Faster Stackelberg Planning via Symbolic Search and Information Sharing

Álvaro Torralba, Patrick Speicher, Robert Künnemann,
Marcel Steinmetz, Jörg Hoffmann

AALBORG UNIVERSITET

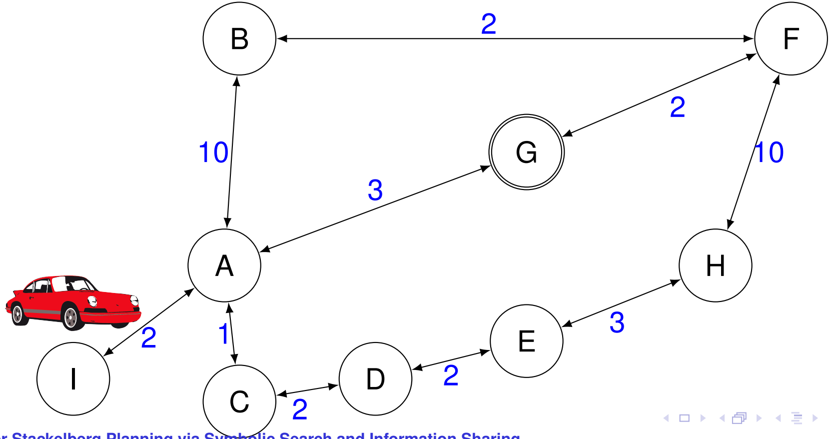
CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

UNIVERSITÄT DES SAARLANDES

Outline

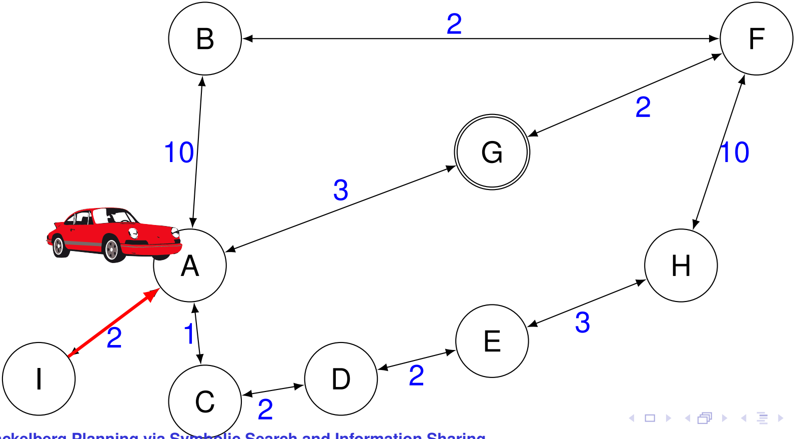
Classical Planning

- Optimal Planning: Find a minimum-cost plan to the goal



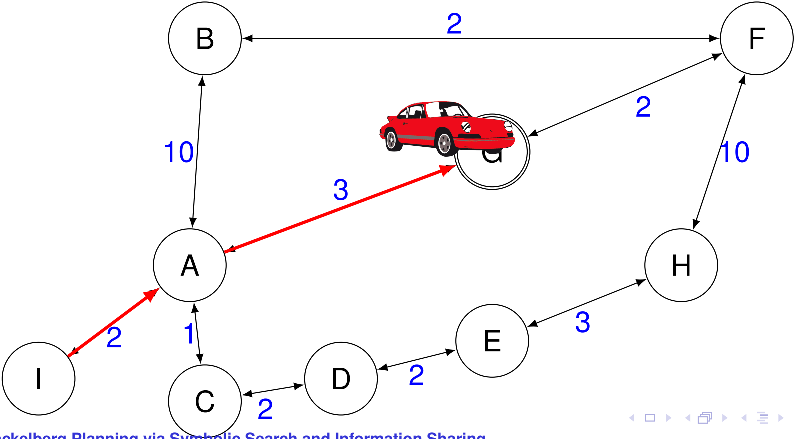
Classical Planning

- Optimal Planning: Find a minimum-cost plan to the goal



Classical Planning

- Optimal Planning: Find a minimum-cost plan to the goal



Classical Planning

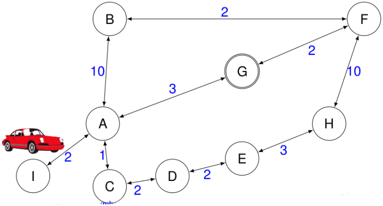
Definition. *A planning task is a 4-tuple* $\Pi = (\mathcal{V}, \mathcal{A}, \mathcal{I}, \mathcal{G})$ *where:*

- $\mathcal{V}$ *is a set of state variables, each* $v \in \mathcal{V}$ *with a finite domain* D_v.
- $\mathcal{A}$ *is a set of actions; each* $a \in \mathcal{A}$ *is a triple* (pre_a, eff_a, c_a), *of precondition and effect (partial assignments), and the action's cost* $c_a \in \mathbb{R}_0^+$.
- *Initial state* $\mathcal{I}$ *(complete assignment), goal* $\mathcal{G}$ *(partial assignment).*

$\rightarrow$ Solution ("Plan"): Action sequence mapping $\mathcal{I}$ into s s.t. $s \models \mathcal{G}$.

Running Example:

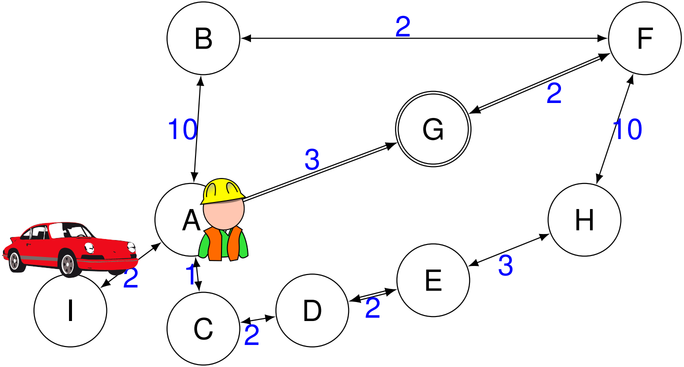
- $\mathcal{V} = \{c\}$
  with $D_c = \{A, B, C, D, E, F, G, H, I\}$.
- $\mathcal{A} = \{drive(x, x')\}$
- $\mathcal{I} = \{c \mapsto I\}$ and $\mathcal{G} = \{c \mapsto G\}$



Stackelberg Planning

Special case of two-player (leader and follower) game where:

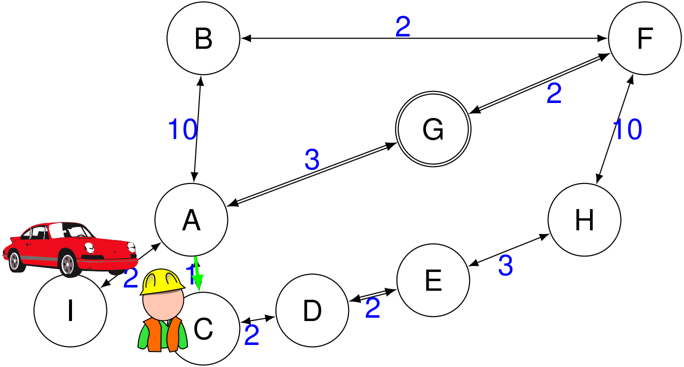
- Leader acts first executing a sequence of actions
- Afterwards, the follower plans to achieve its goal



Stackelberg Planning

Special case of two-player (leader and follower) game where:

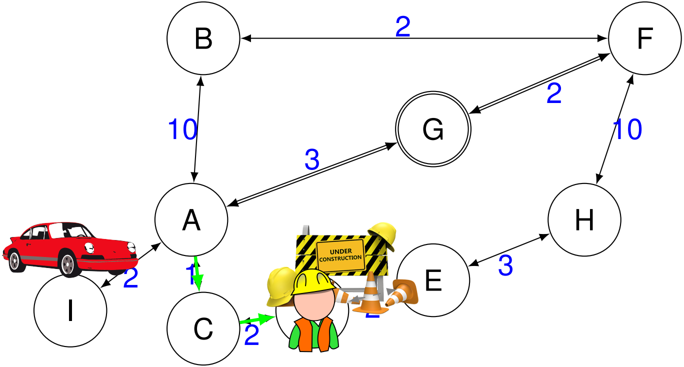
- Leader acts first executing a sequence of actions
- Afterwards, the follower plans to achieve its goal



Stackelberg Planning

Special case of two-player (leader and follower) game where:

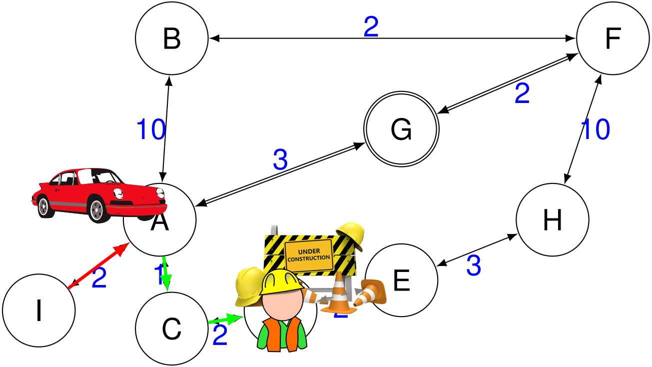
- Leader acts first executing a sequence of actions
- Afterwards, the follower plans to achieve its goal



Stackelberg Planning

Special case of two-player (leader and follower) game where:

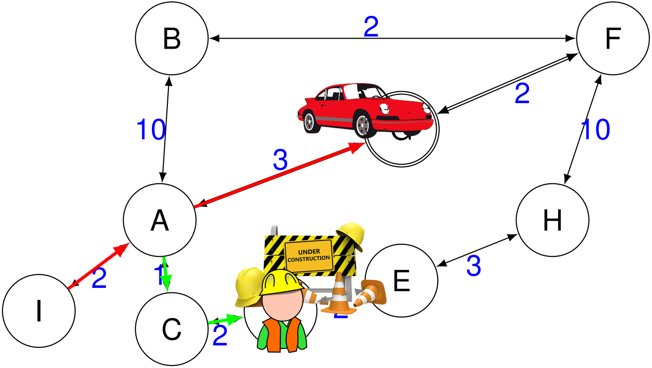
- Leader acts first executing a sequence of actions
- Afterwards, the follower plans to achieve its goal



Stackelberg Planning

Special case of two-player (leader and follower) game where:

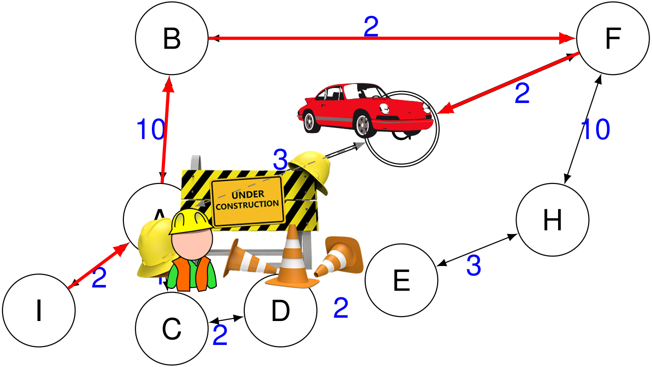
- Leader acts first executing a sequence of actions
- Afterwards, the follower plans to achieve its goal



Stackelberg Planning

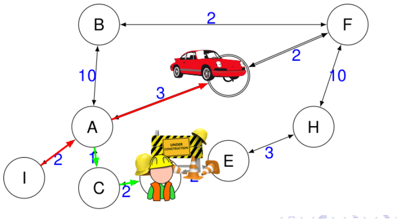
Special case of two-player (leader and follower) game where:

- Leader acts first executing a sequence of actions
- Afterwards, the follower plans to achieve its goal

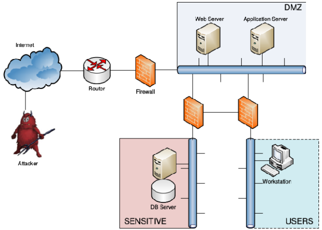


Applications

Robustness Analysis



Pentesting



- Follower: How to achieve the system's goal
- Leader: What could go wrong

- Follower: Attack the network to access sensible information
- Leader: How to defend the network

Stackelberg Planning

Definition. *A* Stackelberg planning task *is a 4-tuple* $\Pi = (\mathcal{V}, \mathcal{A}^L, \mathcal{A}^F, \mathcal{I}, \mathcal{G})$ *where:*

- V *is a set of state variables, each* $v \in V$ *with a finite domain* D_v.
- $\mathcal{A}^L$, $\mathcal{A}^F$ *are sets of actions; each* $a \in \mathcal{A}^L \cup \mathcal{A}^F$ *is a triple* (pre_a, eff_a, c_a), *of precondition and effect (partial assignments), and the action's cost* $c_a \in \mathbb{R}_0^+$.
- *Initial state* I *(complete assignment), goal* G *(partial assignment).*

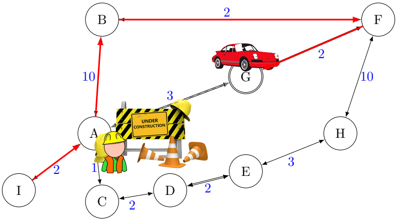
Stackelberg Planning

Definition. *A* Stackelberg planning task *is a 4-tuple* $\Pi = (\mathcal{V}, \mathcal{A}^L, \mathcal{A}^F, \mathcal{I}, \mathcal{G})$ *where:*

- V *is a set of state variables, each* $v \in V$ *with a finite domain* D_v.
- $\mathcal{A}^L$, $\mathcal{A}^F$ *are sets of actions; each* $a \in \mathcal{A}^L \cup \mathcal{A}^F$ *is a triple* (pre_a, eff_a, c_a), *of precondition and effect (partial assignments), and the action's cost* $c_a \in \mathbb{R}_0^+$.
- *Initial state* I *(complete assignment), goal* G *(partial assignment).*

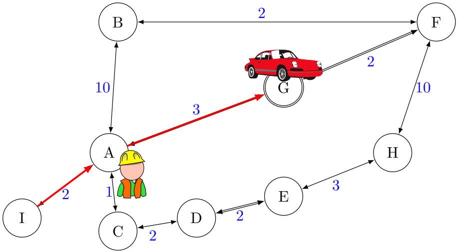
Running Example:

- $\mathcal{V} = \{c, w, r_{AG}, r_{FG}, r_{DE}\}$, $D_c = D_w = \{A, \ldots, I\}$, $D_{r_{ij}} = \{available, blocked\}$.
- $\mathcal{A}^F = \{drive_c(x, x')\}$, $\mathcal{A}^L = \{walk(x, x'), block(x, x')\}$
- $\mathcal{I} = \{c \mapsto I, w \mapsto A, r_{ij} \mapsto available\}$
- $\mathcal{G} = \{c \mapsto G\}$



Solution of a Stackelberg Planning Task

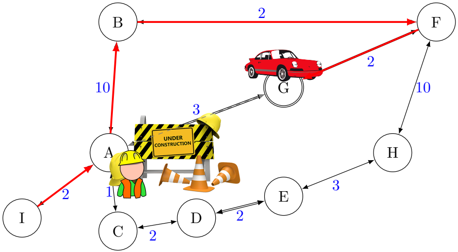
→Find a set of leader plans that minimize leader cost and maximize follower's cost



Leader Plan	c(L)	c(F)
⟨⟩	0	5

Solution of a Stackelberg Planning Task

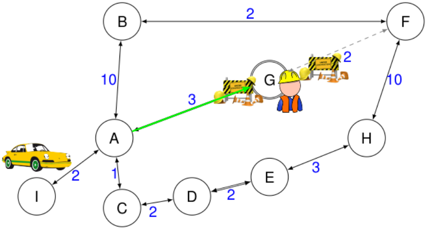
→Find a set of leader plans that minimize leader cost and maximize follower's cost



Leader Plan	c(L)	c(F)
⟨⟩	0	5
⟨block(A,G)⟩	1	16

Solution of a Stackelberg Planning Task

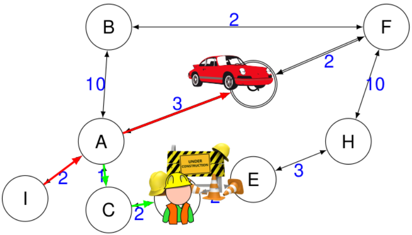
→Find a set of leader plans that minimize leader cost and maximize follower's cost



Leader Plan	c(L)	c(F)
⟨⟩	0	5
⟨block(A,G)⟩	1	16
⟨walk(A,C), walk(C,D), block(D,E)⟩	3	5

Solution of a Stackelberg Planning Task

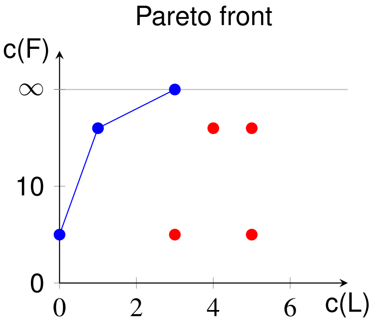
→Find a set of leader plans that minimize leader cost and maximize follower's cost



Leader Plan	c(L)	c(F)
⟨⟩	0	5
⟨block(A,G)⟩	1	16
⟨walk(A,C), walk(C,D), block(D,E)⟩	3	5
⟨walk(A,G), block(G,A), block(G,F)⟩	3	∞

Solution of a Stackelberg Planning Task

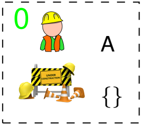
→Find a set of leader plans that minimize leader cost and maximize follower's cost



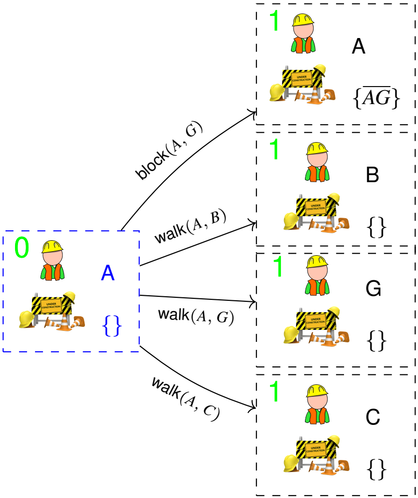
Leader Plan	c(L)	c(F)
⟨⟩	0	5
⟨block(A,G)⟩	1	16
⟨walk(A,C), walk(C,D), block(D,E)⟩	3	5
⟨walk(A,G), block(G,A), block(G,F)⟩	3	∞
...	...	...

Outline

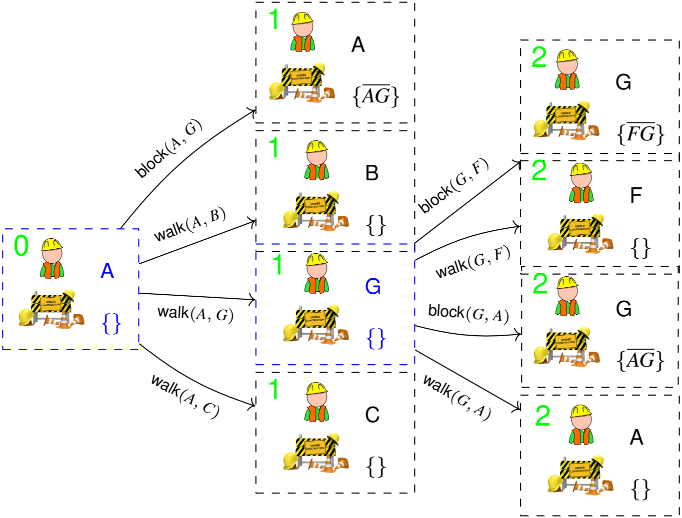
Leader Search



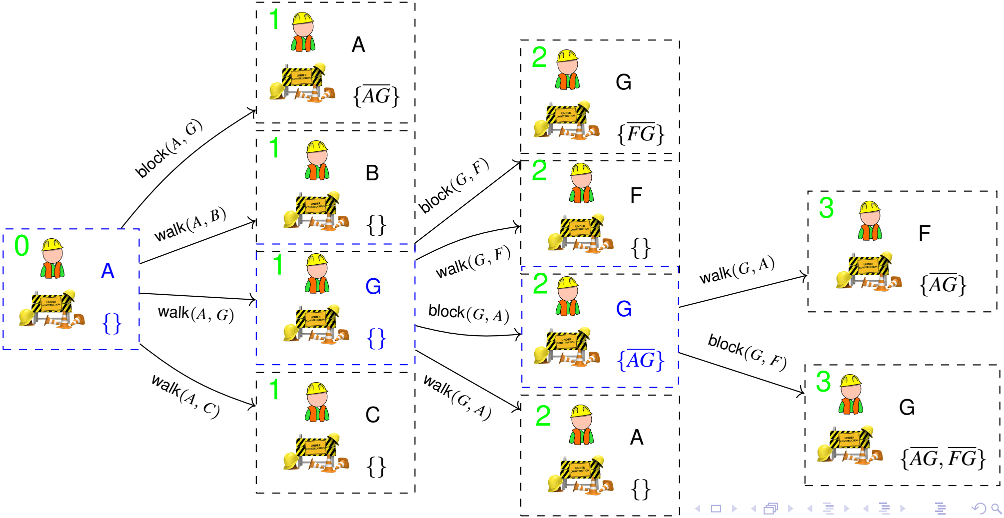
Leader Search



Leader Search



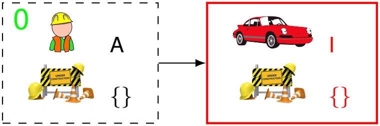
Leader Search



Follower Subtasks

Leader Search Node

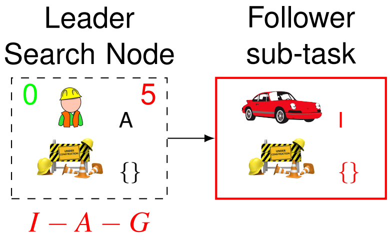
Follower sub-task

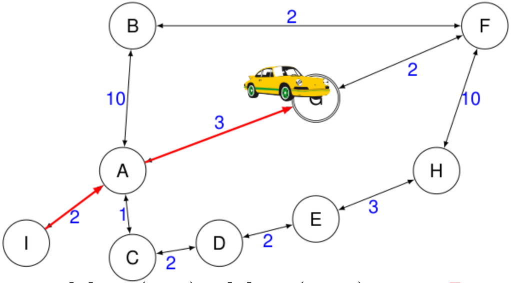


Follower subtask:

- Classical Planning Task →set of actions depends on blocked roads
- Optimal planners:
  - A^* with LM-cut
  - Symbolic Bidirectional Search

Follower Subtasks



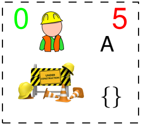


drive (I, A), **drive** (A, G) cost=5

Follower subtask:

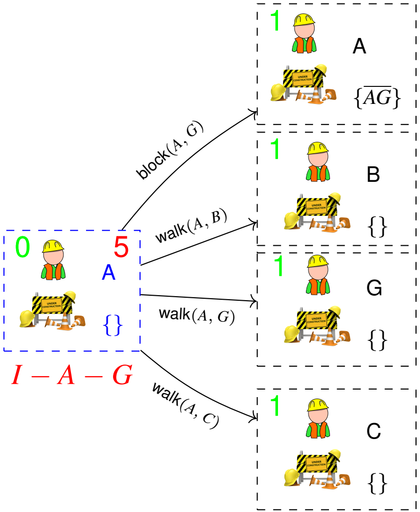
- Classical Planning Task →set of actions depends on blocked roads
- Optimal planners:
  - A^* with LM-cut
  - Symbolic Bidirectional Search

Leader Search (IDS)

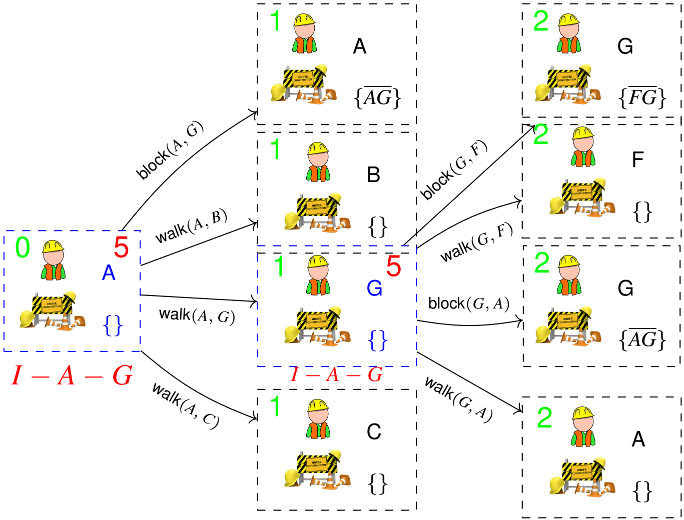


$I - A - G$

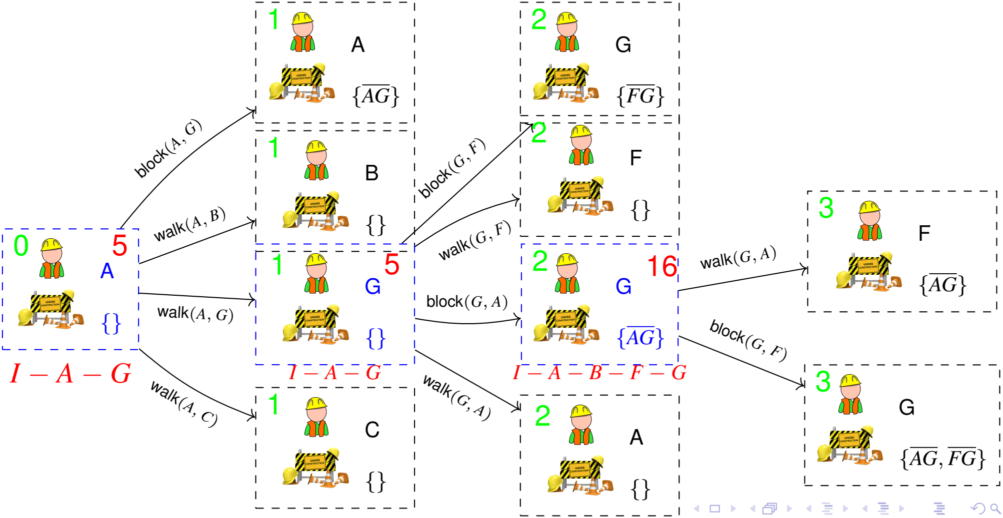
Leader Search (IDS)



Leader Search (IDS)

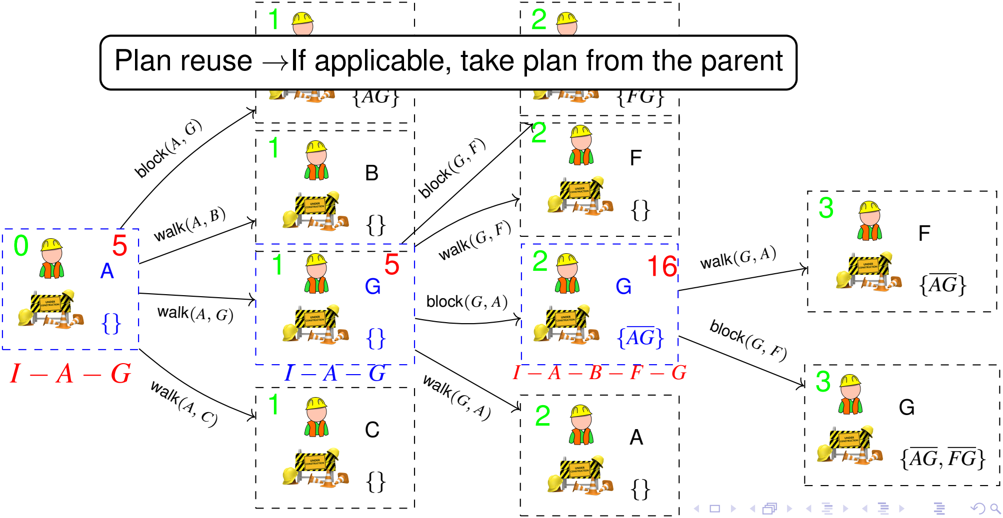


Leader Search (IDS)



Leader Search (IDS)

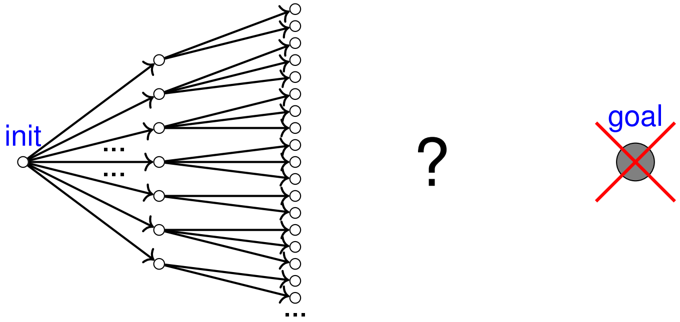
Plan reuse →If applicable, take plan from the parent



Outline

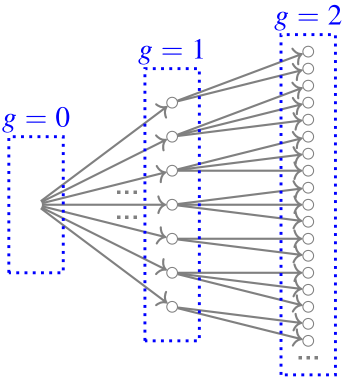
Symbolic Leader Search

Observation 1: Leader search is exhaustive → state space *explosion*

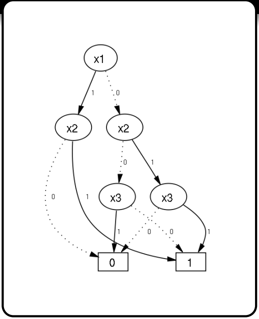


Symbolic Leader Search

Observation 1: Leader search is exhaustive → state space *explosion*



BDDs to the rescue!



Cost-bounded search

Observation 2: We do not always need to compute optimal solutions to follower sub-tasks

Global bounds:

- Leader cost L: explore leader space in ascending leader cost
- Follower cost F: the highest cost of any follower sub-problem so far

$\rightarrow$any new entry in the pareto front will have a cost greater than F

Cost-bounded search

Observation 2: We do not always need to compute optimal solutions to follower sub-tasks

Global bounds:

- Leader cost L: explore leader space in ascending leader cost
- Follower cost F: the highest cost of any follower sub-problem so far

$\rightarrow$any new entry in the pareto front will have a cost greater than F

Cost-bounded/Optimal Planning: Given a follower sub-task and a cost bound F, return any plan of cost $C \leq F$ if one exists, otherwise return an optimal plan if one exists, otherwise return “unsolvable”.

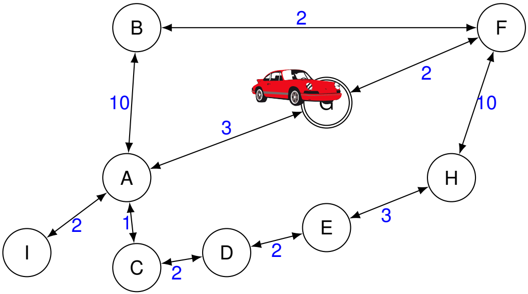
Information Sharing

Observation 3: We need to find optimal solutions for many follower sub-tasks, but they are very similar

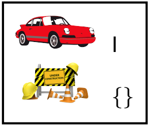
- Same set of variables $\mathcal{V}$ and goal $\mathcal{G}$
- Actions $A \subseteq \mathcal{A}^F$

$\rightarrow$How to re-use information among follower sub-searches?

Information Sharing: Goal Regression



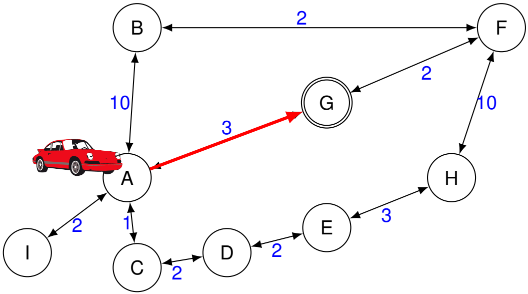
Whenever we compute a new plan for some follower sub-task, use regression from the goal:



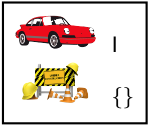
drive (I, A), **drive** (A, G)

$\{c \mapsto G\}$ cost = 0

Information Sharing: Goal Regression



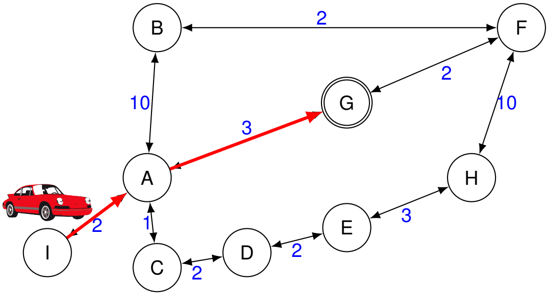
Whenever we compute a new plan for some follower sub-task, use regression from the goal:



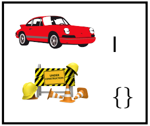
$\{c \mapsto G\}$ cost = 0

drive (I, A), **drive** (A, G) $\{c \mapsto A, r_{AG} \mapsto \mathit{available}\}$ cost = 3

Information Sharing: Goal Regression



Whenever we compute a new plan for some follower sub-task, use regression from the goal:



drive (I, A), **drive** (A, G)

$\{c \mapsto G\}$ cost = 0

$\{c \mapsto A, r_{AG} \mapsto available\}$ cost = 3

$\{c \mapsto I, r_{AG} \mapsto available\}$ cost = 5

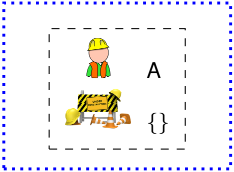
Symbolic Leader Search: Sharing at the Leader Level

L=0 F=0

Pareto Front

c(L)	c(F)

Solved Follower Tasks



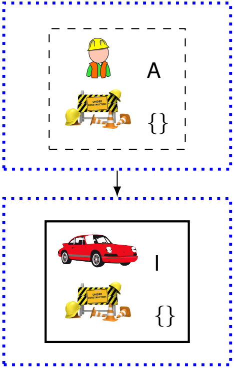
Symbolic Leader Search: Sharing at the Leader Level

L=0 F=0

Pareto Front

c(L)	c(F)

Solved Follower Tasks

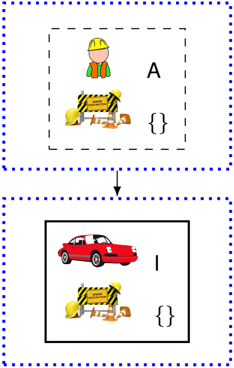


Symbolic Leader Search: Sharing at the Leader Level

L=0 F=0

Pareto Front	
c(L)	c(F)

Solved Follower Tasks



$I - A - G, c = 5$

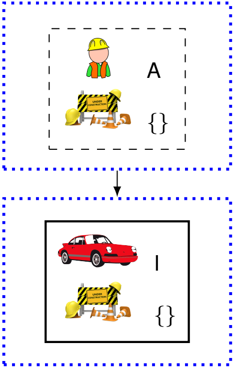
Symbolic Leader Search: Sharing at the Leader Level

L=0 F=5

Pareto Front	
c(L)	c(F)

Solved Follower Tasks

$\{c \mapsto I, r_{AG} \mapsto available\}$



$I - A - G, c = 5$

Symbolic Leader Search: Sharing at the Leader Level

L=0 F=5

Pareto Front	
c(L)	c(F)

Solved Follower Tasks

$\{c \mapsto I, r_{AG} \mapsto available\}$

A

{}

Symbolic Leader Search: Sharing at the Leader Level

L=0 F=5

Pareto Front

c(L)	c(F)
0	5

Solved Follower Tasks

$$\{c \mapsto I, r_{AG} \mapsto available\}$$

A

{}

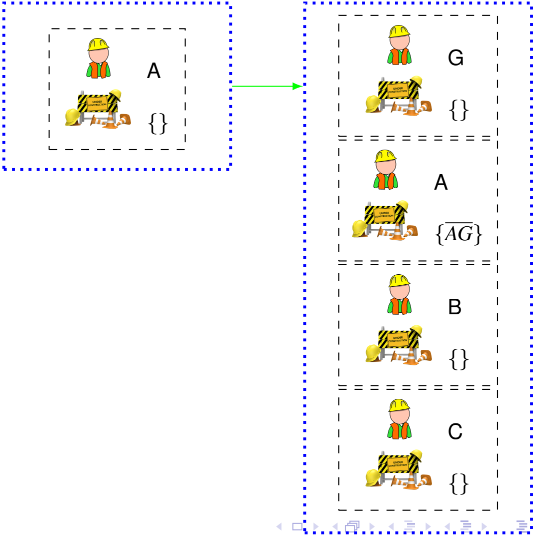
Symbolic Leader Search: Sharing at the Leader Level

L=1 F=5

Pareto Front	
c(L)	c(F)
0	5

Solved Follower Tasks

$\{c \mapsto I, r_{AG} \mapsto available\}$



Symbolic Leader Search: Sharing at the Leader Level

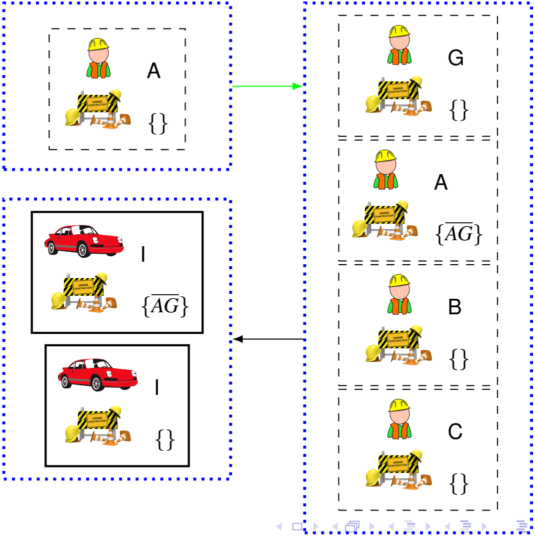
L=1 F=5

Pareto Front

c(L)	c(F)
0	5

Solved Follower Tasks

$$\{c \mapsto I, r_{AG} \mapsto available\}$$



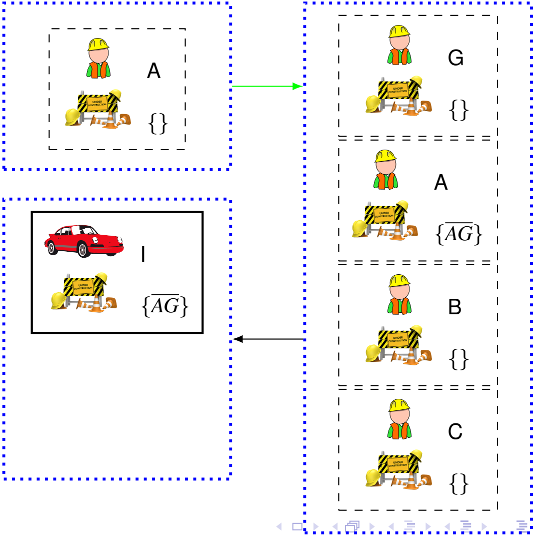
Symbolic Leader Search: Sharing at the Leader Level

L=1 F=5

Pareto Front	
c(L)	c(F)
0	5

Solved Follower Tasks

$$\{c \mapsto I, r_{AG} \mapsto available\}$$



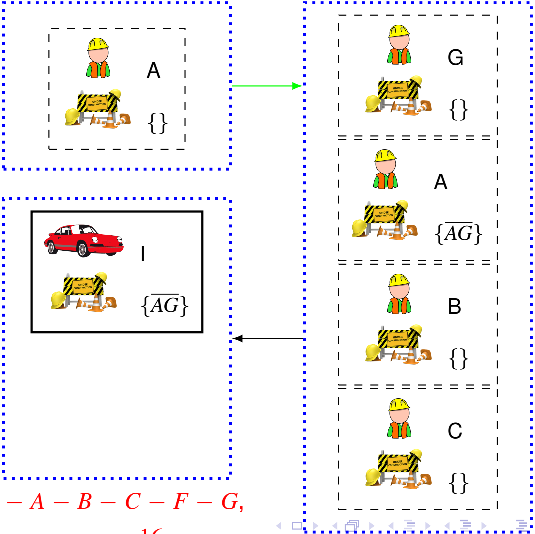
Symbolic Leader Search: Sharing at the Leader Level

L=1 F=5

Pareto Front	
c(L)	c(F)
0	5

Solved Follower Tasks

$$\{c \mapsto I, r_{AG} \mapsto available\}$$



$I - A - B - C - F - G,$

$c = 16$

Symbolic Leader Search: Sharing at the Leader Level

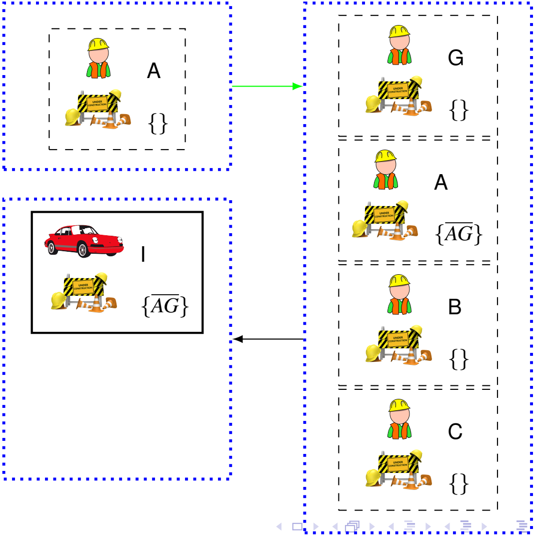
L=1 F=16

Pareto Front

c(L)	c(F)
0	5

Solved Follower Tasks

$$\{c \mapsto I, r_{AG} \mapsto available\}$$
$$\{c \mapsto I, r_{FG} \mapsto available\}$$



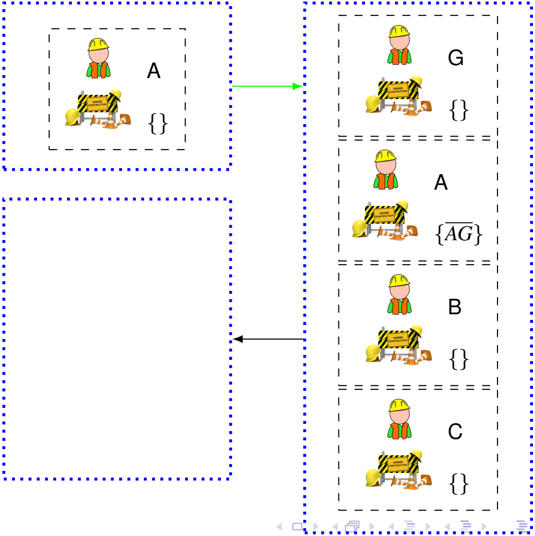
Symbolic Leader Search: Sharing at the Leader Level

L=1 F=16

Pareto Front	
c(L)	c(F)
0	5
1	16

Solved Follower Tasks

$$\{c \mapsto I, r_{AG} \mapsto available\}$$
$$\{c \mapsto I, r_{FG} \mapsto available\}$$



Sharing at the Follower Level

Upper Bound Function: pre-store plans for certain follower states

Sharing at the Follower Level

Upper Bound Function: pre-store plans for certain follower states

How to obtain upper bound functions?

From plans:

$$\{c \mapsto G\} \quad \text{ub} = 0$$
$$\{c \mapsto A, r_{AG} \mapsto \mathit{available}\} \quad \text{ub} = 3$$
$$\{c \mapsto I, r_{AG} \mapsto \mathit{available}\} \quad \text{ub} = 5$$

Sharing at the Follower Level

Upper Bound Function: pre-store plans for certain follower states

How to obtain upper bound functions?

From plans:

$\{c \mapsto G\}$ ub = 0

$\{c \mapsto A, r_{AG} \mapsto \mathit{available}\}$ ub = 3

$\{c \mapsto I, r_{AG} \mapsto \mathit{available}\}$ ub = 5

How to use upper bound functions?

Transform any search-based optimal subsolver in a cost-bounded algorithm:
If some follower state s is seen such that $g(s) + ub(s) \leq F$ stop immediately!

Sharing at the Follower Level

Upper Bound Function: pre-store plans for certain follower states

How to obtain upper bound functions?

From plans:

$\{c \mapsto G\}$ ub = 0

$\{c \mapsto A, r_{AG} \mapsto available\}$ ub = 3

$\{c \mapsto I, r_{AG} \mapsto available\}$ ub = 5

From backward search on the most constrained follower task (Π^{+}):

G

$\{\overline{AG}, \overline{FG}, \overline{DE}\}$

How to use upper bound functions?

Transform any search-based optimal subsolver in a cost-bounded algorithm:
If some follower state s is seen such that $g(s) + ub(s) \leq F$ stop immediately!

Sharing at the Follower Level

Upper Bound Function: pre-store plans for certain follower states

How to obtain upper bound functions?

From plans:

$\{c \mapsto G\}$	ub = 0
$\{c \mapsto A, r_{AG} \mapsto available\}$	ub = 3
$\{c \mapsto I, r_{AG} \mapsto available\}$	ub = 5

From backward search on the most constrained follower task (Π^{+}):

G

$\{\overline{AG}, \overline{FG}, \overline{DE}\}$

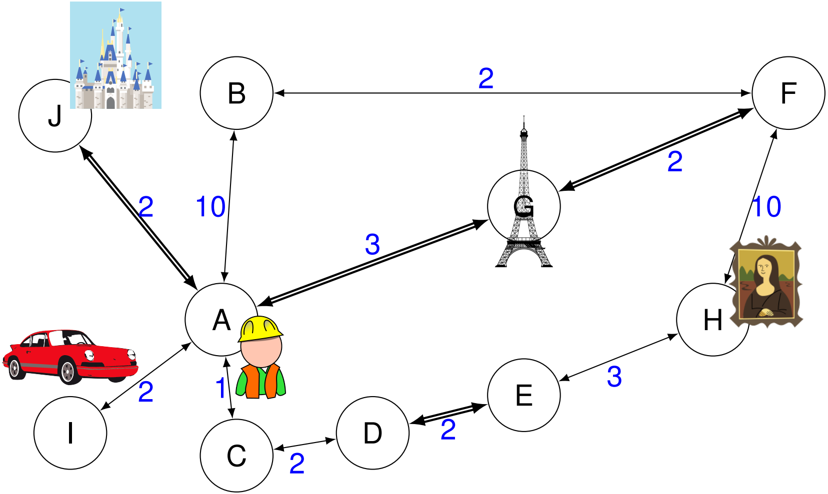
How to use upper bound functions?

Transform any search-based optimal subsolver in a cost-bounded algorithm:
If some follower state s is seen such that $g(s) + ub(s) \leq F$ stop immediately!

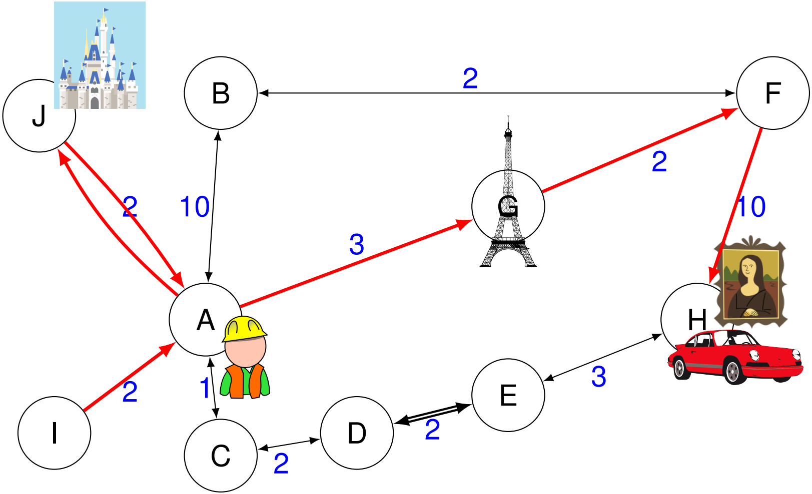
$\rightarrow$In the paper: General conditions under which lower- and upper-bounds can be shared accross follower sub-tasks

Outline

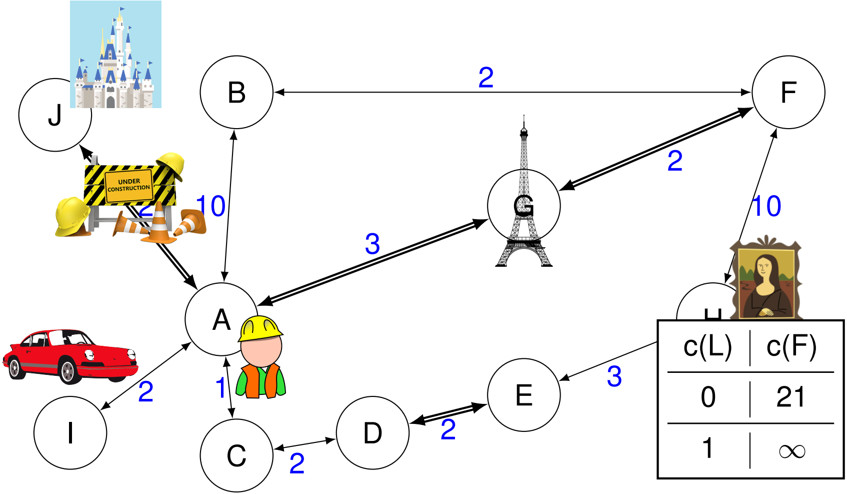
Net-Benefit Stackelberg Planning



Net-Benefit Stackelberg Planning



Net-Benefit Stackelberg Planning



c(L)	c(F)
0	21
1	∞

Net-Benefit Stackelberg Planning

- Soft goals: Each goal has a corresponding cost the follower must pay if it is not achieved

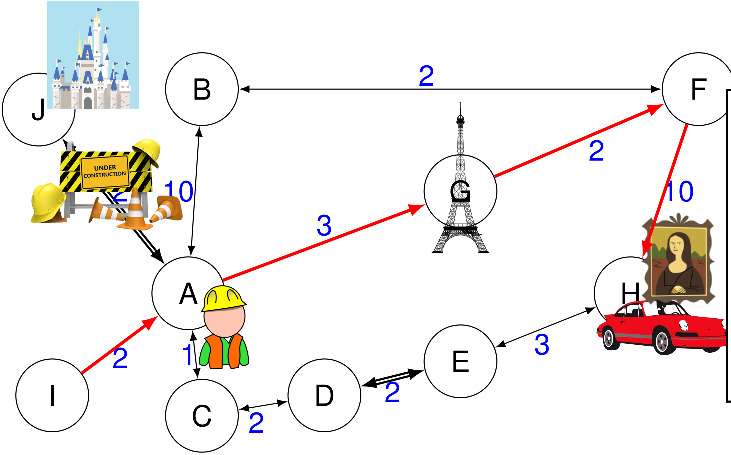
Net-Benefit Stackelberg Planning

- Soft goals: Each goal has a corresponding cost the follower must pay if it is not achieved
- Compilation to classical planning by Keyder and Geffner (2009) →No specialized algorithms are required

Net-Benefit Stackelberg Planning

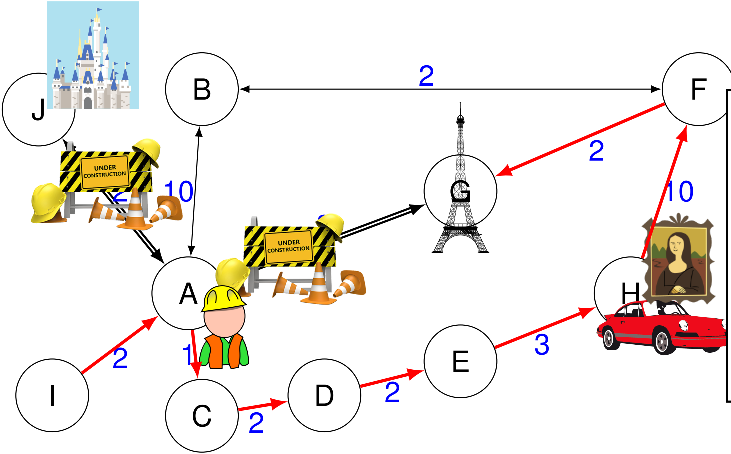
- Soft goals: Each goal has a corresponding cost the follower must pay if it is not achieved
- Compilation to classical planning by Keyder and Geffner (2009) →No specialized algorithms are required
- In our experiments we set a cost of 10000 for each individual sub-goal →follower chooses the cheapest plan among the ones that maximize the number of achieved goals

Net-Benefit Stackelberg Planning



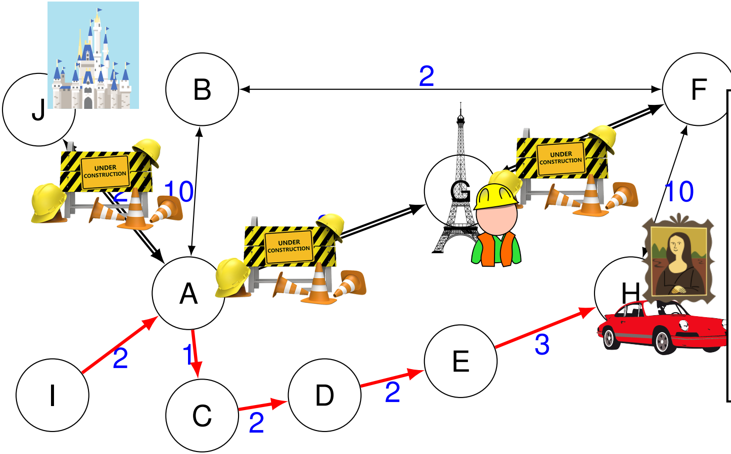
c(L)	c(F)
0	21
1	10017

Net-Benefit Stackelberg Planning



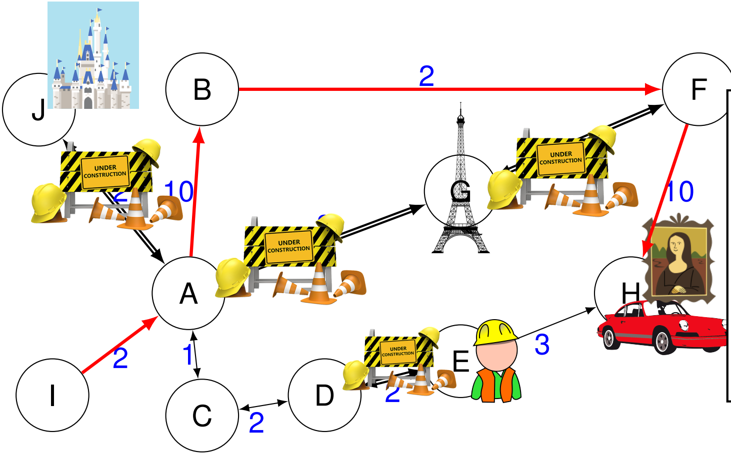
c(L)	c(F)
0	21
1	10017
2	10022

Net-Benefit Stackelberg Planning



c(L)	c(F)
0	21
1	10017
2	10022
4	20010

Net-Benefit Stackelberg Planning



c(L)	c(F)
0	21
1	10017
2	10022
4	20010
8	20024

Outline

Experimental Setup

Configurations:

- IDS (Speicher et al. 2018)
- Symbolic Leader Search
  - –
  - $+ ub$: use upper bound functions from plans
  - $+\Pi^{+}$: use upper bound functions from plans and backward search
  - +FF: use cost-bounded planner (GBFS with FF heuristic for $1s$)

$\rightarrow$Time limit of 30m and memory limit of 4GB.

Benchmarks

We use the benchmarks from previous work (Speicher et al. 2018)

Pareto frontier size ($|PF(\Pi^S)|$):

OLD (Speicher et al. 2018)

Domain	*avg*	*max*
Logistics	1.85	3
Mystery	1.59	3
Rovers	1.86	3
Sokoban	1.92	2
Tpp	2.00	2
Visitall	2.32	7
Pentesting	1.26	2

Benchmarks

We use the benchmarks from previous work (Speicher et al. 2018)

Pareto frontier size ($|PF(\Pi^S)|$):

OLD (Speicher et al. 2018)			NEW			NET	
Domain	*avg*	*max*	Domain	*avg*	*max*	*avg*	*max*
Logistics	1.85	3	Logistics	2.61	6	3.83	16
Mystery	1.59	3	Nomystery	2.58	7	3.25	13
Rovers	1.86	3	Rovers	1.85	4	3.21	13
Sokoban	1.92	2	Transport	4.24	17	3.71	9
Tpp	2.00	2	Tpp	2.15	7	2.35	13
Visitall	2.32	7	Visitall	2.97	7	5.45	34
Pentesting	1.26	2	Pentesting	1.71	4	4.06	13

→New benchmark set for standard and net-benefit planning

Overall Results

Coverage: Tasks solved under 30 minutes and 4GB

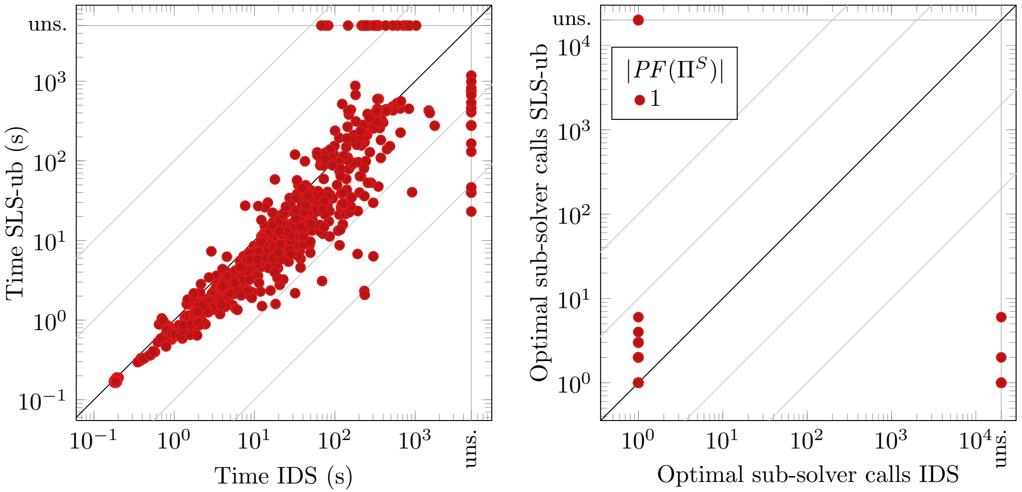
Follower sub-solver		IDS Π^+	**SLS** —
LMcut	OLD (1987)	**681**	630
	NEW (1059)	681	**708**
	NET (1064)	526	**536**
Symbolic Bidirectional	OLD (1987)	584	**621**
	NEW (1059)	632	**819**
	NET (1064)	540	**654**

Overall Results

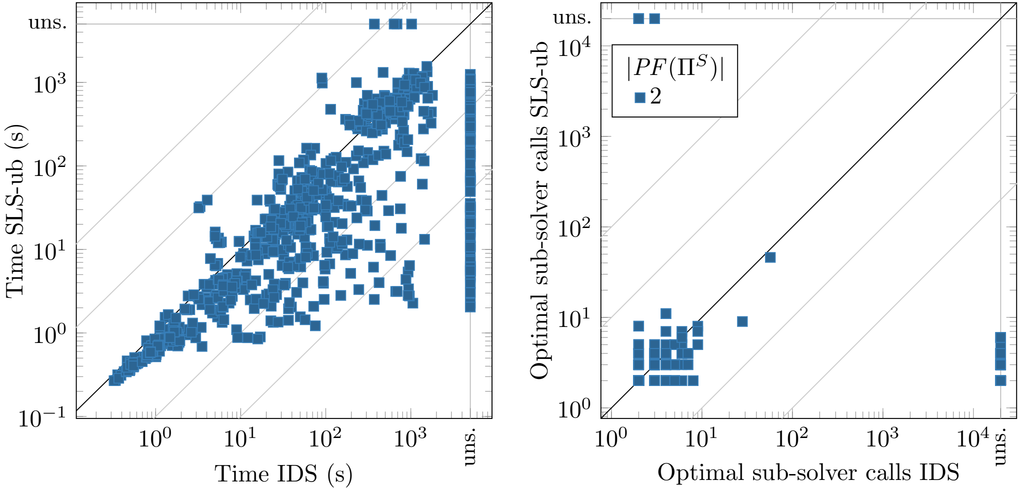
Coverage: Tasks solved under 30 minutes and 4GB

Follower sub-solver		IDS Π^{+}	**SLS** —	**SLS** $+ub$
LMcut	OLD (1987)	**681**	630	634
	NEW (1059)	681	708	**743**
	NET (1064)	526	536	**574**
Symbolic Bidirectional	OLD (1987)	584	621	**628**
	NEW (1059)	632	819	**823**
	NET (1064)	540	654	**671**

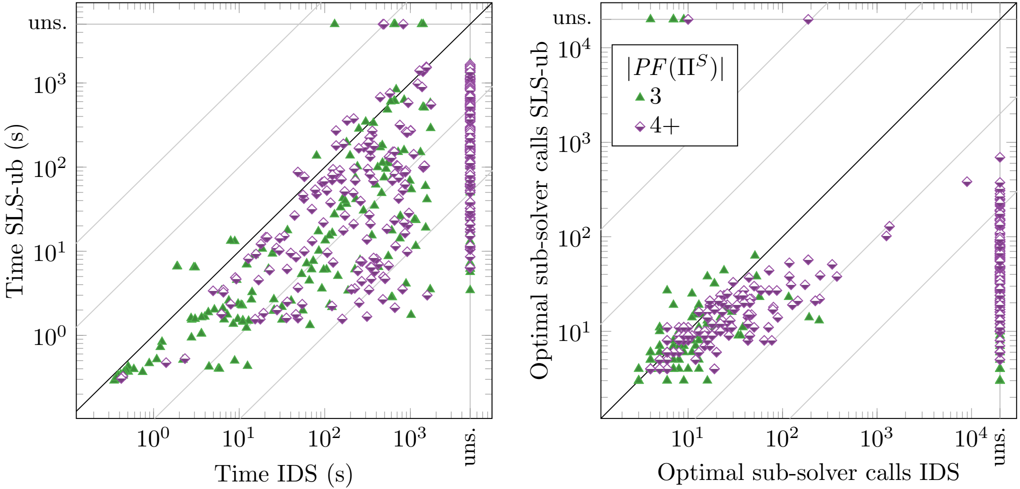
Results: SLS-ub vs IDS



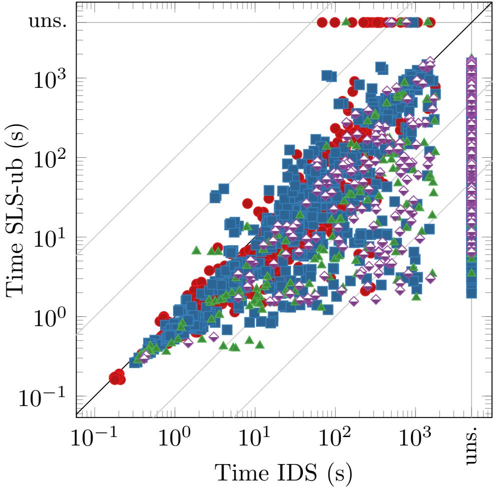
Results: SLS-ub vs IDS

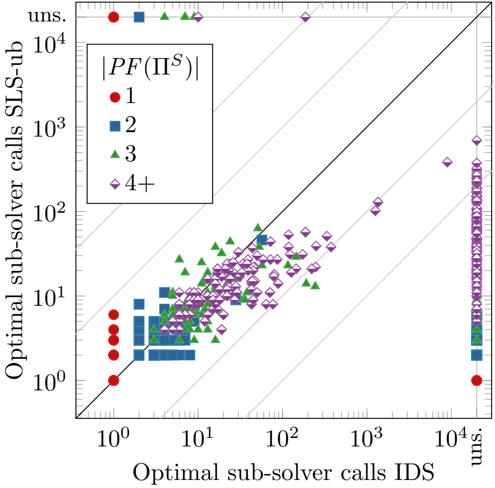


Results: SLS-ub vs IDS



Results: SLS-ub vs IDS





Overall Results

Coverage: Tasks solved under 30 minutes and 4GB

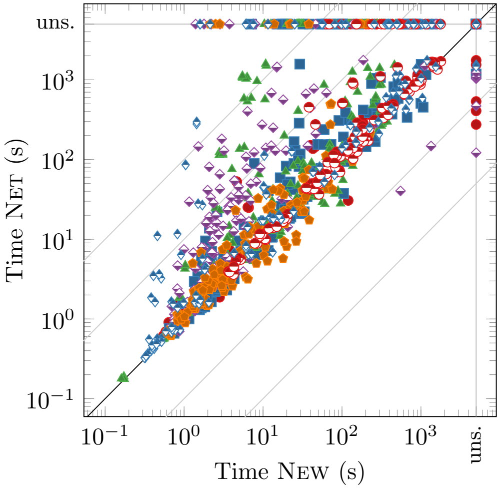
Follower sub-solver		IDS Π^+	SLS —	SLS $+ub$	SLS $+\Pi^+$
LMcut	OLD (1987)	**681**	630	634	652
	NEW (1059)	681	708	**743**	740
	NET (1064)	526	536	574	**613**
Symbolic Bidirectional	OLD (1987)	584	621	628	**652**
	NEW (1059)	632	819	823	**825**
	NET (1064)	540	654	671	**726**

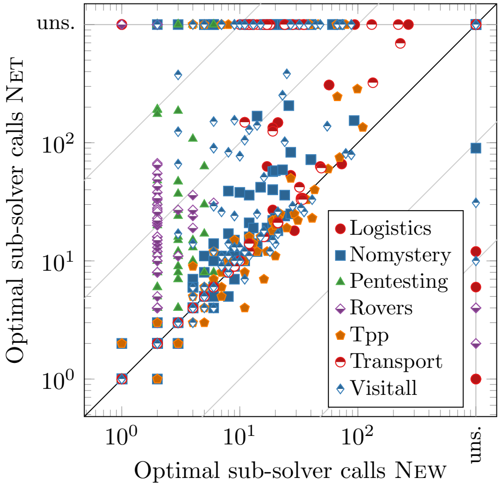
Overall Results

Coverage: Tasks solved under 30 minutes and 4GB

Follower sub-solver		IDS Π^+	SLS —	SLS $+ub$	SLS $+\Pi^+$	SLS + FF
LMcut	OLD (1987)	**681**	630	634	652	651
	NEW (1059)	681	708	**743**	740	736
	NET (1064)	526	536	574	613	**619**
Symbolic Bidirectional	OLD (1987)	584	621	628	**652**	**652**
	NEW (1059)	632	819	823	825	**826**
	NET (1064)	540	654	671	**726**	720

Results: Standard vs Net-Benefit Instances





Outline

Conclusions

- Stackelberg planning is an interesting form of adversarial planning for robustness analysis, pentesting, etc.

Conclusions

- Stackelberg planning is an interesting form of adversarial planning for robustness analysis, pentesting, etc.
- Symbolic Leader Search
  - Symbolic search for efficient exhaustive exploration
  - Information sharing across follower sub-problems

$\rightarrow$outperforms the previous state of the art

Conclusions

- Stackelberg planning is an interesting form of adversarial planning for robustness analysis, pentesting, etc.
- Symbolic Leader Search
    - Symbolic search for efficient exhaustive exploration
    - Information sharing across follower sub-problems

  $\rightarrow$outperforms the previous state of the art
- Net-benefit Stackelberg planning
    - Soft goals
    - More fine-grained analysis of how many goals can the follower achieve

  $\rightarrow$increases usefulness of this framework

Conclusions

- Stackelberg planning is an interesting form of adversarial planning for robustness analysis, pentesting, etc.
- Symbolic Leader Search
  - Symbolic search for efficient exhaustive exploration
  - Information sharing across follower sub-problems

  $\rightarrow$outperforms the previous state of the art
- Net-benefit Stackelberg planning
  - Soft goals
  - More fine-grained analysis of how many goals can the follower achieve

  $\rightarrow$increases usefulness of this framework
- Promising future work:
  - Apply information sharing ideas in other contexts
  - Improved cost-bounded planning algorithms