Heuristic search

Weighted A*

Kustaa Kangas

October 17, 2013

Weighted A*

Weighted A* search – unifying view and application

Rüdiger Ebendt, Rolf Drechsler, 2009

Weighted A*

- Weight the heuristic to quickly direct the search.
- Save time, get bounded suboptimality in exchange.

Outline

1. Three approaches: WA*, DWA*, A^*_ε
2. Unifying view
3. Monotone heuristic
4. Approximate BDD minimization
5. Experiments

Standard A*

Standard A*

$$f(q) = g(q) + h(q)$$

Finds an optimal path if h is admissible, i.e. $h(q) \leq h^*(q)$.

Constant inflation

WA*: constant inflation

$$f^{\uparrow}(q) = g(q) + (1+\varepsilon)h(q)$$

where $\varepsilon \geq 0$.

If h is admissible, then WA* is ε-admissible, i.e.

$$g(q) \leq (1+\varepsilon)C^*$$

for all expanded q where C^* is the length of an optimal path.

ε-admissibility

If h is admissible, then WA* is ε-admissible.

Proof.

Let $s \ldots q \ldots t$ be an optimal path where q is the first node of the path in the open list. Assume a goal state t is expanded. This can happen only if

$$\begin{aligned} g(t) = f(t) &\leq f(q) \\ &= g(q) + (1+\varepsilon)h(q) \\ &\leq g^*(q) + (1+\varepsilon)h^*(q) \\ &\leq (1+\varepsilon)(g^*(q) + h^*(q)) \\ &= (1+\varepsilon)C^* \end{aligned}$$

$\square$

Dynamic weighting

DWA*: Dynamic weighting

$$f^{DW}(q) = g(q) + \left(1 + \varepsilon \cdot \left[1 - \frac{d(q)}{N}\right]\right) h(q)$$

where

$d(q)$	depth of q
N	depth of optimal solution

Idea: as the search goes deeper, emphasize the heuristic less.

How do we get N?

- Sometimes known beforehand: e.g. BDD minimization.
- Generally not known: use an upper bound.

A^*_ε

Keep the original cost function

$$f(q) = g(q) + h(q)$$

Instead of expanding q with the smallest $f(q)$, define

$$FOCAL = \left\{ q \in OPEN \mid f(q) \leq (1+\varepsilon) \cdot \min_{r \in OPEN} f(r) \right\}$$

Use *another* heuristic h_F to choose a minimum from FOCAL, i.e.

$$\hat{q} = \arg\min_{q \in FOCAL} h_F(q)$$

A^*_ε

$$f(q) = g(q) + h(q)$$

$$\hat{q} = \operatorname*{arg\,min}_{q \in FOCAL} h_F(q)$$

Original idea:

- h estimates solution cost
- h_F estimates remaining search effort

Suggestions for h_F:

- $h_F = h$
- $h_F(q) = N - d(q)$

Unifying view

$$f(q) = g(q) + h(q)$$

$$FOCAL = \left\{ q \in OPEN \mid f(q) \leq (1+\varepsilon) \cdot \min_{r \in OPEN} f(r) \right\}$$

WA* and DWA* are actually special cases of A^*_ε

$$h_F(q) = f^{\uparrow}(q) = g(q) + (1+\varepsilon)h(q) \quad \text{WA}^*$$

$$h_F(q) = f^{DW}(q) = g(q) + \left(1 + \varepsilon \cdot \left[1 - \frac{d(q)}{N}\right]\right) h(q) \quad \text{DWA}^*$$

Unifying view

A^*_ε is a unifying framework.

- Any result for A^*_ε follows for WA^* and DWA^*
  - e.g. ε-admissibility
- Makes the approaches comparable (same f)

Unifying view

Concern: what if weighted A^* expands many q with

$$C^* \leq f(q) \leq (1+\varepsilon)C^*$$

- Could overcome the advantages of directing the search.
- General A^*_ε makes no guarantees.
- For WA^* and DWA^* this happens relatively rarely.

Monotone heuristic

Monotone heuristic

If for every state q and its descendant q'

$$h(q) \leq c(q, q') + h(q')$$

the heuristic is *monotone* or *consistent*.

A* with a monotone heuristic

- When q is expanded, $g(q) = g^*(q)$
- Expanded states are never reopened

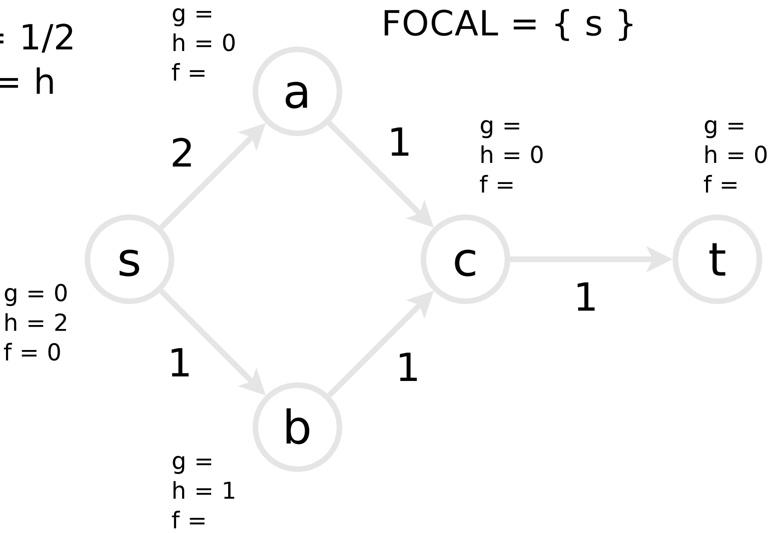
Does this hold for weighted A*?

Monotone heuristic

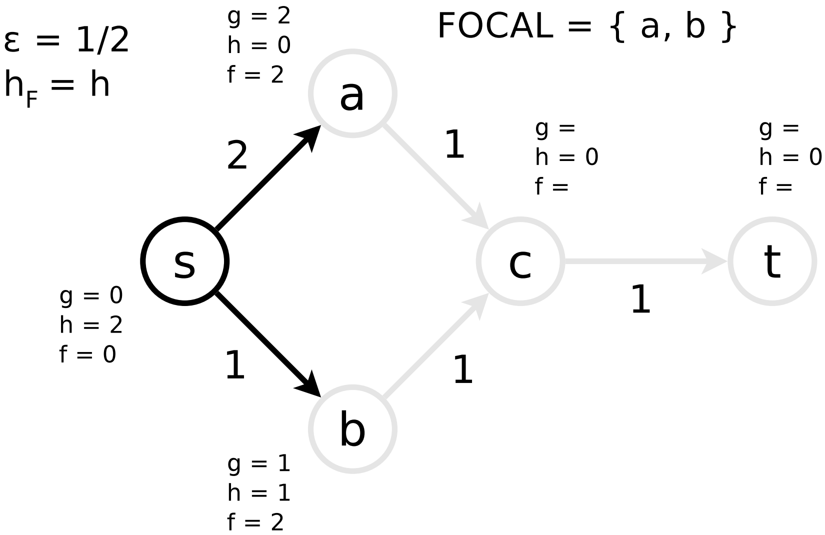
$\varepsilon = 1/2$

$h_F = h$

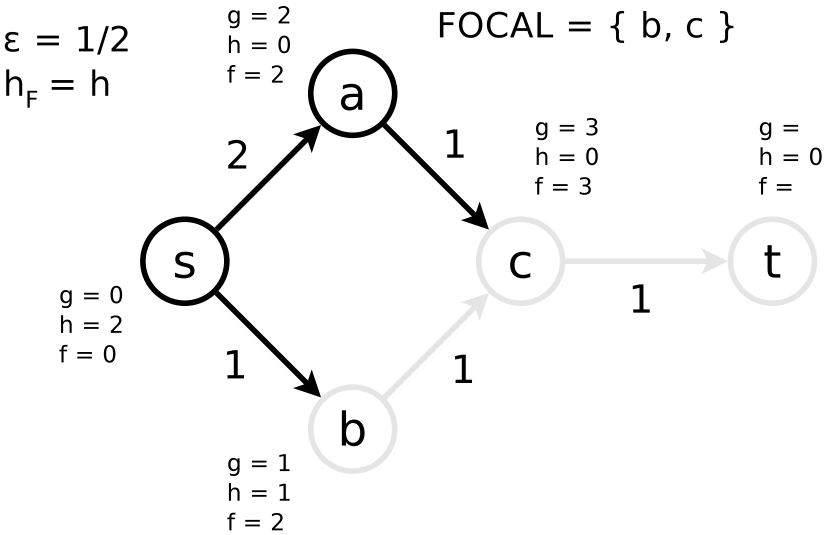
FOCAL = { s }



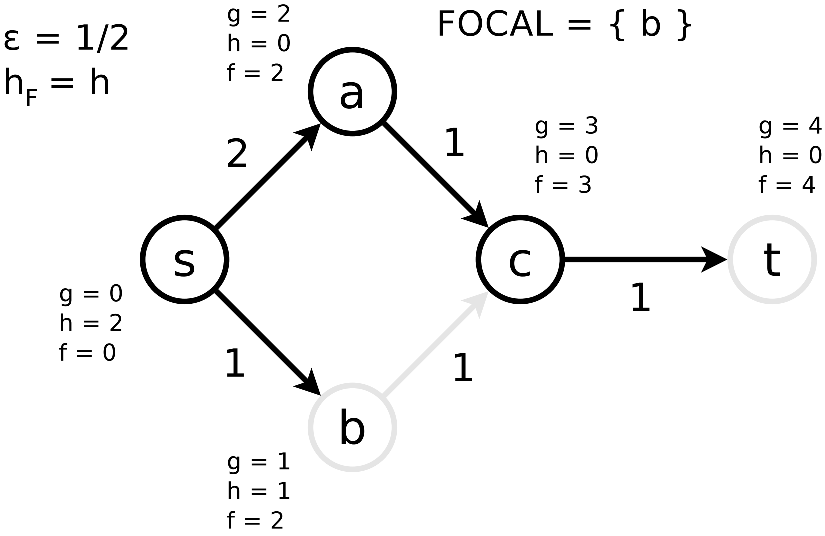
Monotone heuristic



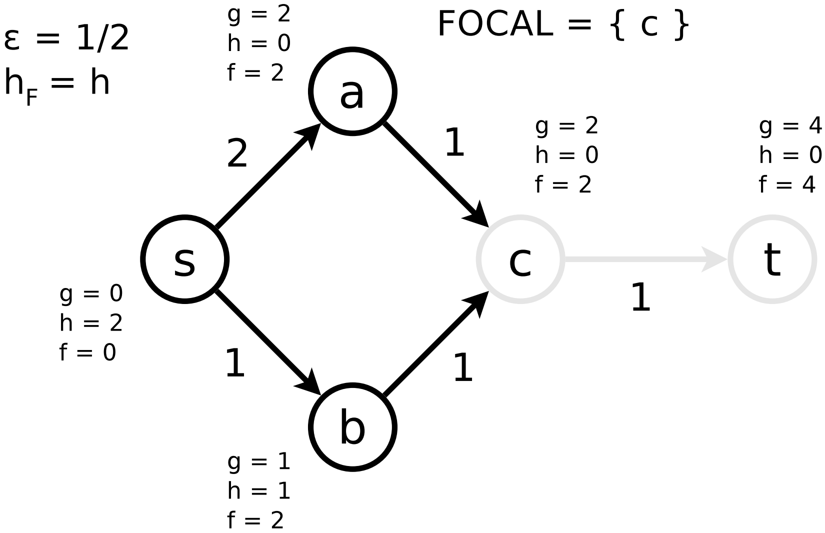
Monotone heuristic



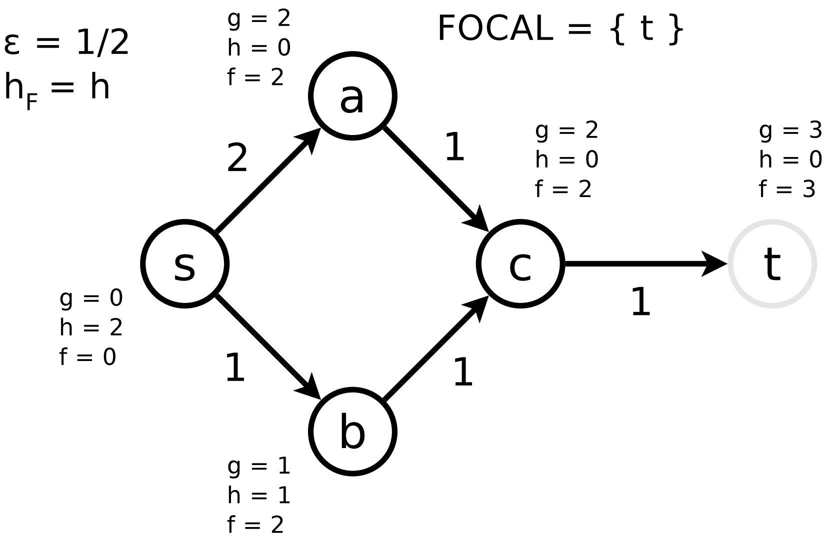
Monotone heuristic



Monotone heuristic



Monotone heuristic

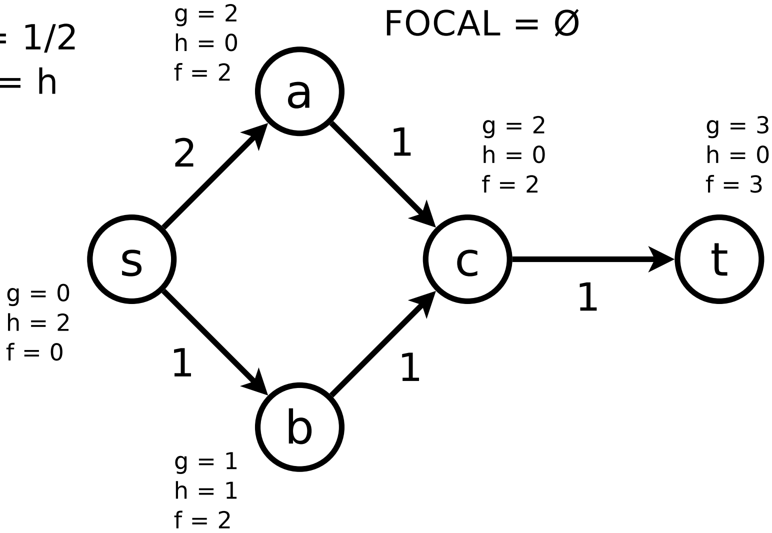


Monotone heuristic

$\varepsilon = 1/2$

$h_F = h$

FOCAL = Ø



Monotone heuristic

Turns out no. However, we do get the bound

$$g(q) \leq (1+\varepsilon)g^*(q) + \varepsilon \cdot h(q)$$

for all expanded q.

- Weighted A* benefits less from a monotone heuristic.
- Reopening may increase running times significantly.

Without reopening

What if we don't reopen states? Simply ignore any new better path.

Without reopening

What if we don't reopen states? Simply ignore any new better path.

- Turns out the $C \leq (1+\varepsilon)C^*$ bound no longer holds.

Without reopening

What if we don't reopen states? Simply ignore any new better path.

- Turns out the $C \leq (1+\varepsilon)C^*$ bound no longer holds.
- Instead, we can show

$$C \leq (1+\varepsilon)^{\lfloor N/2 \rfloor} C^*$$

where N is the depth of the optimal solution.

Without reopening

What if we don't reopen states? Simply ignore any new better path.

- Turns out the $C \leq (1+\varepsilon)C^*$ bound no longer holds.
- Instead, we can show

$$C \leq (1+\varepsilon)^{\lfloor N/2 \rfloor} C^*$$

  where N is the depth of the optimal solution.
- Intuition: shortcutting requires always at least two states.

Without reopening

What if we don't reopen states? Simply ignore any new better path.

- Turns out the $C \leq (1+\varepsilon)C^*$ bound no longer holds.
- Instead, we can show

$$C \leq (1+\varepsilon)^{\lfloor N/2 \rfloor} C^*$$

  where N is the depth of the optimal solution.
- Intuition: shortcutting requires always at least two states.
- Each shortcut accumulates the error by a factor of $(1+\varepsilon)$.

Without reopening

What if we don't reopen states? Simply ignore any new better path.

- Turns out the $C \leq (1+\varepsilon)C^*$ bound no longer holds.
- Instead, we can show

$$C \leq (1+\varepsilon)^{\lfloor N/2 \rfloor} C^*$$

  where N is the depth of the optimal solution.
- Intuition: shortcutting requires always at least two states.
- Each shortcut accumulates the error by a factor of $(1+\varepsilon)$.
- WA* and DWA* are still ε-admissible without reopening.

Experiments

All variants were evaluated on a number of problems:

- BDD minimization
- Blocksworld
- Sliding-tile puzzle
- Depots
- Logistics
- PSR
- Satellite
- Freecell
- Driverlog

Experiments

All variants were evaluated on a number of problems:

- BDD minimization
- Blocksworld
- Sliding-tile puzzle
- Depots
- Logistics
- PSR
- Satellite
- Freecell
- Driverlog

Boolean functions

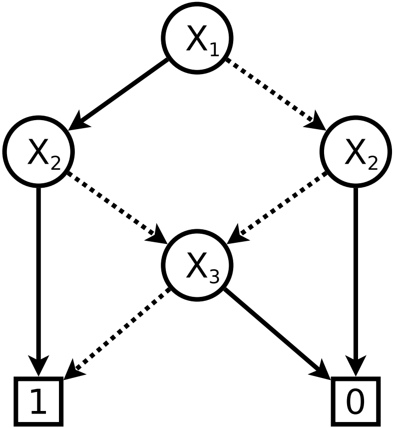
A Boolean function is a function

$$f : \{0,1\}^n \to \{0,1\}$$

Can be represented as a table, e.g. $n = 3$:

X_1	X_2	X_3		Y
0	0	0	$\mapsto$	1
0	0	1	$\mapsto$	0
0	1	0	$\mapsto$	0
0	1	1	$\mapsto$	0
1	0	0	$\mapsto$	1
1	0	1	$\mapsto$	0
1	1	0	$\mapsto$	1
1	1	1	$\mapsto$	1

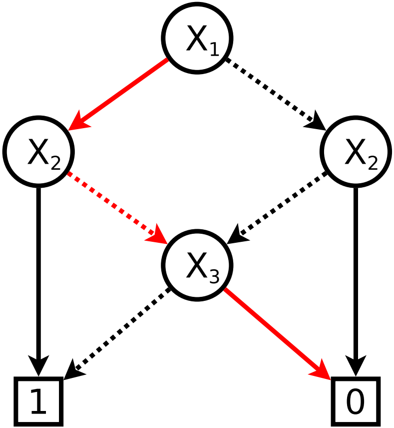
Boolean decision diagrams



Boolean decision diagrams

$X_1 = 1$
$X_2 = 0$
$X_3 = 1$

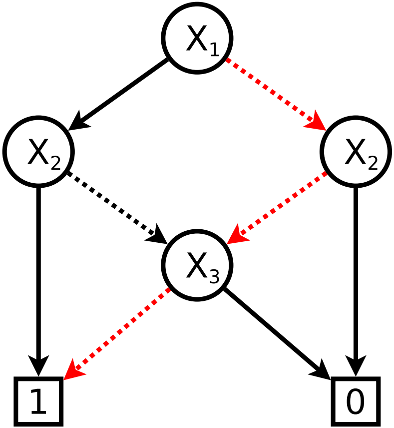
$Y = 0$



Boolean decision diagrams

$X_1 = 0$
$X_2 = 0$
$X_3 = 0$

$Y = 1$



Boolean decision diagrams

Several applications

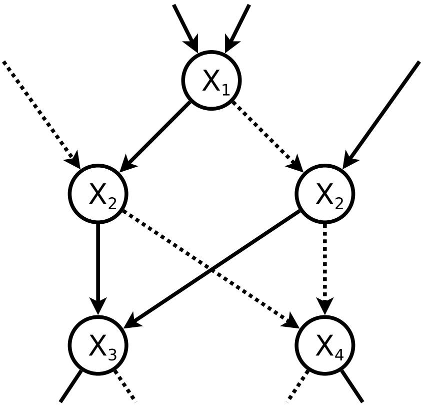
- Model checking
- Sparse-memory applications
- Planning
- Symbolic heuristic search
- Enchancing heuristic search (e.g. A^*)

In general we want BDDs to be as small as possible.

- Easier to read
- Take less memory
- Faster to evaluate

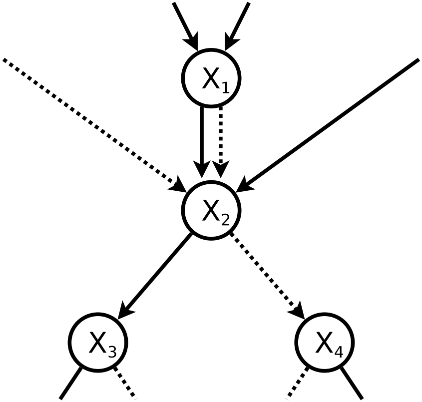
Boolean decision diagrams

BDDs are not unique and can often be simplified.



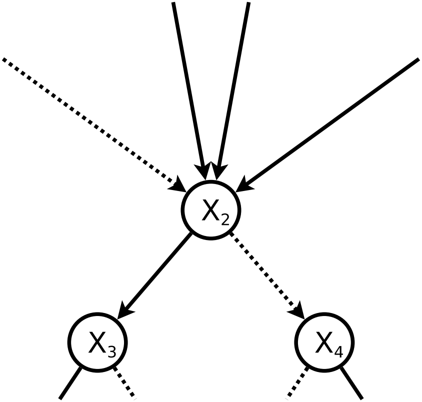
Boolean decision diagrams

BDDs are not unique and can often be simplified.



Boolean decision diagrams

BDDs are not unique and can often be simplified.

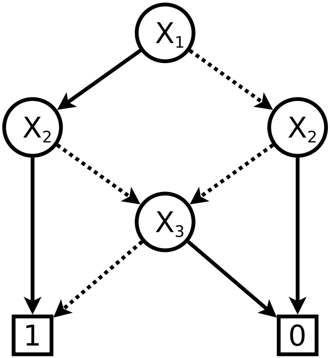


Boolean decision diagrams

- For a fixed permutation of variables, applying merge and deletion iteratively yields a minimal BDD.
- However, the permutation determines how small BDDs we can achieve.

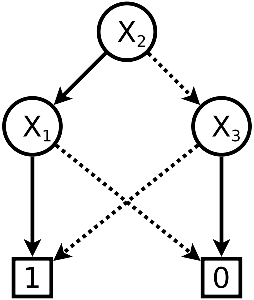
Boolean decision diagrams

Optimal BDD for permutation X_1, X_2, X_3.



Boolean decision diagrams

Optimal BDD for permutation X_2, X_1, X_3 (and X_2, X_3, X_1)



Boolean decision diagrams

BDD minimization problem: find an ordering of variables that yields a minimal BDD (least nodes)

- NP-hard (decision version is NP-complete)
- Can be solved exactly in $O(3^n n)$.
- Can often be solved fast with heuristic search.

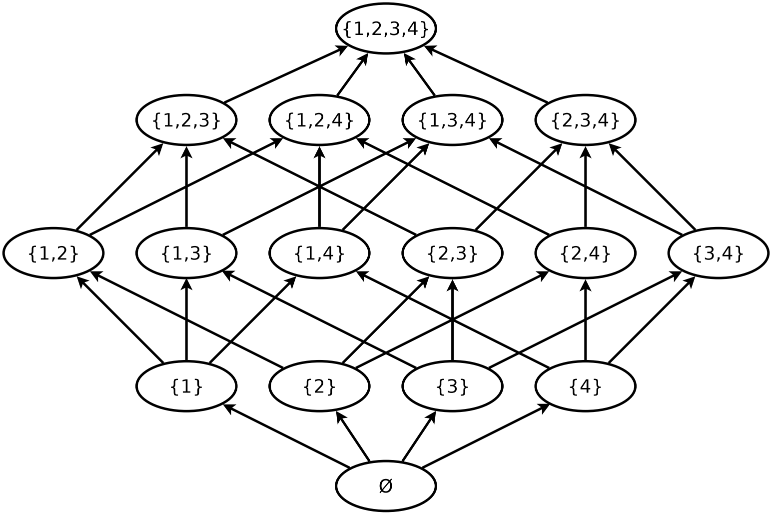
Boolean decision diagrams

In particular, we can formulate it as a path search problem.

- A state is a set of variables whose position in the order has been fixed.
- Each transition fixes the position of a variable.
- A path of length k defines the first k variables in the ordering.

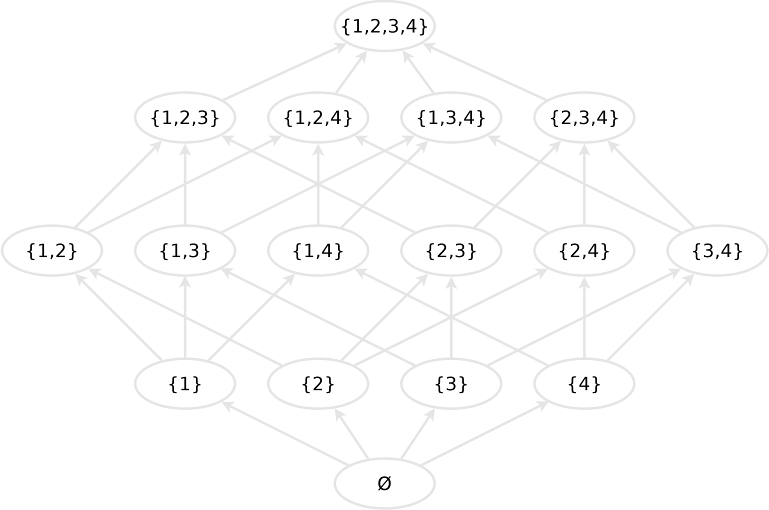
Variable orderings

A permutation corresponds to a path in the subset lattice.



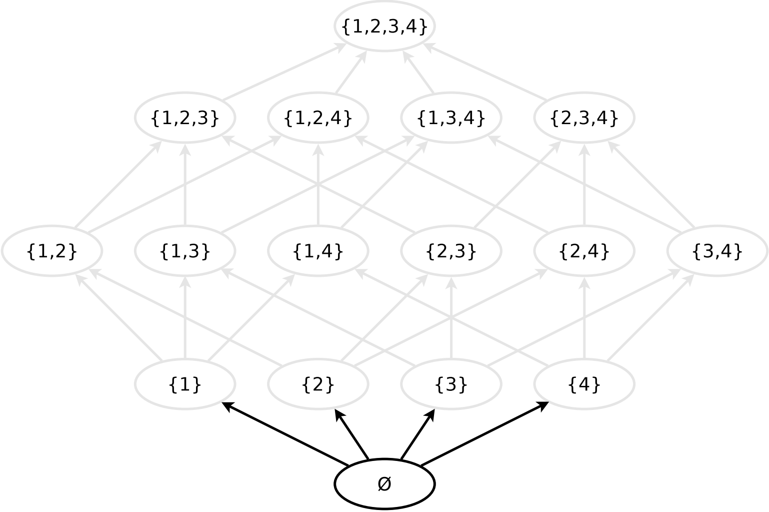
Variable orderings

A permutation corresponds to a path in the subset lattice.



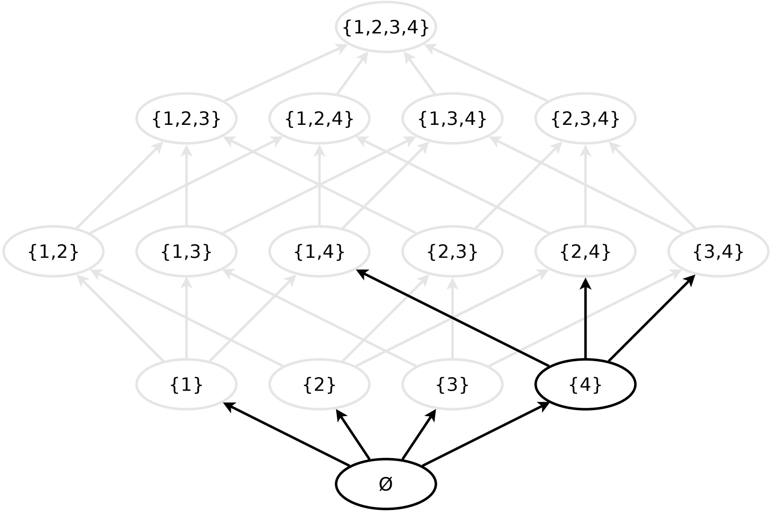
Variable orderings

A permutation corresponds to a path in the subset lattice.



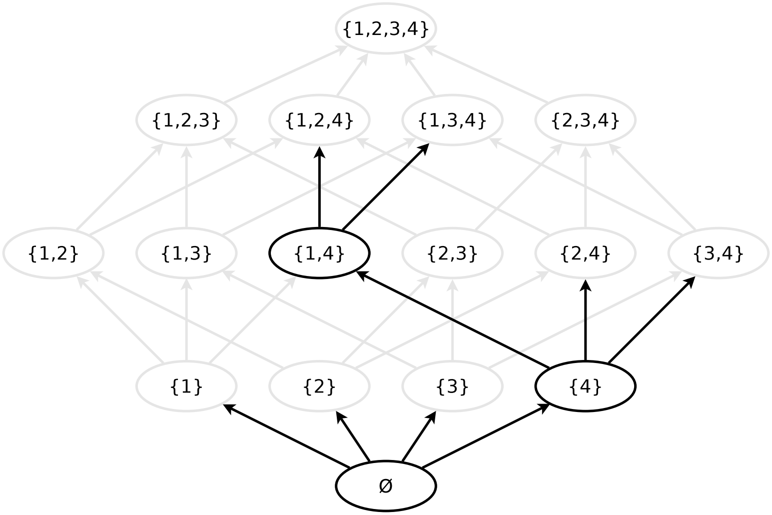
Variable orderings

A permutation corresponds to a path in the subset lattice.



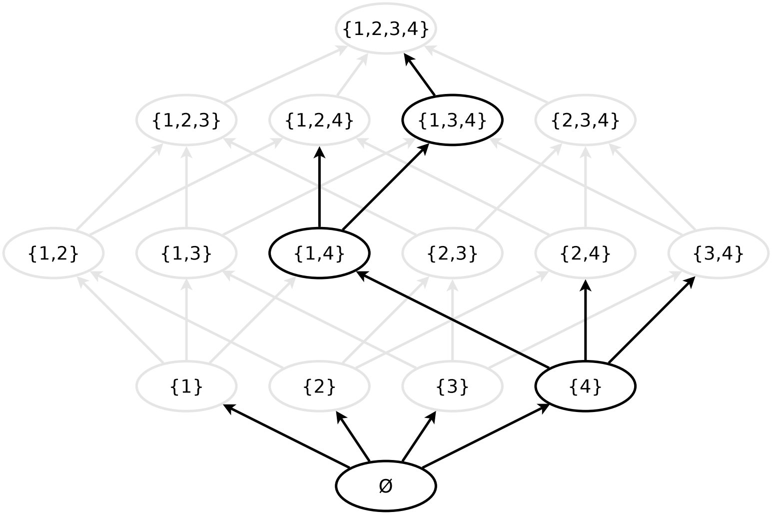
Variable orderings

A permutation corresponds to a path in the subset lattice.



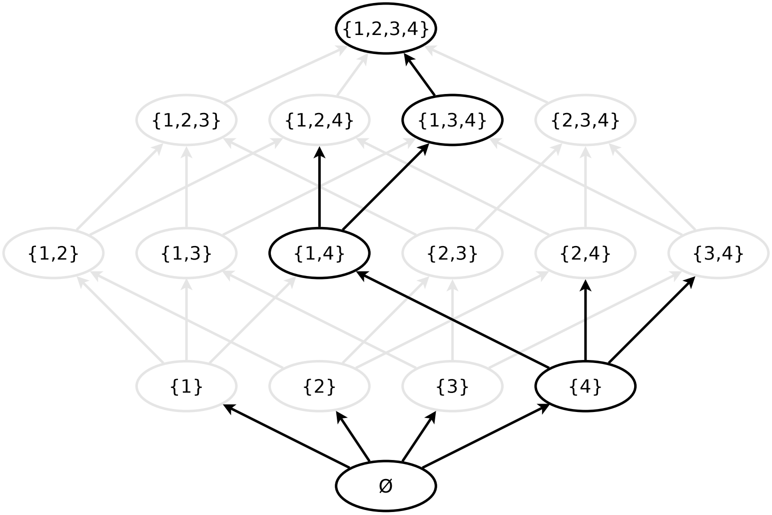
Variable orderings

A permutation corresponds to a path in the subset lattice.



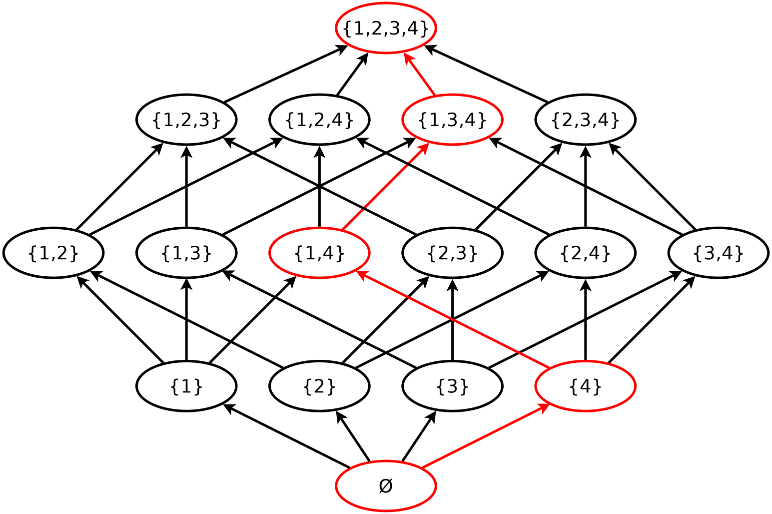
Variable orderings

A permutation corresponds to a path in the subset lattice.



Variable orderings

A permutation corresponds to a path in the subset lattice: 4,1,3,2



Variable orderings

Finding an optimal path is equivalent to finding an optimal BDD.

> g: For a path of length k, the size of the first k levels of the BDD.
> h: The number of cofactors: a lower bound on the size of the BDD.

Weighted A* used for approximate BDD mimimization.

Experiments

Experimental results.

Questions?