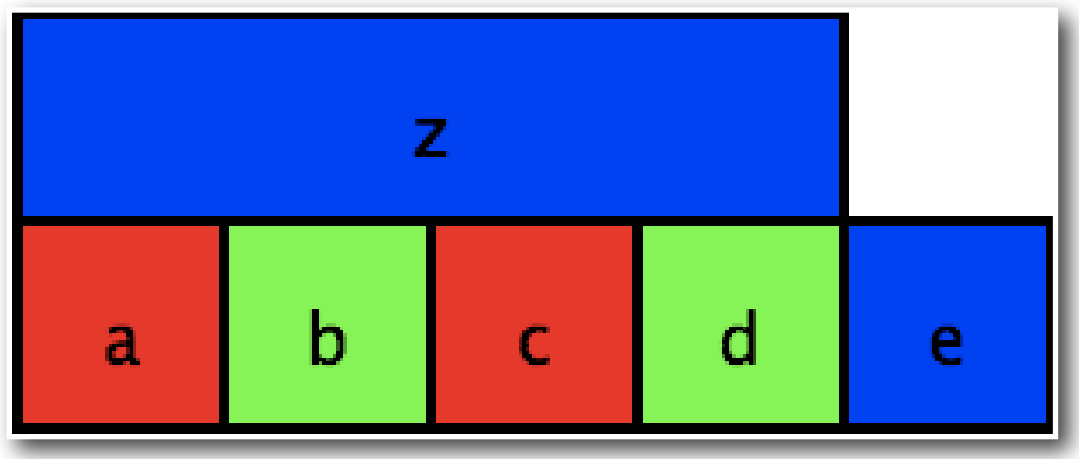
Planning and robotics

Geoff Gordon

A search problem

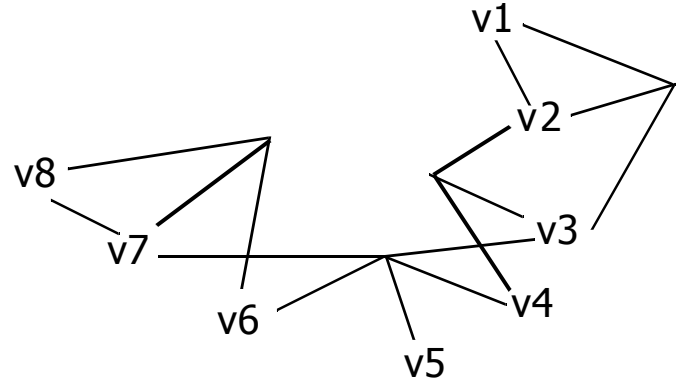


- 3-coloring

Other important search problems

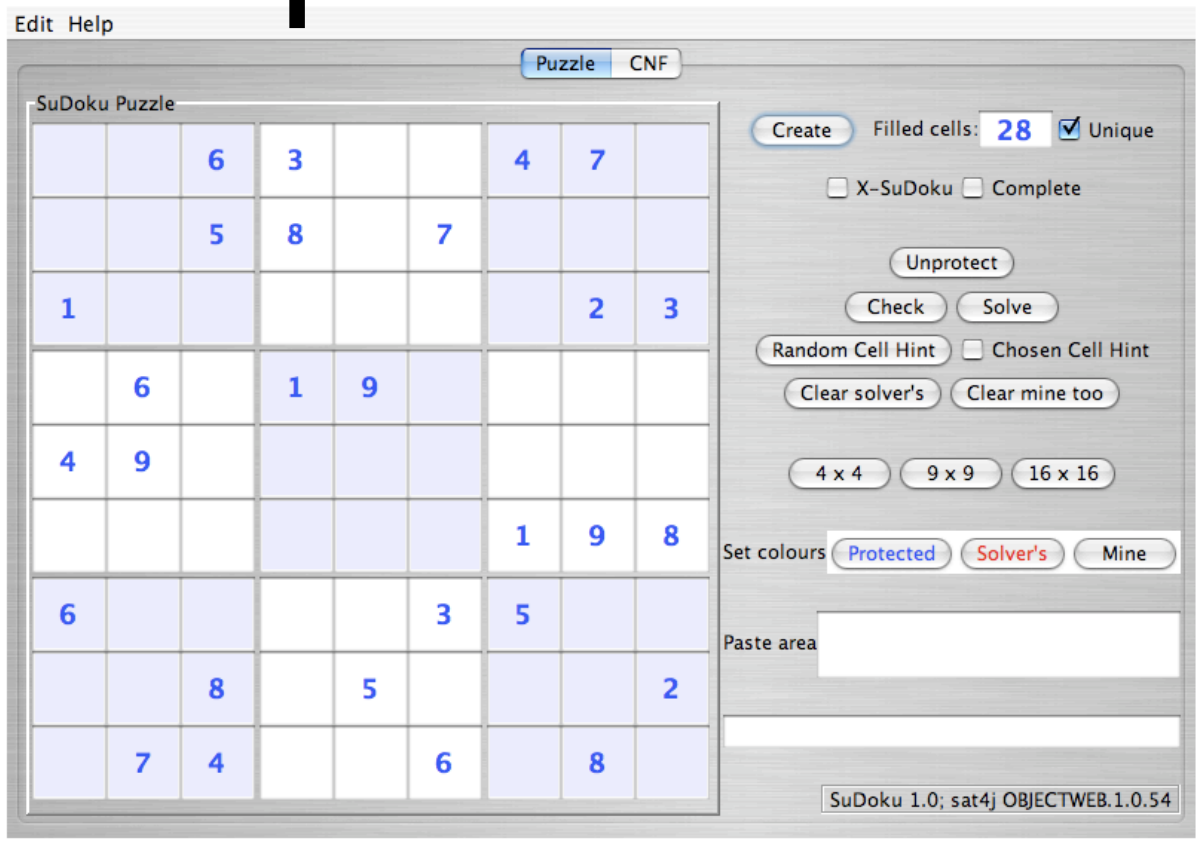
0	0	1	v1		
0	0	1	v2		
0	0	1	v3		
1	1	2	v4		
v8	v7	v6	v5		

V = { v1 , v2 , v3 , v4 , v5 , v6 , v7 , v8 }, D = { B (bomb) , S (space) }
C = { (v1,v2) : { (B , S) , (S,B) } ,(v1,v2,v3) : { (B,S,S) , (S,B,S) , (S,S,B)},...}



- Minesweeper (image courtesy Andrew Moore)

Other important search problems



- Sudoku

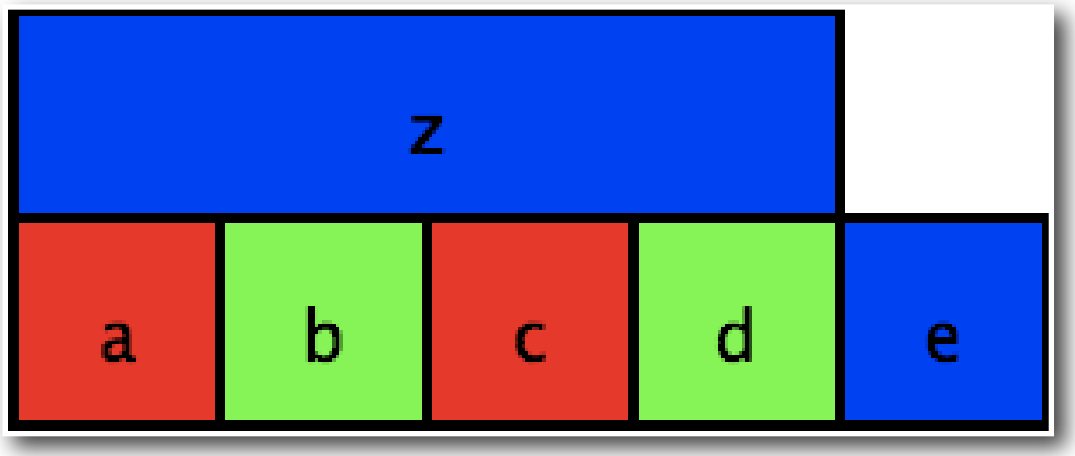
  http://www.cs.qub.ac.uk/~I.Spence/SuDoku/SuDoku.html

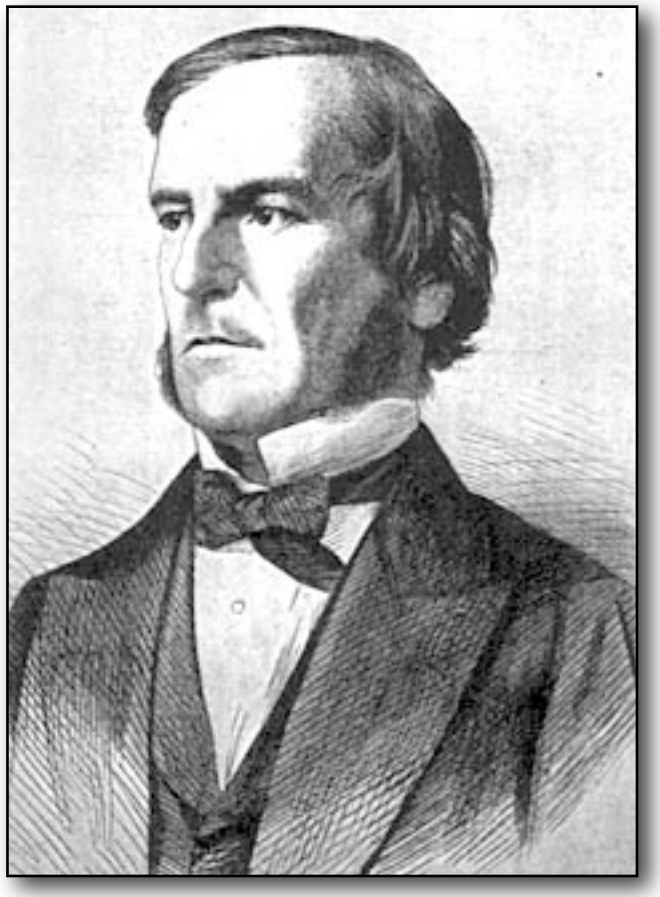
Other important search problems

Other important search problems

- Scheduling (e.g., of factory production)
- Facility location
- Circuit layout
- Multi-robot planning

Propositional logic





George Boole
1815–1864

The meaning of truth

$(a \vee \neg a)$ $(a \vee b \vee c) \wedge (b \vee \neg d)$

- Model:
- Valid:
- Satisfiable (SAT):

Ex: planning

init: have(cake)

goal: have(cake), eaten(cake)

eat(cake):

    pre: have(cake)

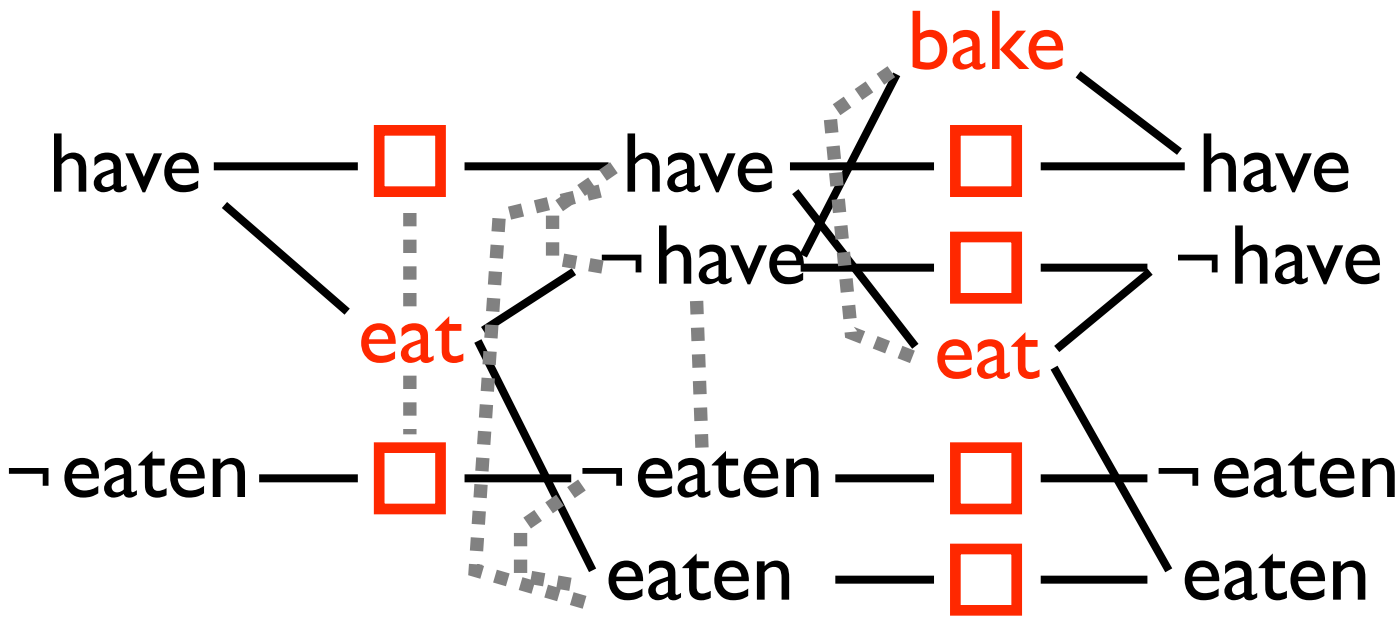
    eff: -have(cake), eaten(cake)

bake(cake):

    pre: -have(cake)

    eff: have(cake)

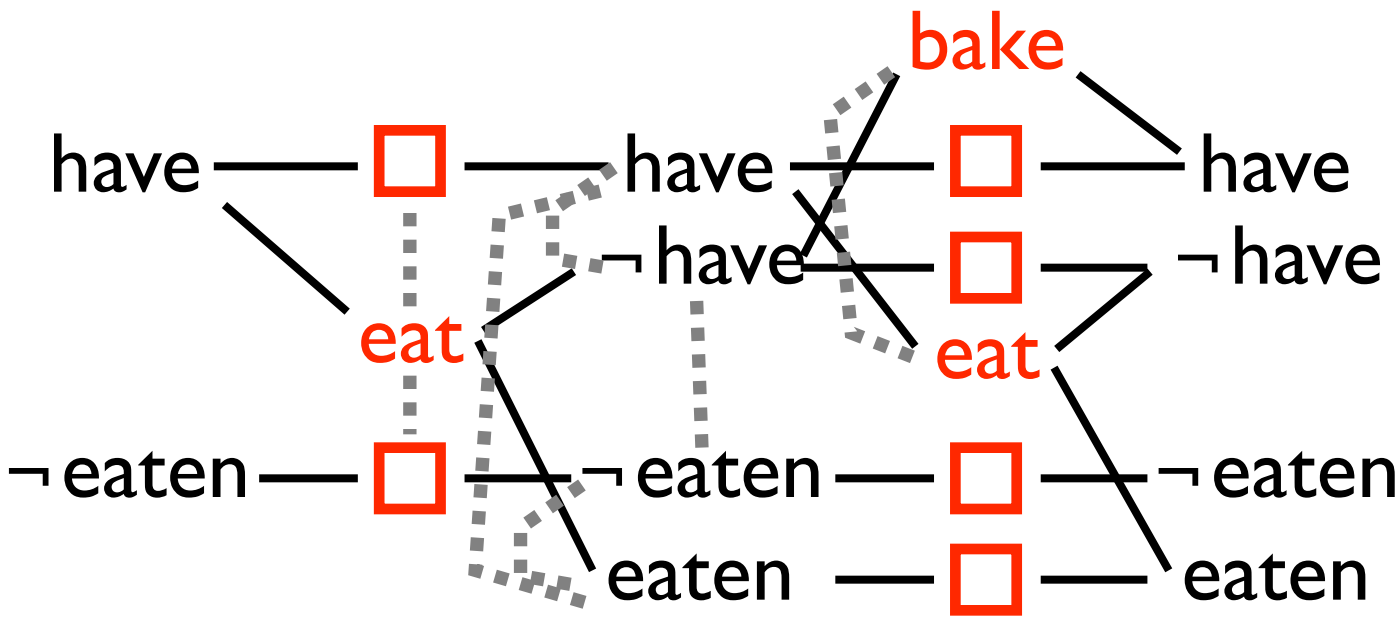
Plan graph


bake
have
have
have
¬ have
¬ have
eat
eat
¬ eaten
¬ eaten
¬ eaten
eaten
eaten

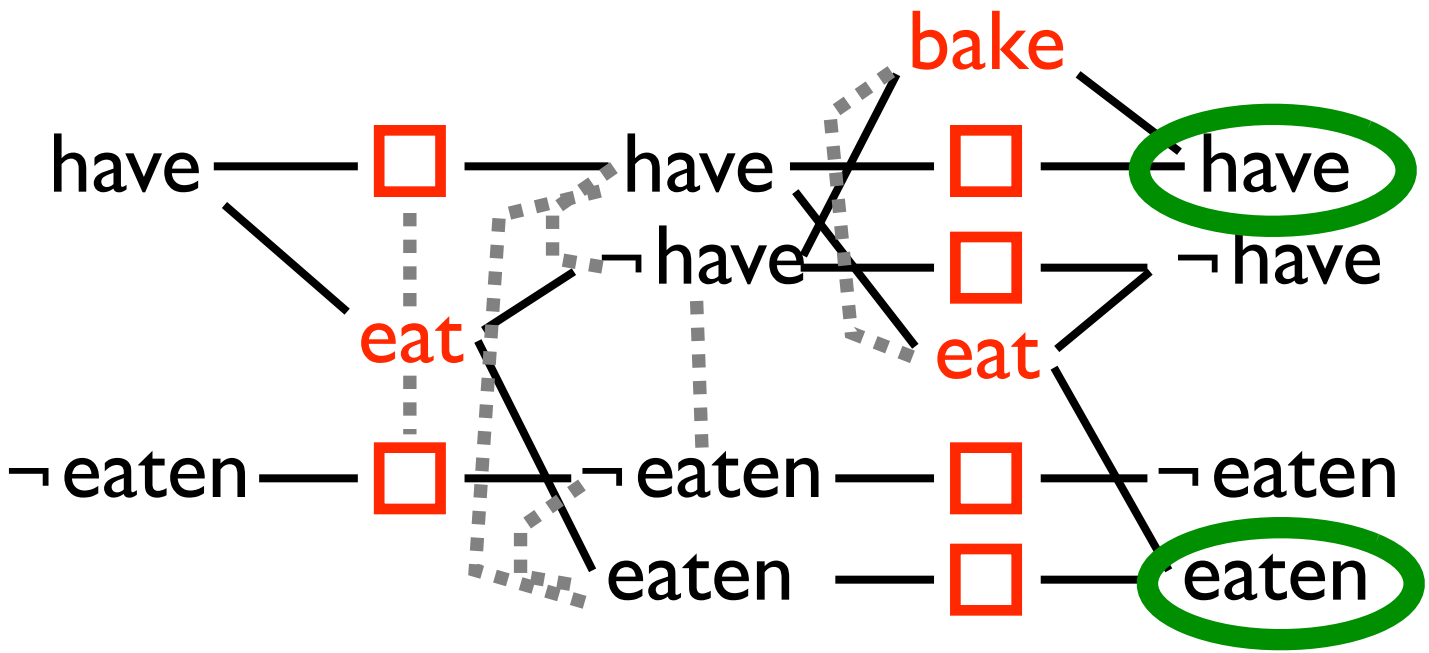
Planning as logic

$$(\overline{h}_1 \vee \overline{M}_1) \wedge (h_1 \vee M_1) \wedge (\overline{e}_1 \vee M_1) \wedge (e_1 \vee \overline{M}_1)$$
$$\wedge (\overline{M}_1 \vee \overline{h}_1) \wedge (\overline{M}_1 \vee e_1) \wedge (\overline{M}_2 \vee h_1) \wedge (\overline{M}_2 \vee \overline{h}_2)$$
$$\wedge (\overline{M}_2 \vee e_2) \wedge (\overline{B}_2 \vee \overline{h}_1) \wedge (\overline{B}_2 \vee h_2) \wedge (\overline{h}_2 \vee h_1 \vee B_2)$$
$$\wedge (\overline{h}_2 \vee \overline{M}_2 \vee B_2) \wedge (h_2 \vee \overline{h}_1 \vee M_2)$$
$$\wedge (h_2 \vee \overline{B}_2 \vee M_2) \wedge (\overline{e}_2 \vee e_1 \vee M_2)$$
$$\wedge (e_2 \vee \overline{e}_1) \wedge (e_2 \vee \overline{M}_2) \wedge (\overline{M}_2 \vee \overline{B}_2) \wedge (h_2) \wedge (e_2)$$

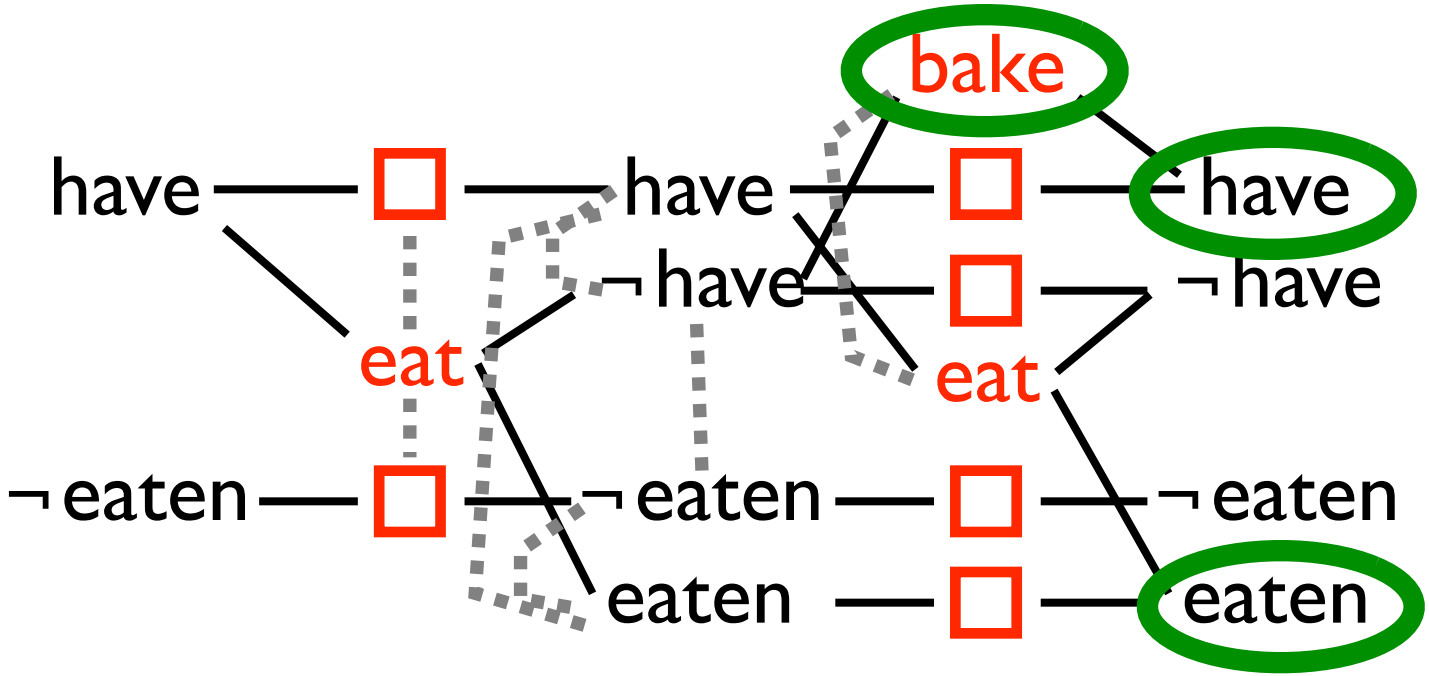
Plan search


bake
have
have
have
¬ have
¬ have
eat
eat
¬ eaten
¬ eaten
¬ eaten
eaten
eaten

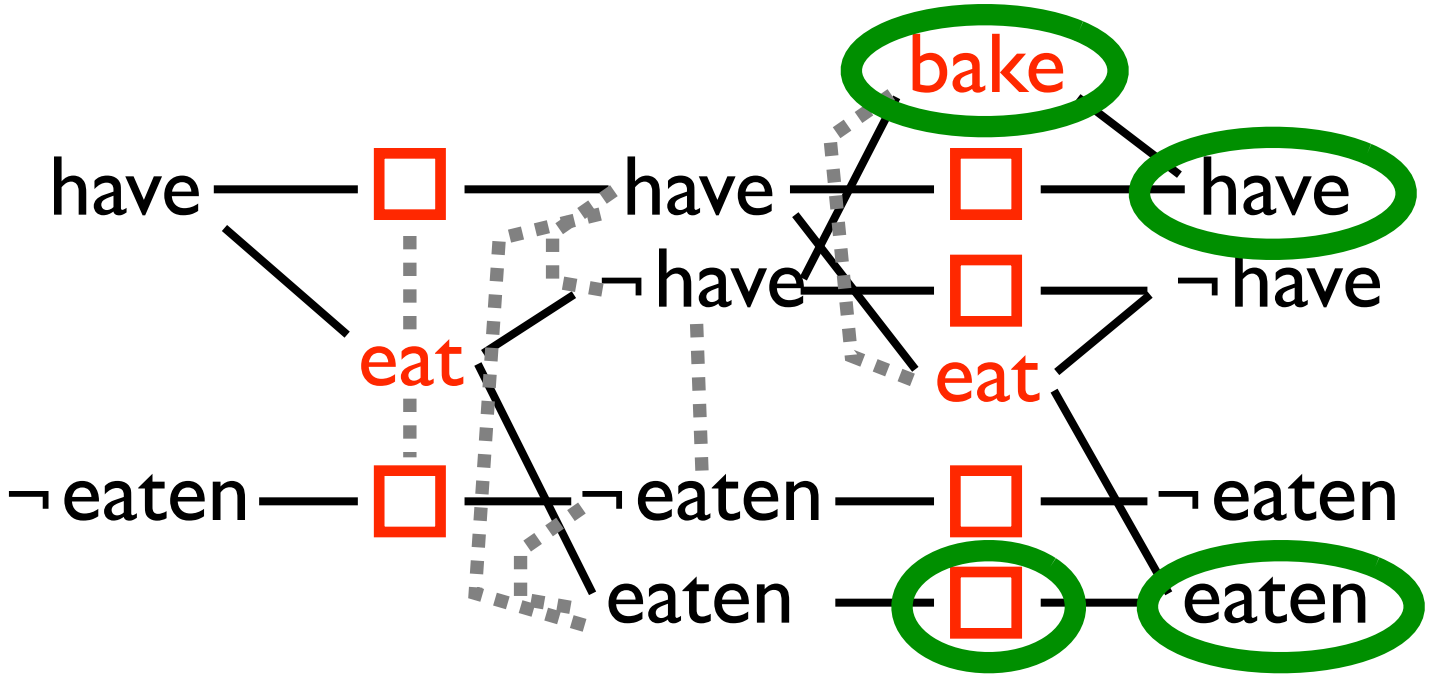
Plan search


bake
have
have
have
¬ have
¬ have
eat
eat
¬ eaten
¬ eaten
¬ eaten
eaten
eaten

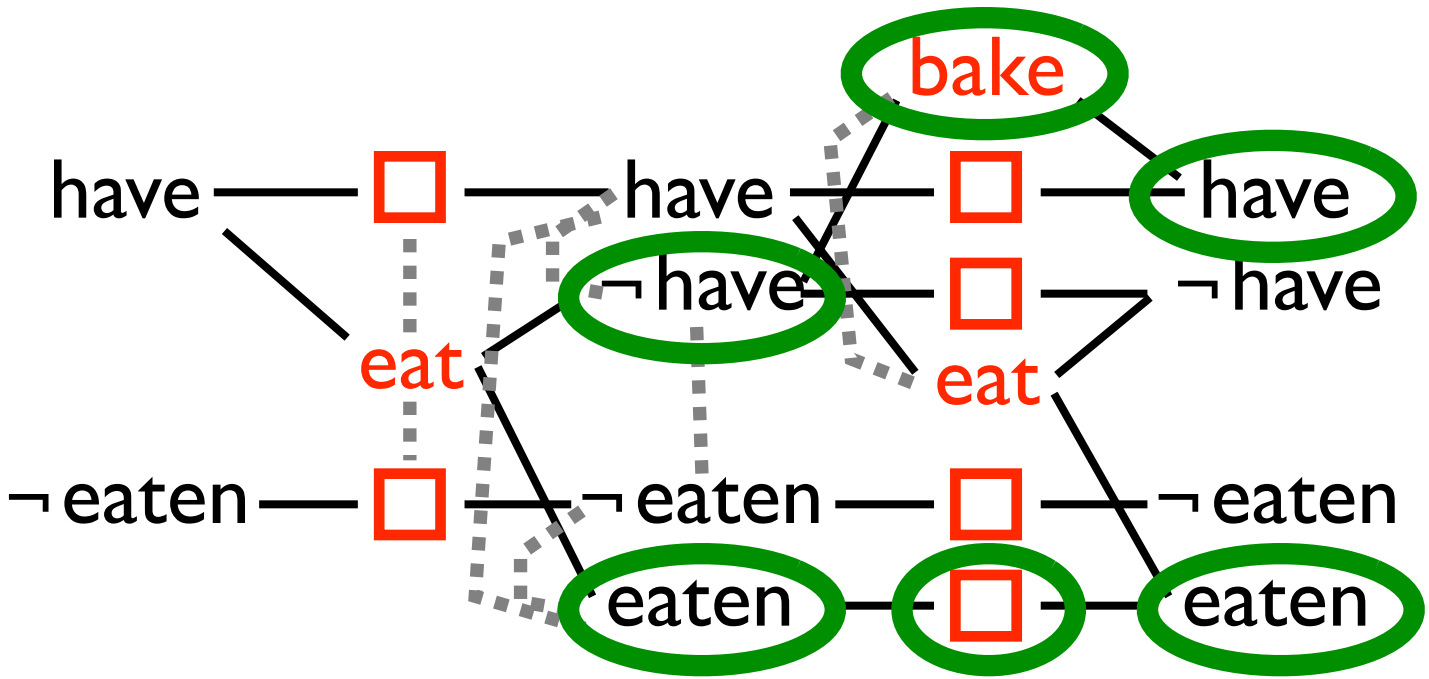
Plan search


bake
have
have
have
¬ have
¬ have
eat
eat
¬ eaten
¬ eaten
¬ eaten
eaten
eaten

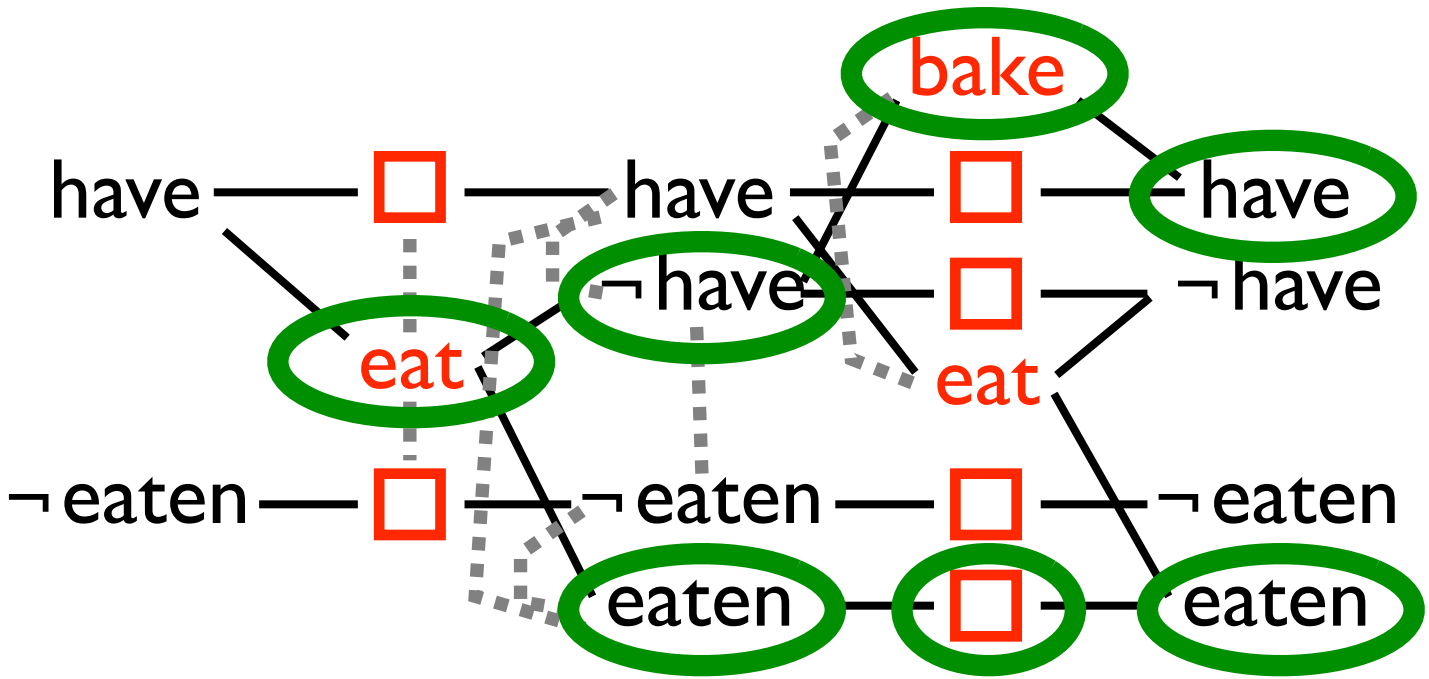
Plan search


bake
have
have
have
¬ have
¬ have
eat
eat
¬ eaten
¬ eaten
¬ eaten
eaten
eaten

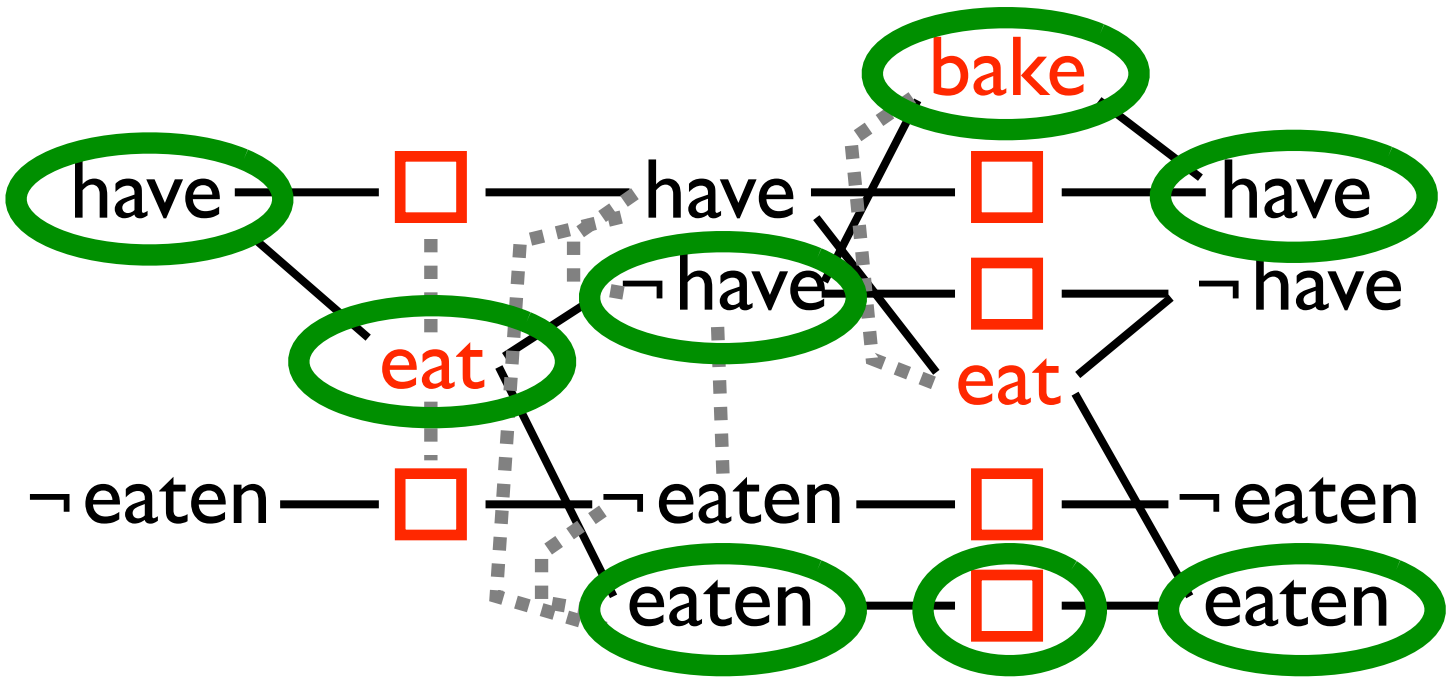
Plan search


bake
have
have
have
¬ have
¬ have
eat
eat
¬ eaten
¬ eaten
¬ eaten
eaten
eaten

Plan search


bake
have
have
have
¬ have
¬ have
eat
eat
¬ eaten
¬ eaten
¬ eaten
eaten
eaten

Plan search


bake
have
have
have
¬ have
¬ have
eat
eat
¬ eaten
¬ eaten
¬ eaten
eaten
eaten

SAT is general

- S. A. Cook (1971) proved that many useful search problems are "equivalent" to SAT
  - showed how to simulate a computer running "guess & check" search w/ (very complicated, but still poly-size) SAT problem
- Equivalently, SAT is exactly as hard (in theory) as other search problems
- In practice: SAT & MILP (below) go a long way

Working with logic

α, β, γ are arbitrary formulas

$$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha) \quad \text{commutativity of } \wedge$$
$$(\alpha \vee \beta) \equiv (\beta \vee \alpha) \quad \text{commutativity of } \vee$$
$$((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha \wedge (\beta \wedge \gamma)) \quad \text{associativity of } \wedge$$
$$((\alpha \vee \beta) \vee \gamma) \equiv (\alpha \vee (\beta \vee \gamma)) \quad \text{associativity of } \vee$$
$$\neg(\neg\alpha) \equiv \alpha \quad \text{double-negation elimination}$$

$$(\alpha \wedge (\beta \vee \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) \quad \text{distributivity of } \wedge \text{ over } \vee$$
$$(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) \quad \text{distributivity of } \vee \text{ over } \wedge$$

$$\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta) \quad \text{de Morgan}$$
$$\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta) \quad \text{de Morgan}$$

Proofs

rains $\Rightarrow$ pours

pours $\wedge$ outside $\Rightarrow$ rusty

rains

outside

Modus ponens

$$\frac{(a \wedge b \wedge c \Rightarrow d) \quad a \quad b \quad c}{d}$$

Modus tollens

$$\frac{(a \Rightarrow b) \quad \neg b}{\neg a}$$

- If it's raining the grass is wet; the grass is not wet; so, it's not raining

Resolution

$$\frac{(a \vee b \vee c) \quad (\neg c \vee d \vee e)}{a \vee b \vee d \vee e}$$

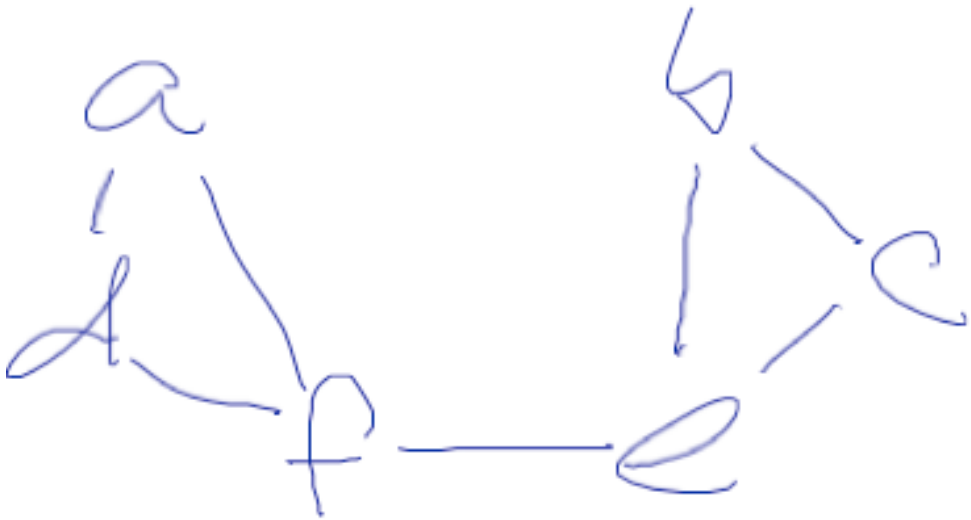
Proof of Resolution

$(a \vee b \vee c) \wedge (\neg c \vee d \vee e)$

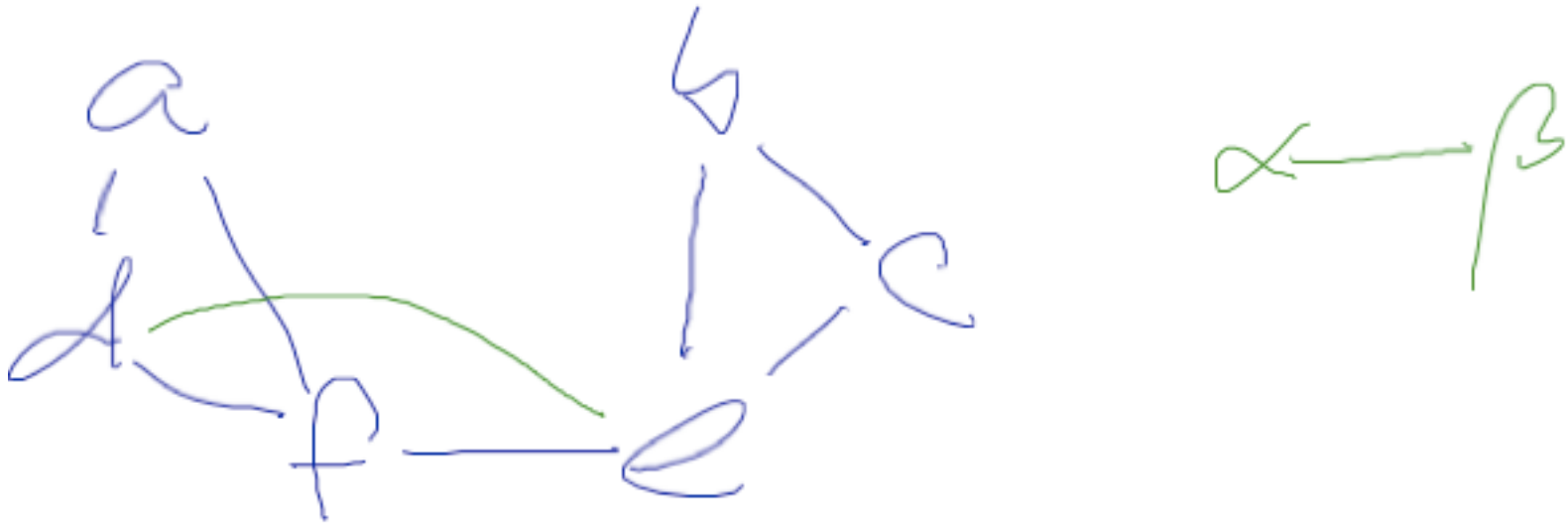
Solving SAT

- SAT-solvers routinely handle instances w/ 1,000,000 variables
- How? Depth-first search + optimizations.
  - search heuristics
  - constraint propagation
  - clause learning
  - randomization

Example: 3-coloring



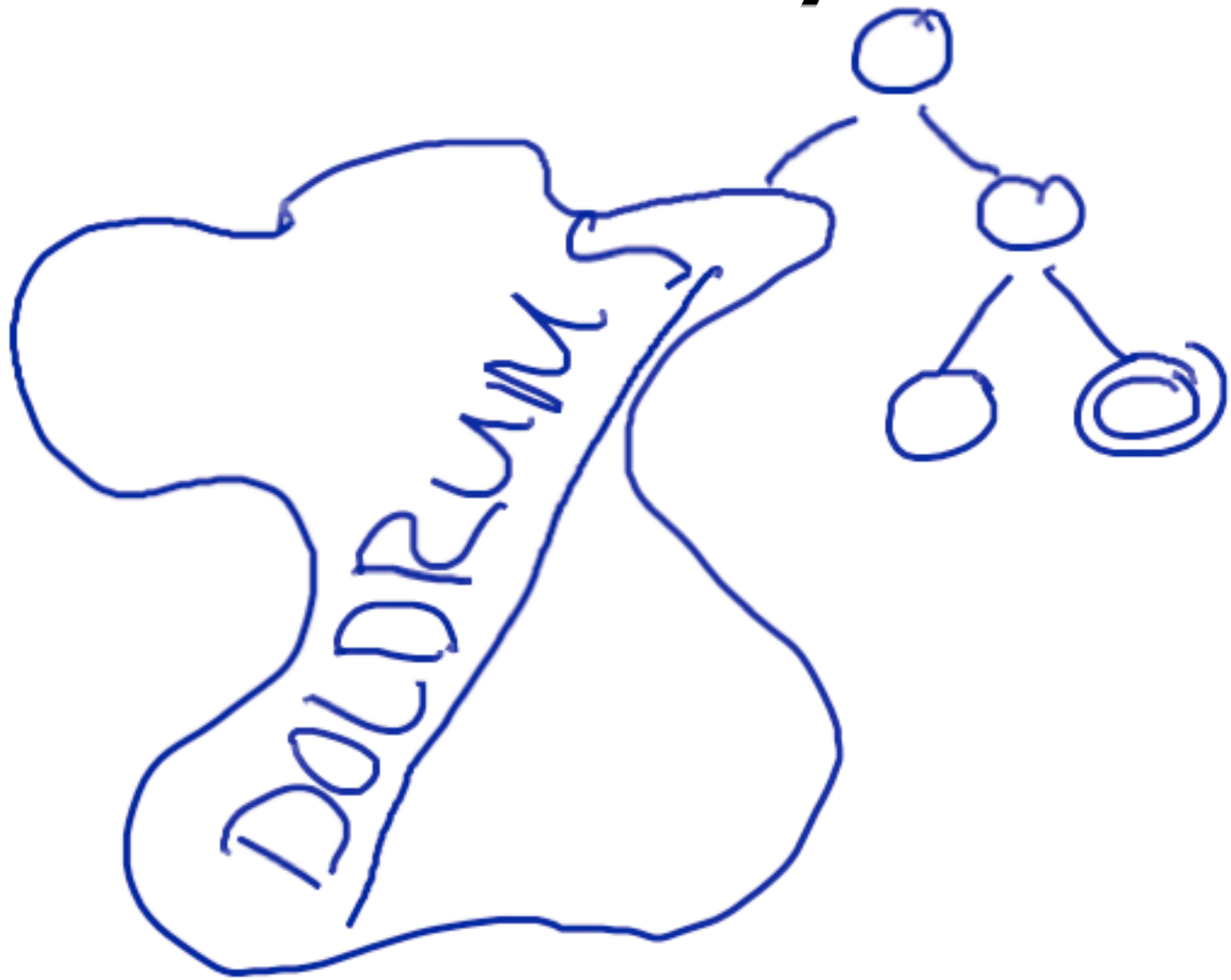
Clause learning



Randomness

- Standard SAT solvers are randomized
- Result is a significant variance in solution times for same formula (Chaff authors report seconds vs. days)

We can be very lucky or unlucky



The GWYDYR CASTLE in the Doldrums, while sharks hang around her, awaiting the next offering of galley slops

Doldrums: One Of Murphy's Yarns

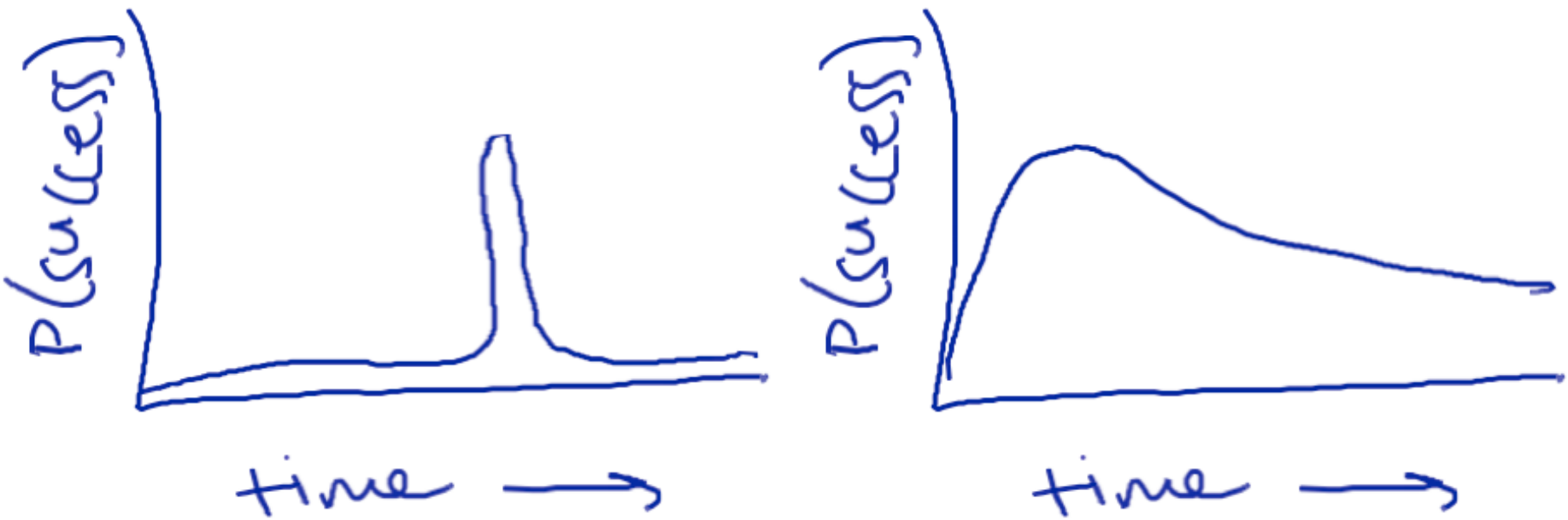
http://oldpoetry.com/opoem/56157 **Cicely Fox Smith**

"I heard onst of a barque," said Murphy.
"Becalmed, that couldn't get a breath,
Till all the crowd was sick with scurvy
An' the skipper drunk himself to death."

Exploiting variance

- Try multiple random seeds
  - influences order of expanding neighbors (when ordering heuristics are tied)
  - influences starting point in WalkSAT
- Interleave computation (or iterative lengthening)
- When does this work?

Randomization cont'd



- Randomization works well if search times are sometimes short but have heavy tail

Implementations

- This style of SAT-solver is "DPLL"
  - Davis, Putnam, Logemann, and Loveland
- Free code: Chaff, MiniSAT
- Not-free: ILOG CP (CPLEX)

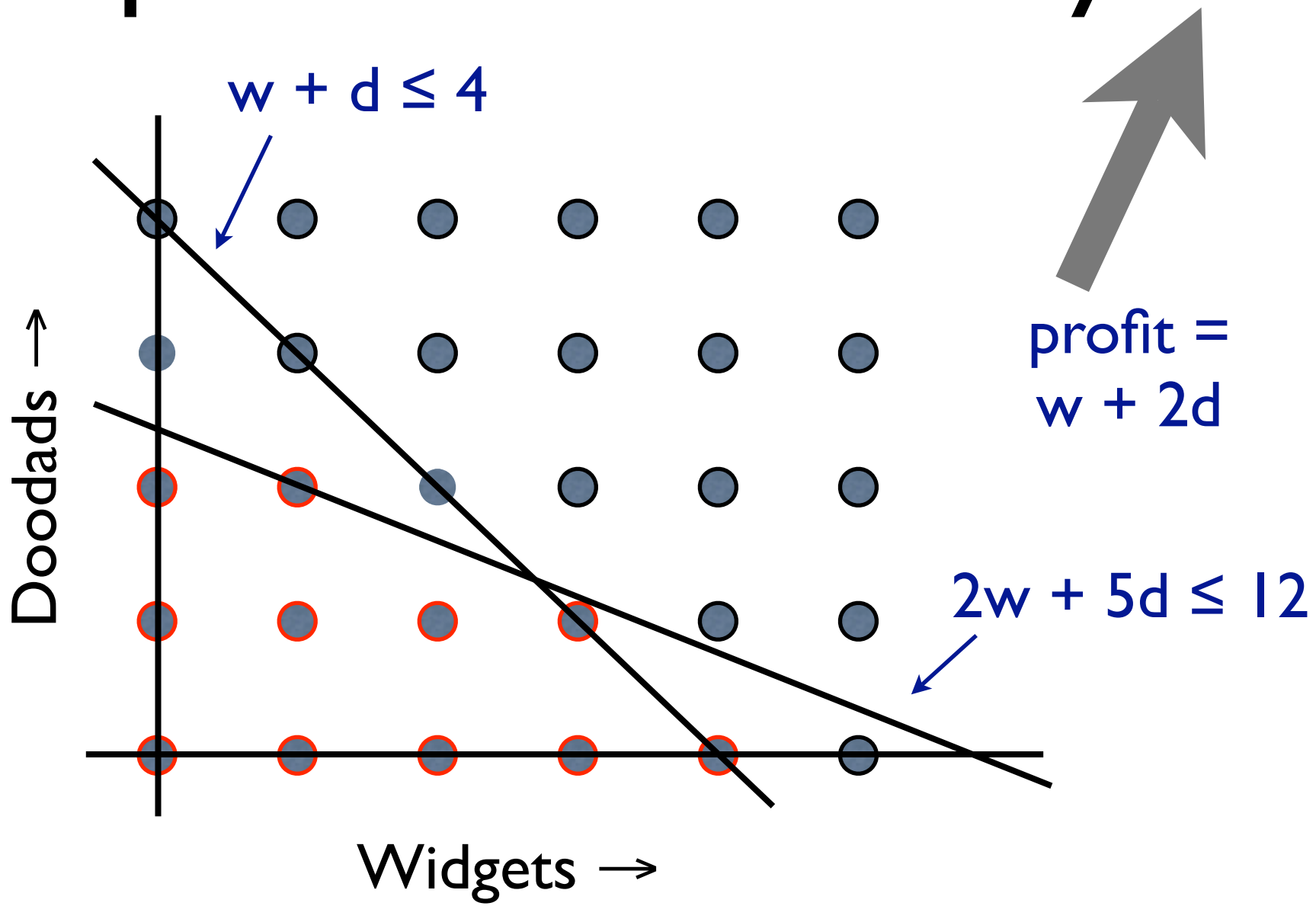
Optimization: w-MAXSAT

$$(\overline{h}_1 \vee \overline{M}_1) \wedge (h_1 \vee M_1) \wedge (\overline{e}_1 \vee M_1) \wedge (e_1 \vee \overline{M}_1)$$
$$\wedge (\overline{M}_1 \vee \overline{h}_1) \wedge (\overline{M}_1 \vee e_1) \wedge (\overline{M}_2 \vee h_1) \wedge (\overline{M}_2 \vee \overline{h}_2)$$
$$\wedge (\overline{M}_2 \vee e_2) \wedge (\overline{B}_2 \vee \overline{h}_1) \wedge (\overline{B}_2 \vee h_2) \wedge (\overline{h}_2 \vee h_1 \vee B_2)$$
$$\wedge (\overline{h}_2 \vee \overline{M}_2 \vee B_2) \wedge (h_2 \vee \overline{h}_1 \vee M_2)$$
$$\wedge (h_2 \vee \overline{B}_2 \vee M_2) \wedge (\overline{e}_2 \vee e_1 \vee M_2)$$
$$\wedge (e_2 \vee \overline{e}_1) \wedge (e_2 \vee \overline{M}_2) \wedge (\overline{M}_2 \vee \overline{B}_2) \wedge (h_2) \wedge (e_2)$$

Weights on literals

$$(\overline{h}_1 \vee \overline{M}_1) \wedge (h_1 \vee M_1) \wedge (\overline{e}_1 \vee M_1) \wedge (e_1 \vee \overline{M}_1)$$

Optimization: factory



(M)ILP

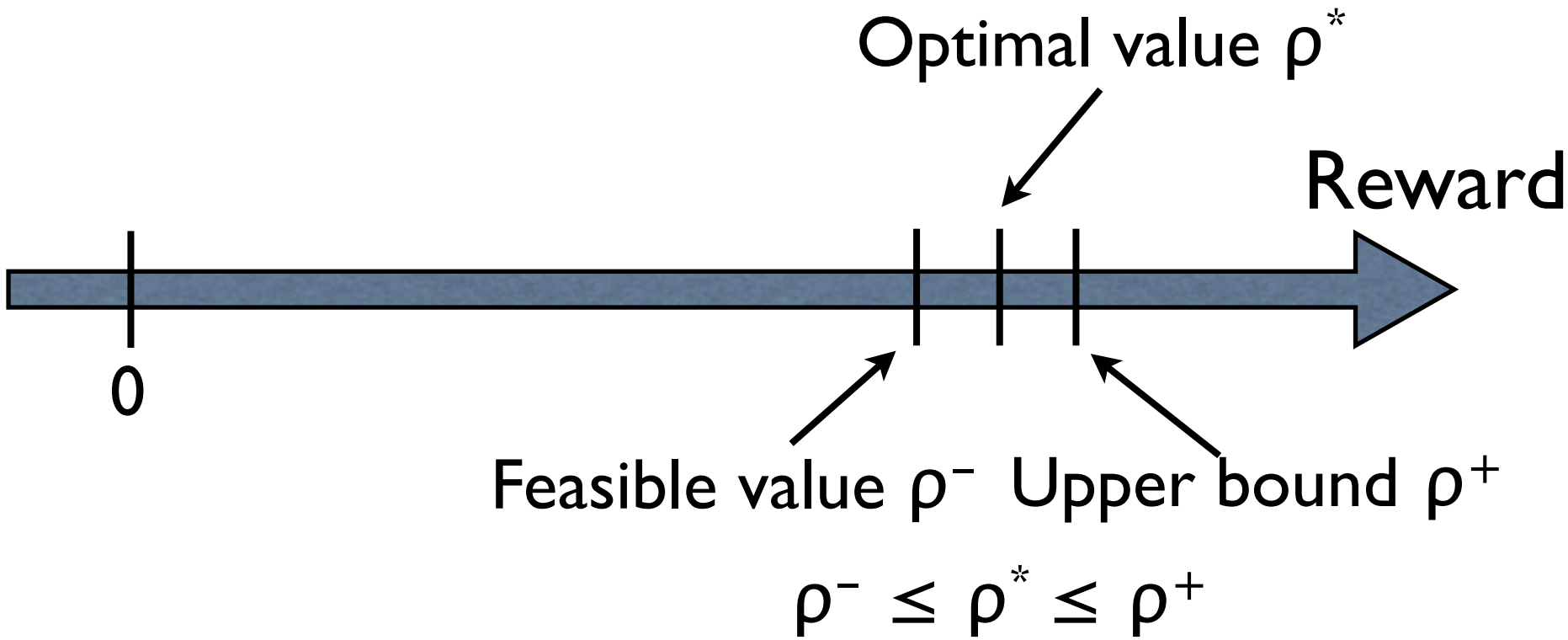
- These are **integer linear programs**
- Interesting related problems:
  - 0-1 ILP: all variables in {0, 1}
  - SAT: 0-1 ILP, all constraints of form

    $x + (1-y) + (1-z) \geq 1$
  - LP: no integer restrictions, all vars in $\mathbb{R}$
  - MILP: some variables in $\mathbb{R}$, others $\mathbb{Z}$

Solving MILPs

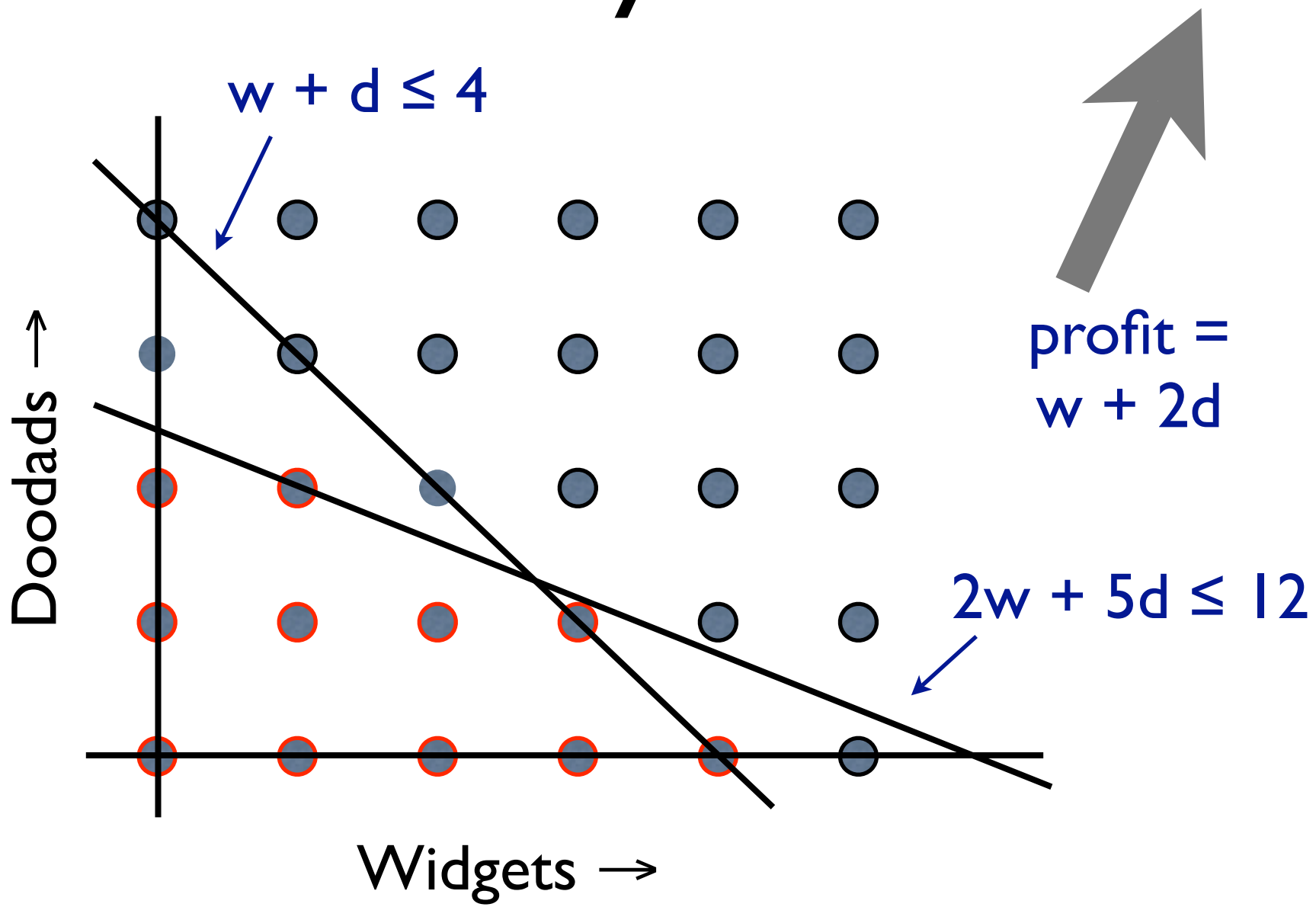
- Search
- Bounds
  - pessimistic: value of any feasible solution
  - optimistic: **relaxation**

Bounds

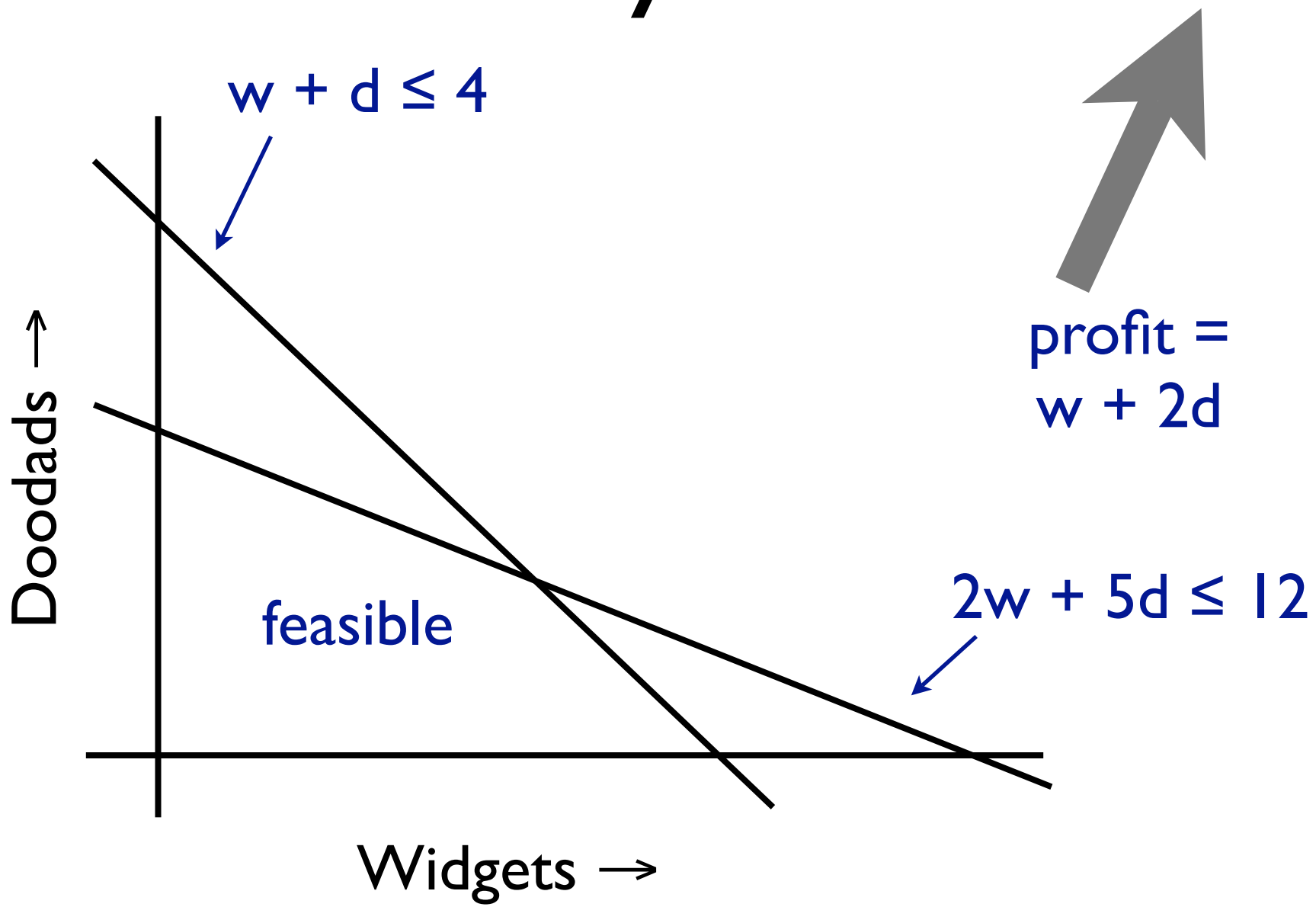


$$1 \geq \rho^-/\rho^* \geq \rho^-/\rho^+$$

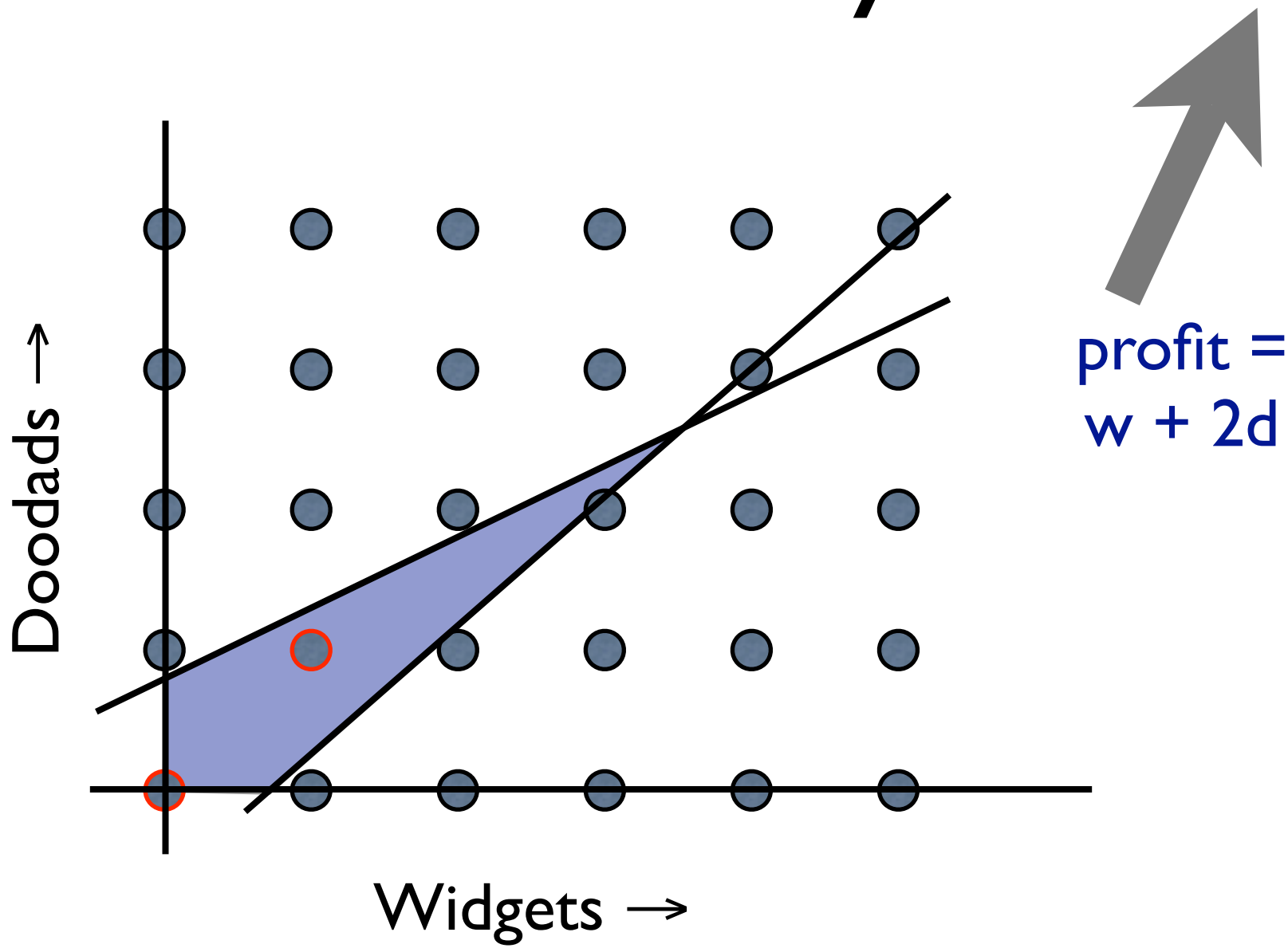
Factory ILP

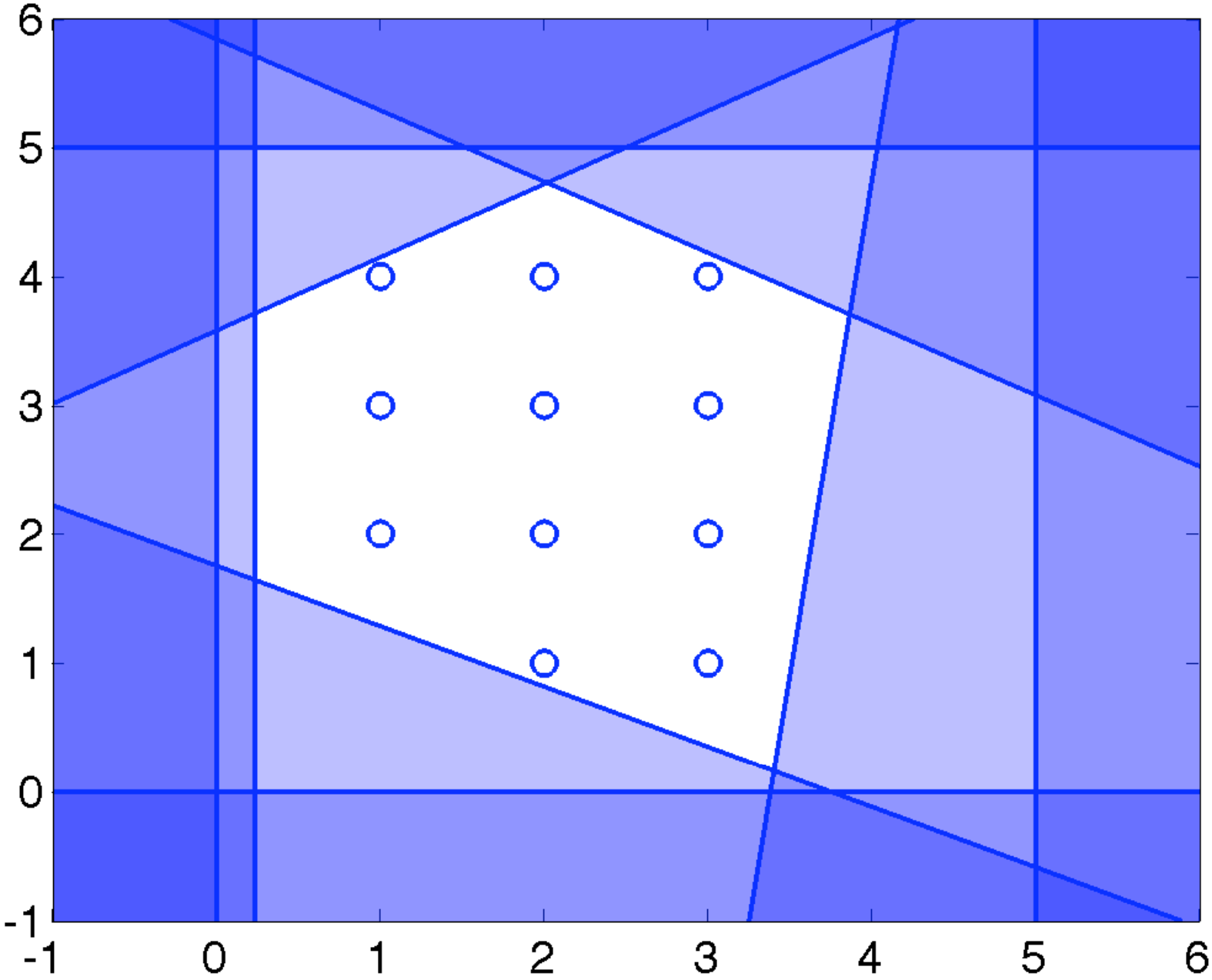


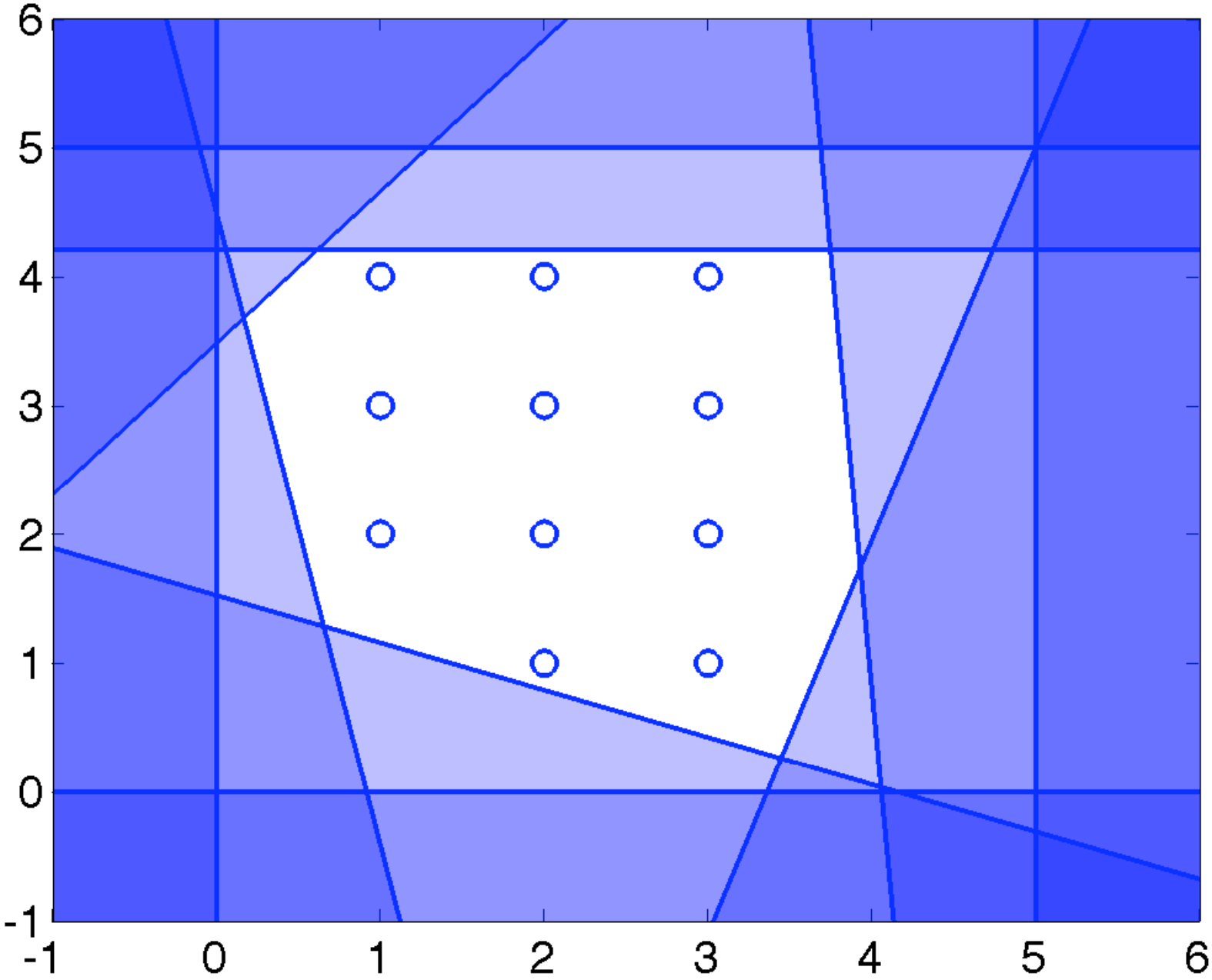
Factory LP

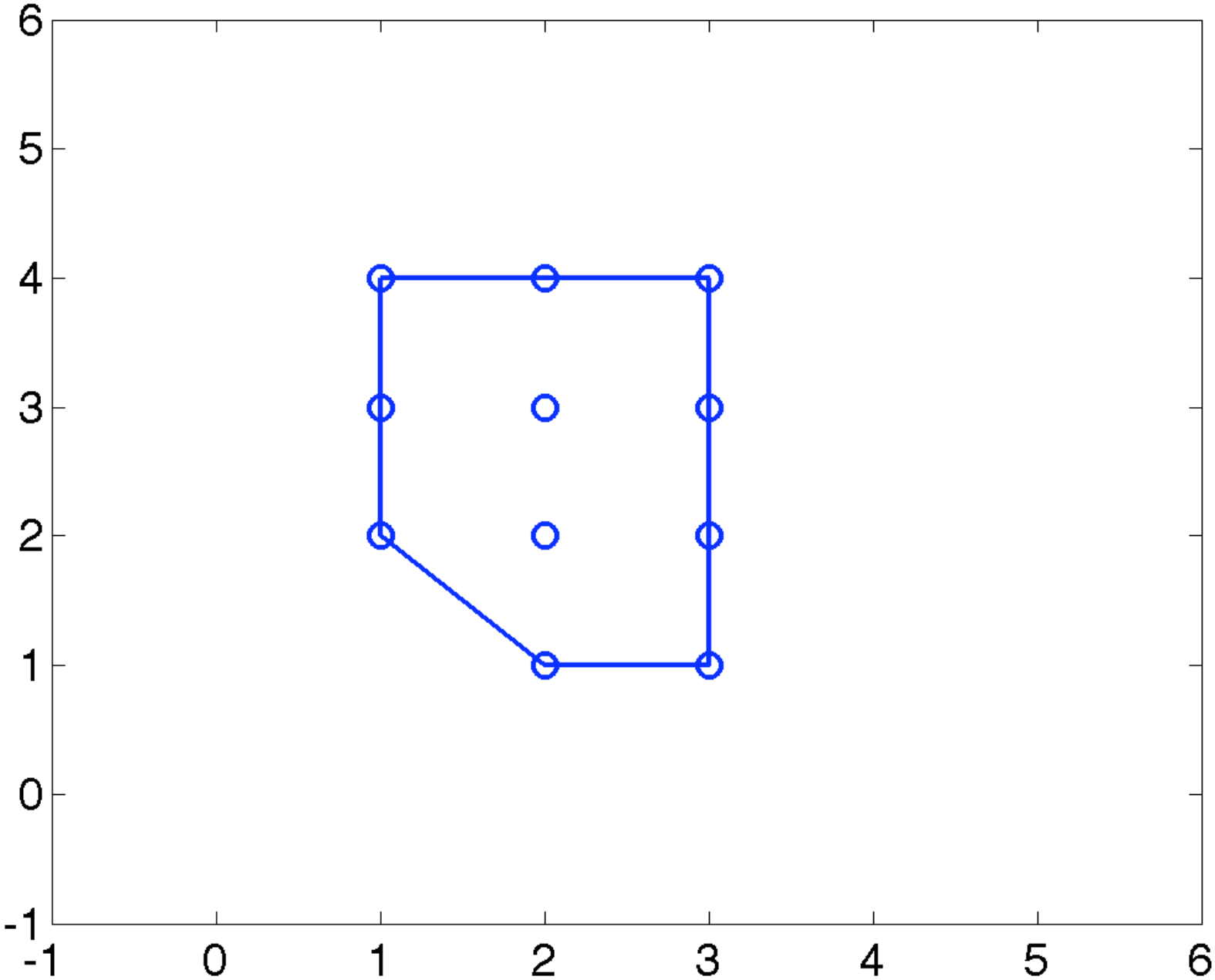


Unfortunately…




6
5
4
3
2
1
0
-1
-1
0
1
2
3
4
5
6


6
5
4
3
2
1
0
-1
-1
0
1
2
3
4
5
6


6
5
4
3
2
1
0
-1
-1
0
1
2
3
4
5
6

Branch & bound

[schema, value] = bb(F, sch, bnd)

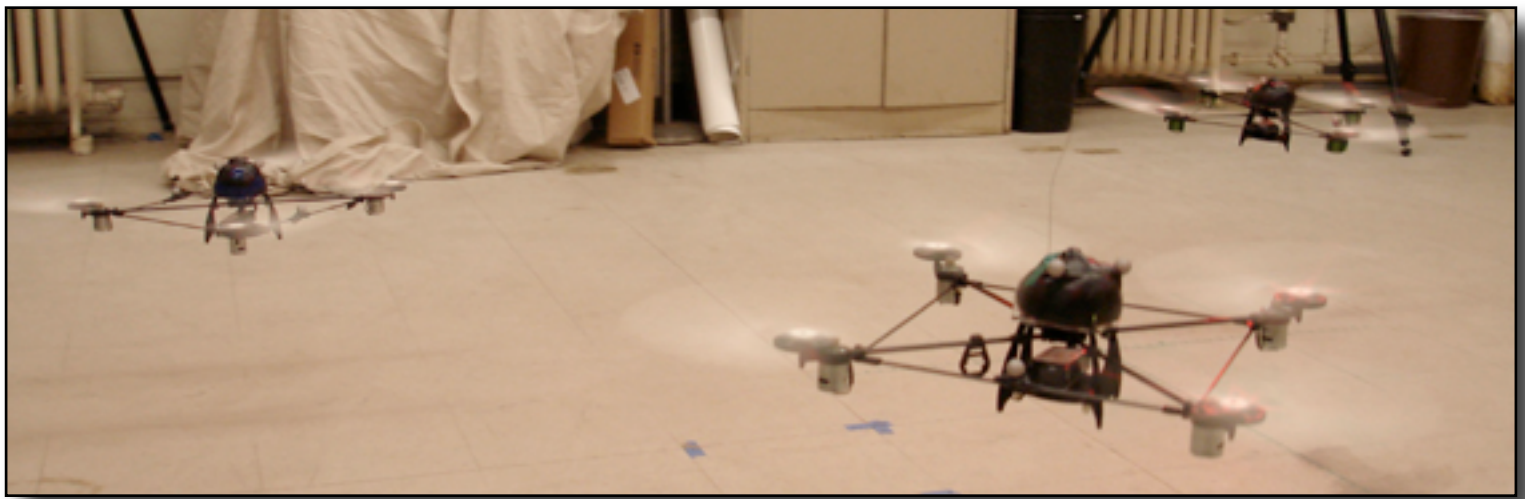
- $[v_{rx}, sch_{rx}]$ = relax(F, sch)
- if integer(sch_{rx}): return $[sch_{rx}, v_{rx}]$
- if $v_{rx} \geq$ bnd: return $[sch, v_{rx}]$
- Pick variable x_i
- $[sch^0, v^0]$ = bb(F, sch/(x_i: 0), bnd)
- $[sch^1, v^1]$ = bb(F, sch/(x_i: 1), min(bnd, v^0))
- if ($v^0 \leq v^1$): return $[sch^0, v^0]$
- else: return $[sch^1, v^1]$

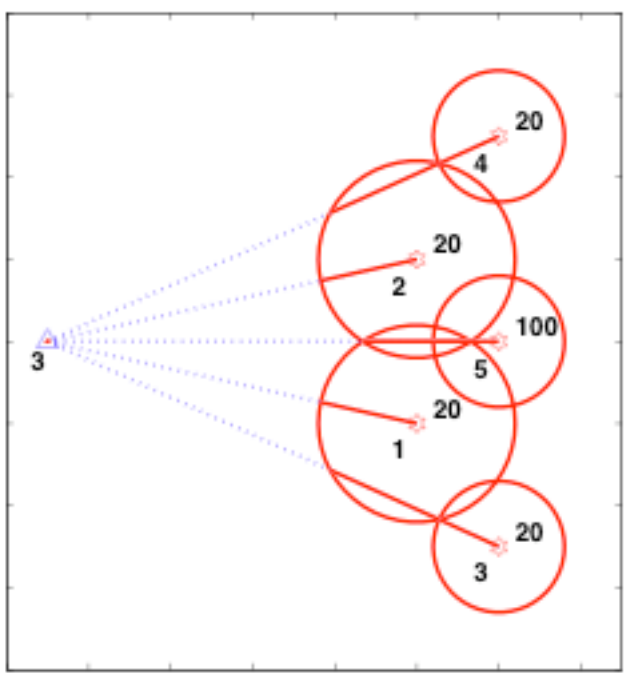
A random 3-CNF

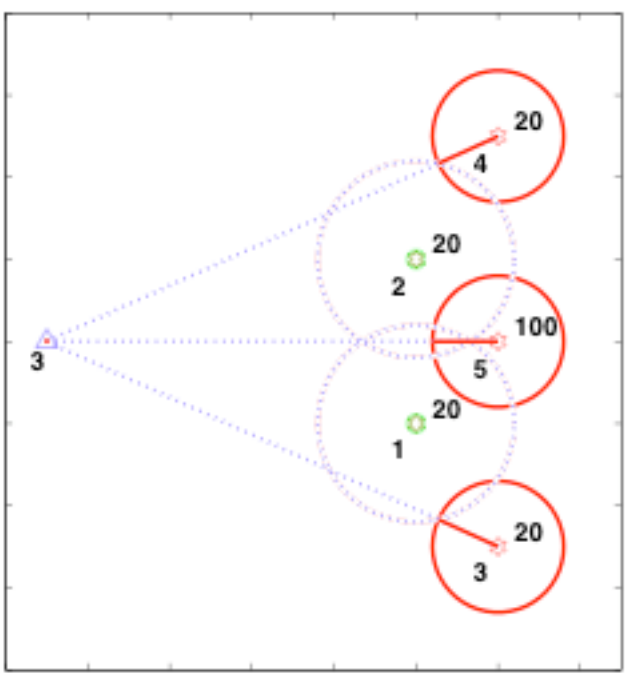
$$
\begin{aligned}
&(x_5 \vee x_1 \vee x_2) \wedge (x_7 \vee x_2 \vee \bar{x}_4) \wedge (x_5 \vee x_2 \vee \bar{x}_8) \wedge (\bar{x}_6 \vee \bar{x}_1 \vee \bar{x}_7) \\
\wedge\ &(x_1 \vee x_3 \vee x_5) \wedge (\bar{x}_7 \vee x_1 \vee \bar{x}_6) \wedge (x_8 \vee x_5 \vee x_7) \wedge (\bar{x}_4 \vee \bar{x}_6 \vee \bar{x}_7) \\
\wedge\ &(\bar{x}_7 \vee x_2 \vee x_1) \wedge (\bar{x}_6 \vee x_4 \vee \bar{x}_4) \wedge (\bar{x}_2 \vee x_3 \vee \bar{x}_2) \wedge (x_4 \vee x_2 \vee \bar{x}_1) \\
\wedge\ &(x_1 \vee \bar{x}_6 \vee x_6) \wedge (x_7 \vee \bar{x}_8 \vee \bar{x}_3) \wedge (x_3 \vee \bar{x}_4 \vee x_4) \wedge (\bar{x}_4 \vee x_7 \vee \bar{x}_3) \\
\wedge\ &(x_2 \vee x_4 \vee x_1) \wedge (\bar{x}_6 \vee \bar{x}_7 \vee x_5) \wedge (\bar{x}_2 \vee x_7 \vee \bar{x}_4) \wedge (\bar{x}_5 \vee x_6 \vee x_3) \\
\wedge\ &(x_7 \vee \bar{x}_1 \vee x_6) \wedge (x_7 \vee x_4 \vee x_7) \wedge (\bar{x}_5 \vee \bar{x}_6 \vee x_5) \wedge (x_7 \vee x_8 \vee \bar{x}_1) \\
\wedge\ &(\bar{x}_1 \vee \bar{x}_1 \vee x_3) \wedge (\bar{x}_8 \vee x_3 \vee \bar{x}_3) \wedge (x_5 \vee x_4 \vee \bar{x}_6) \wedge (x_4 \vee \bar{x}_1 \vee x_4) \\
\wedge\ &(\bar{x}_8 \vee x_4 \vee x_4) \wedge (\bar{x}_4 \vee \bar{x}_4 \vee \bar{x}_1) \wedge (\bar{x}_8 \vee x_7 \vee x_7) \wedge (\bar{x}_2 \vee x_8 \vee \bar{x}_8) \\
&\wedge (x_1 \vee x_2 \vee x_6) \wedge (\bar{x}_5 \vee \bar{x}_2 \vee x_1)
\end{aligned}
$$

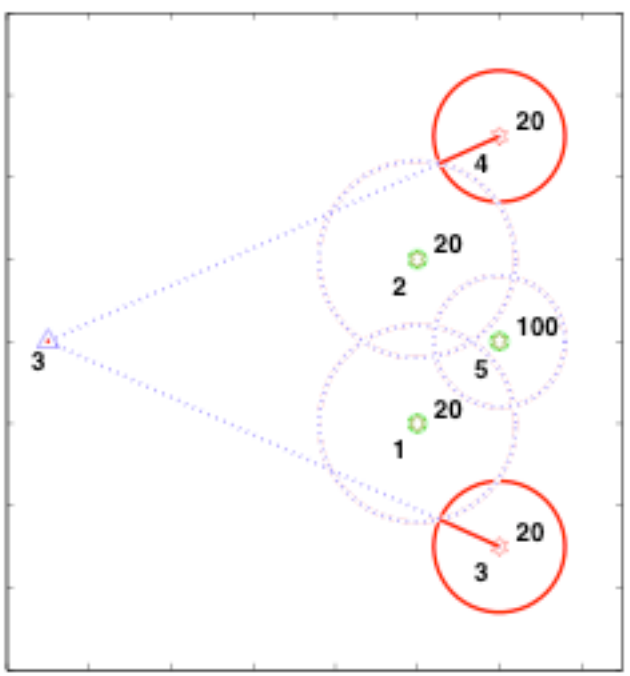
Branch & bound tree

Example: task allocation

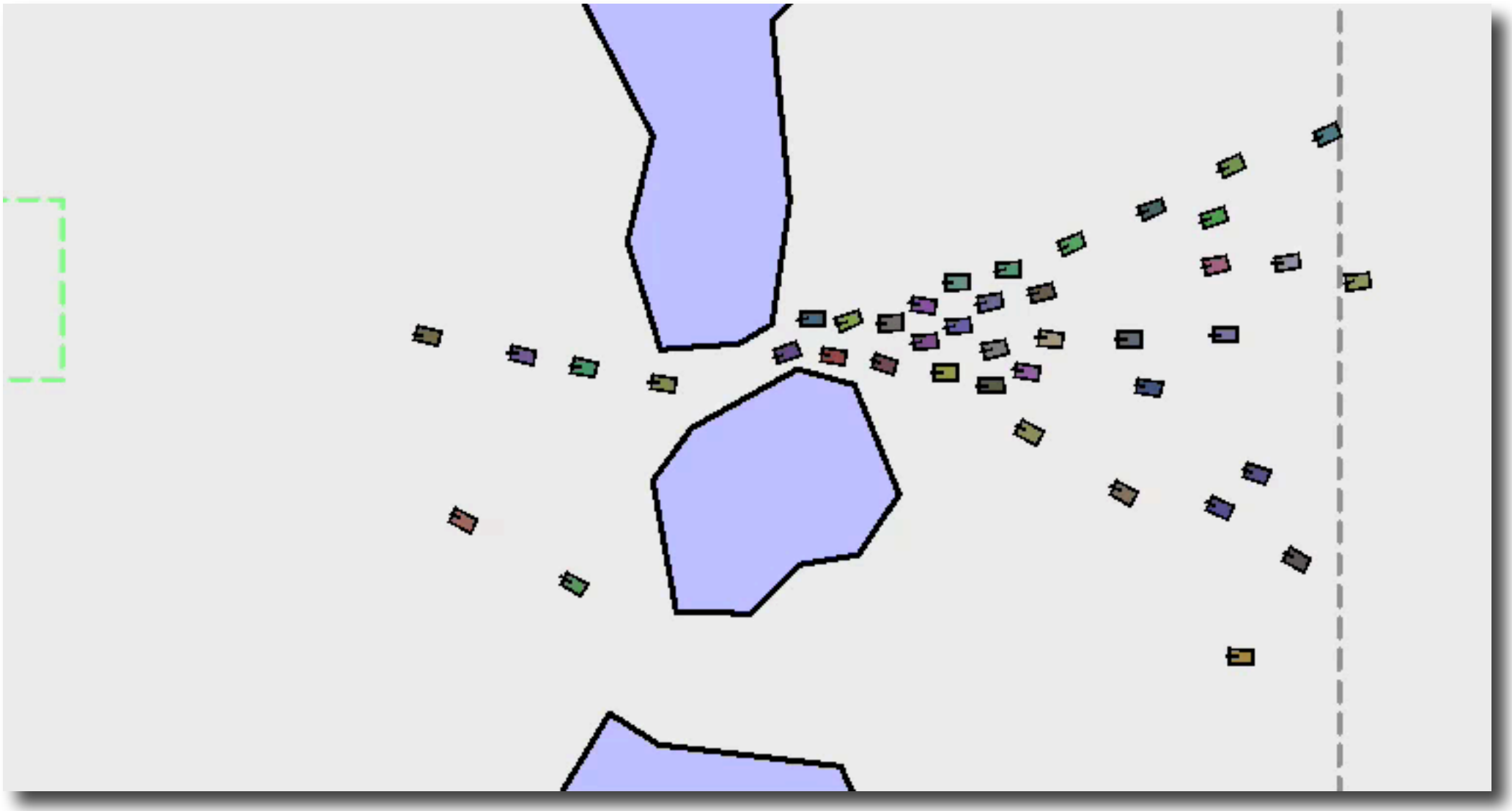




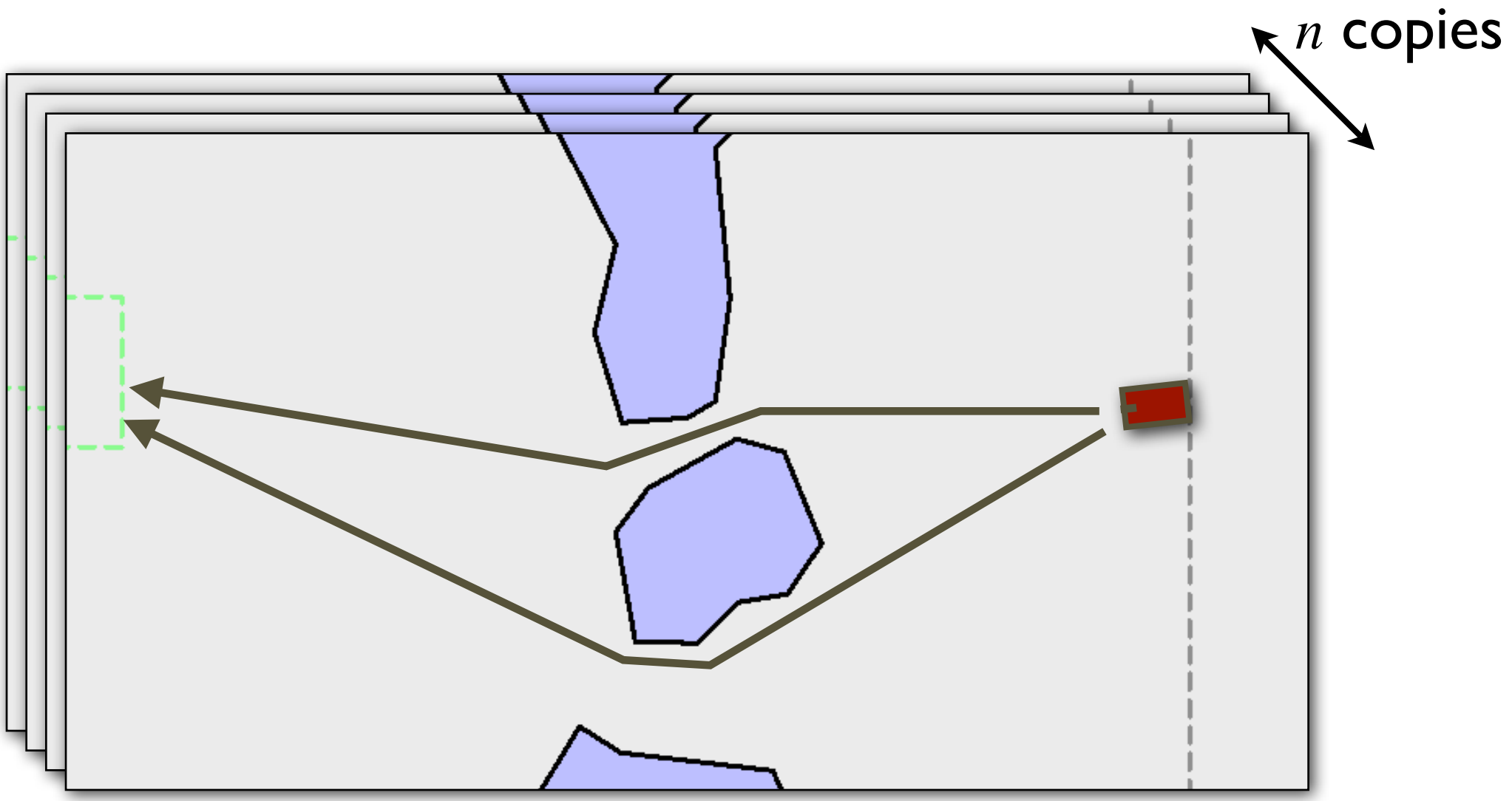




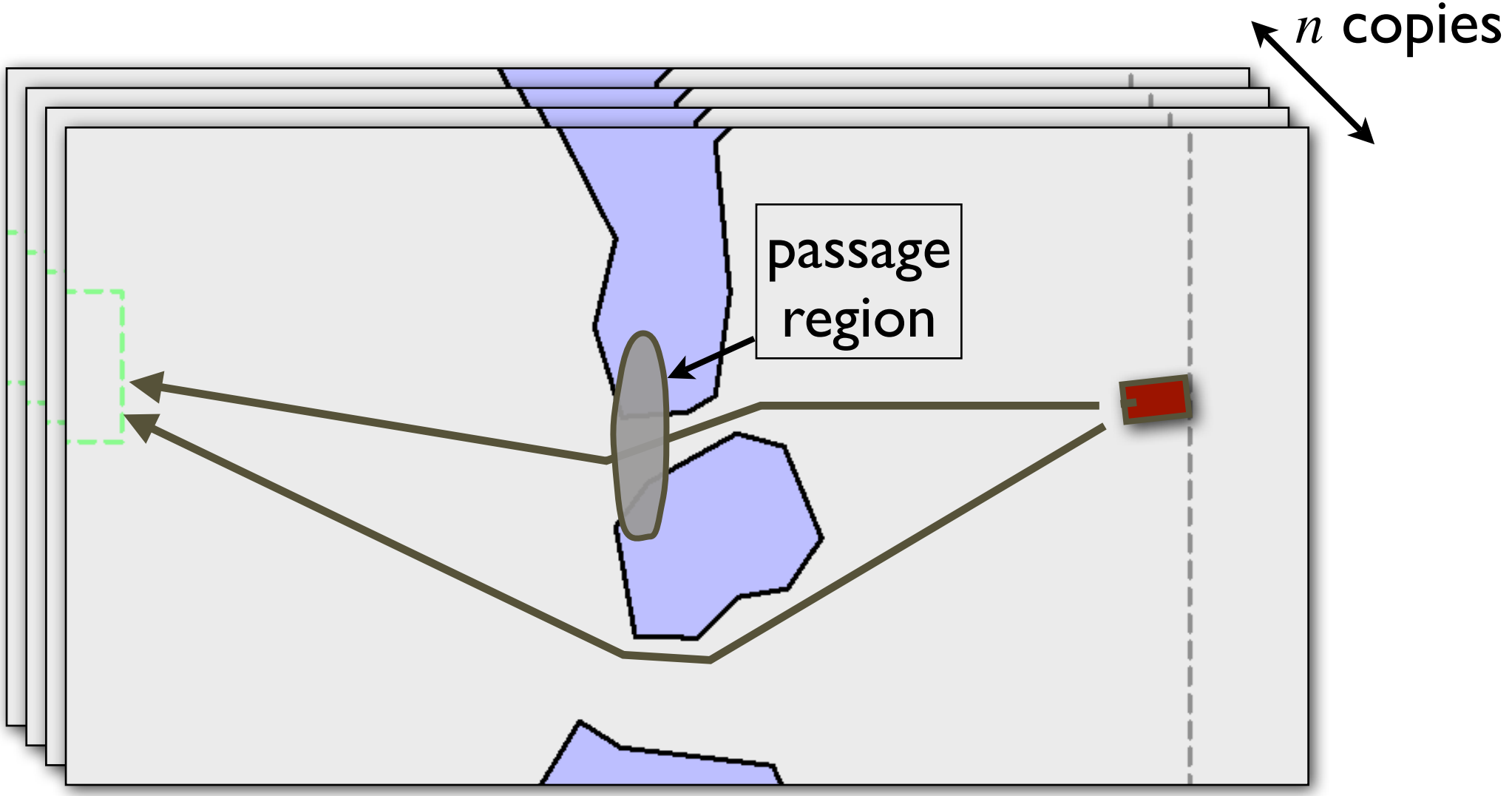
Example: resource allocation



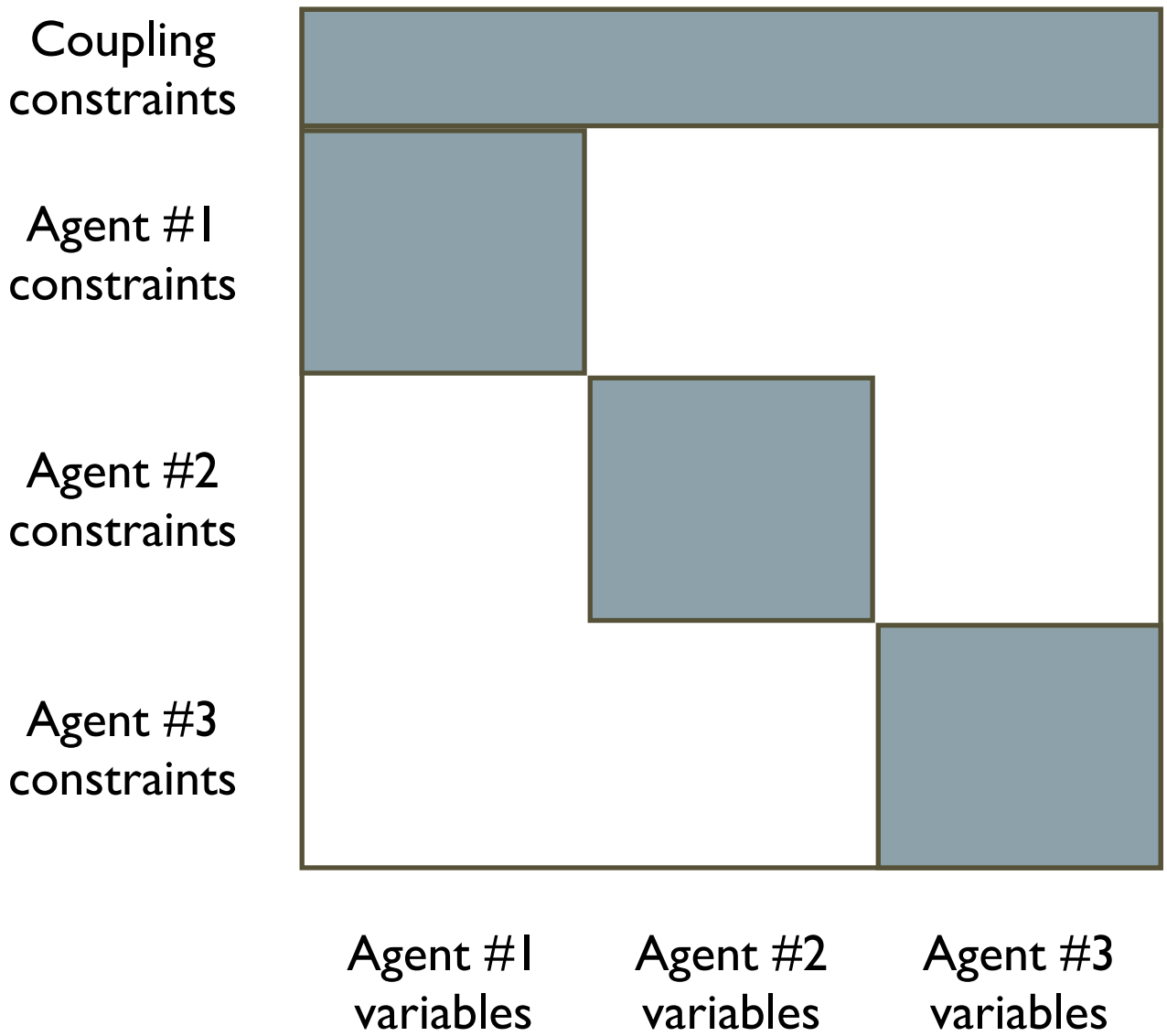
Problem setup



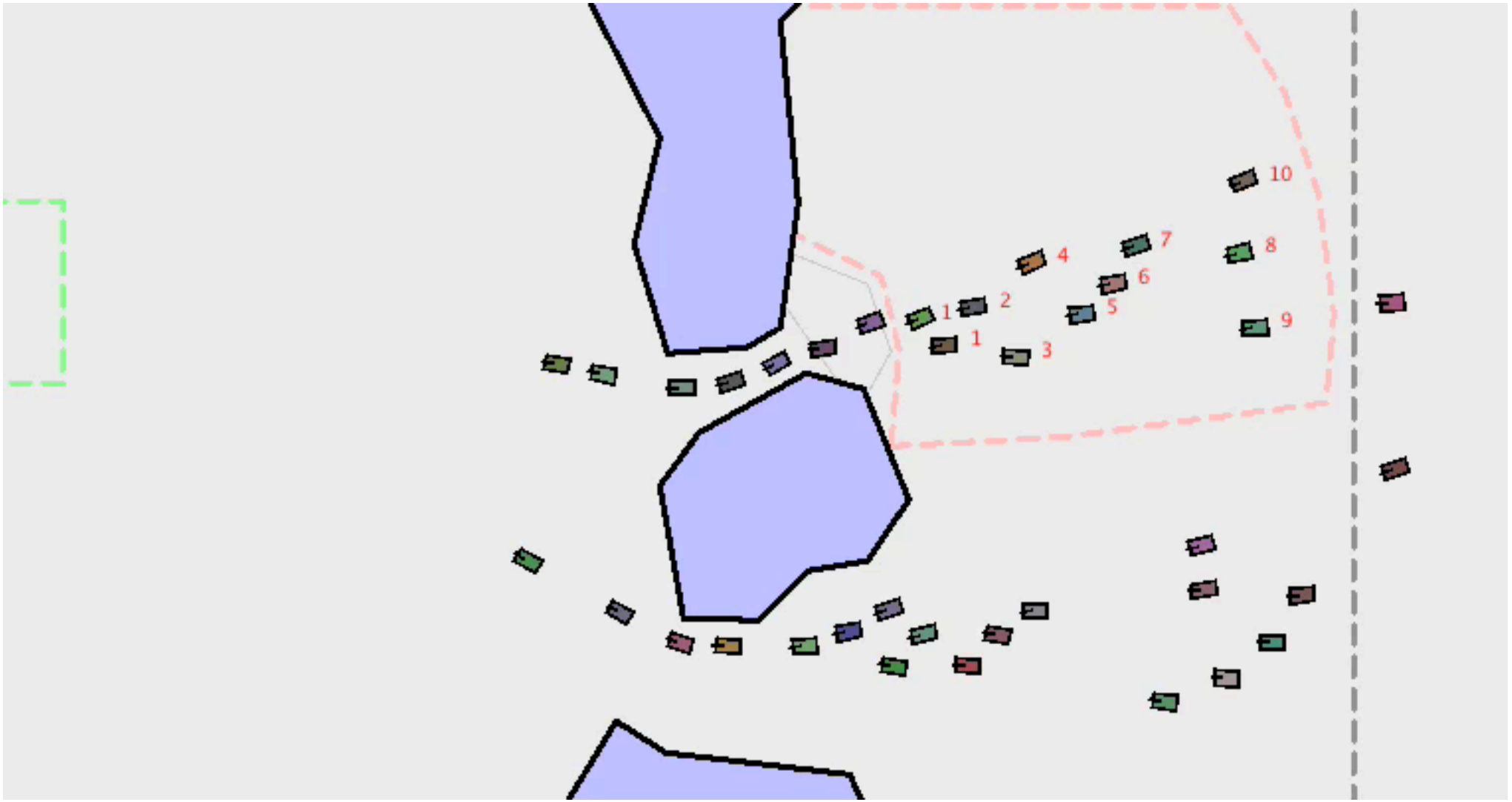
Problem setup



Problem structure



Smart cars



Bound: better than 93.7% of optimal