# A*-based compilation of decision diagrams for the maximum independent set problem

## Dissertantenseminar

Matthias Horn[1]

[1] Institute of Logic and Computation, TU Wien, Vienna, Austria,
horn@ac.tuwien.ac.at

Oct 19, 2020

ALGORITHMS AND COMPLEXITY GROUP

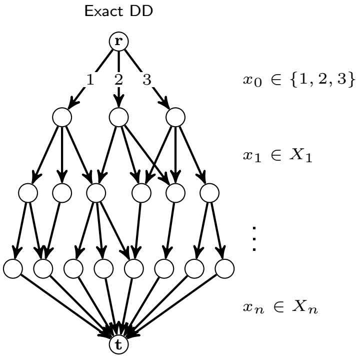
# Overview

- Decision Diagrams 101
  - exact, relaxed and restricted decision diagrams
  - link to dynamic programming
  - compilation of decision diagrams
  - $A^*$ construction
- Maximum Independent Set Problem
  - definition
  - top-down construction + advanced techniques
  - applying $A^*$ construction method
  - results

# Decision Diagrams (DDs) 101

- well known in computer science for decades
  - logic circuit design, formal verification, . . .
- get popular in combinatorial optimization in the last decade
  - graphical representation of solutions of a combinatorial optimization problem (COP)
  - weighted directed acyclic multi-graph with one root node $\mathbf{r}$ and one target node $\mathbf{t}$
  - each $\mathbf{r}$-$\mathbf{t}$ path corresponds to a solution of the COP
  - length of a path coincides with the solution's objective value
- state of the art results could be obtained on several problems

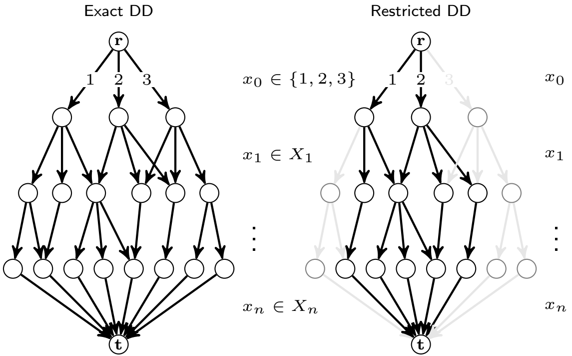
# Decision Diagrams (DDs) 101




## Exact DDs

- represent precisely the set of feasible solutions of a COP
- longest path: corresponds to optimal solution
- tend to be exponential in size ⇒ approximate exact DD

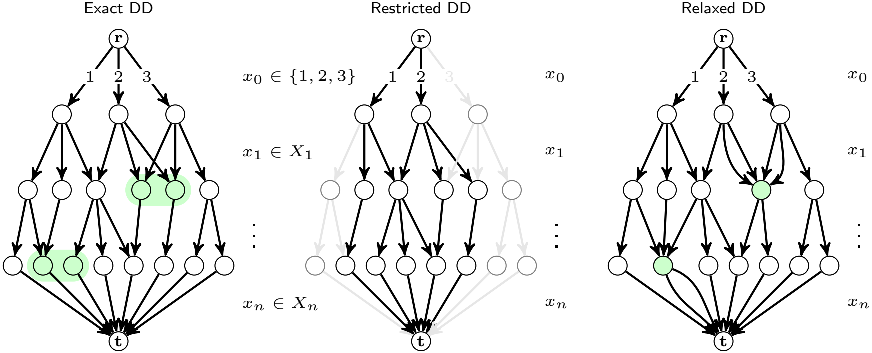
# Decision Diagrams (DDs) 101




## Restricted DDs

- represent subset of feasible solutions of a COP
- by removing nodes and edges
- length of longest path: corresponds to a primal bound

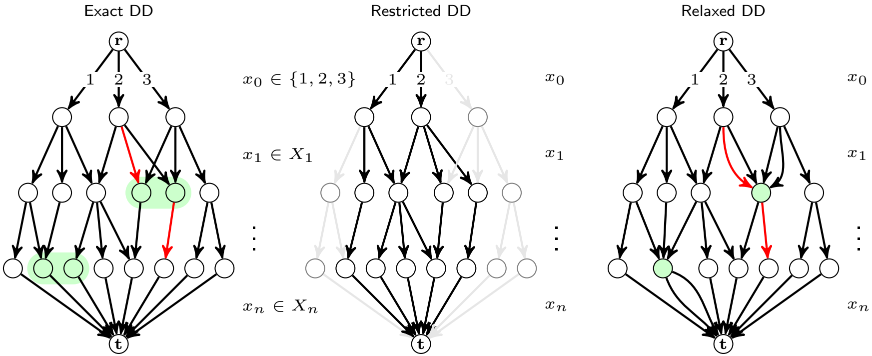
# Decision Diagrams (DDs) 101




## Relaxed DDs

- represent superset of feasible solutions of a COP
- by merging nodes
- length of longest path: corresponds to an upper bound
- discrete relaxation of solution space

# Decision Diagrams (DDs) 101




## Relaxed DDs

- represent superset of feasible solutions of a COP
- by merging nodes
- length of longest path: corresponds to an upper bound
- discrete relaxation of solution space

# Relaxed DDs

- discrete relaxation of solution space
- usage
  - to obtain dual bounds
  - as constraint store in constraint propagation
  - derivation of cuts in mixed integer programming (MIP)
  - branch-and-bound: branching on merged nodes
  - ...
- excellent results on e.g.
  - set covering (Bergman et al., 2011)
  - independent set (Bergman et al., 2014)
  - time dependent traveling salesman (Cire and Hoeve, 2013)
  - time dependent sequential ordering (Kinable et al. 2017)

# DDs - Construction Methods

- Top-Down Construction (TDC)
  - compile relaxed DD layer by layer
  - layer width is limited
  - if current layer gets too large $\Rightarrow$ merge nodes
- Incremental Refinement (IR)
  - start with relaxed DD of width one
  - iteratively refine by splitting nodes and filtering arcs
- A*-based Construction (A*C)
  - accepted for publication in *Computers & Operations Research*
  - construct a relaxed DD by a modified A* algorithm
  - the size of the open list is limited by parameter $\phi$
  - for PC-JSOCMSR: obtained smaller DDs with stronger bounds in shorter time

# DDs and Dynamic Programming

- dynamic programming (DP)
  - controls $x_i \in X_i$, current state $s_i$, stages $i = 1, \ldots, n$
  - transitions: $s_{i+1} = \phi_i(s_i, x_i), \quad i = 1, \ldots, n$
  - objective function: $f(x) = \sum_{i=1}^{n} c_i(s_i, x_i)$
  - can be solved recursively

$$g_i(x_i) = \min_{x_i \in X_i(s_i)} \left\{ c_i(s_i, x_i) + g_{i+1}(\phi_i(s_i, x_i)) \right\}, \quad i = 1, \ldots, n$$

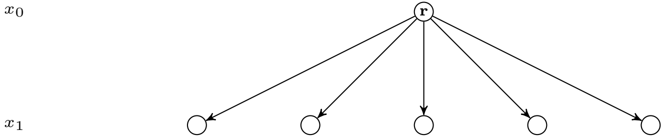
- exact DDs are strongly related to DP
  - J. N. Hooker, *Decision Diagrams and Dynamic Programming*, 2013.
  - each node of DD is associated to a DP state
  - root node $s_0$, target node $s_{n+1}$
  - arc $(s_i, \phi_i(s_i, x_i))$ with cost $c_i(s_i, x_i)$ for each control $x_i \in X_i$
  - create a DD based on a DP formulation without solving it
  - provides recursive formulations of the COP

# Relaxed DD - Top Down Construction

- compiled layer by layer
- the size of each layer is limited by width $\beta$ (here $\beta = 3$)
- merge strategy: rank states of the current layer and merge the worst states

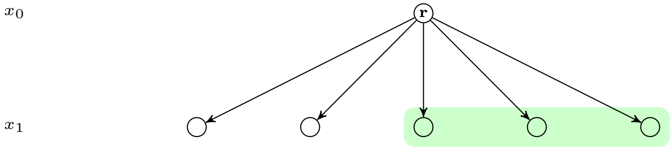
# Relaxed DD - Top Down Construction

- compiled layer by layer
- the size of each layer is limited by width $\beta$ (here $\beta = 3$)
- merge strategy: rank states of the current layer and merge the worst states



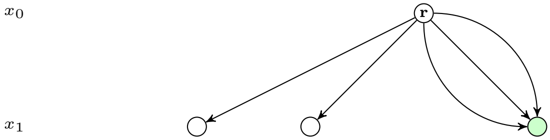
# Relaxed DD - Top Down Construction

- compiled layer by layer
- the size of each layer is limited by width $\beta$ (here $\beta = 3$)
- merge strategy: rank states of the current layer and merge the worst states



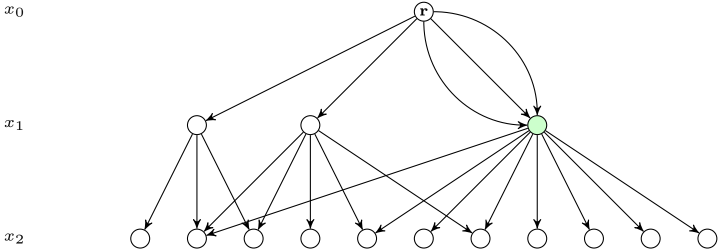
# Relaxed DD - Top Down Construction

- compiled layer by layer
- the size of each layer is limited by width $\beta$ (here $\beta = 3$)
- merge strategy: rank states of the current layer and merge the worst states



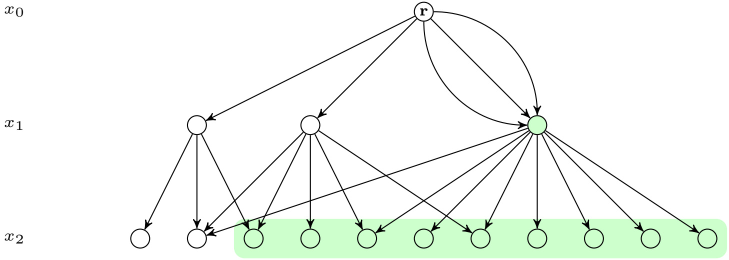
# Relaxed DD - Top Down Construction

- compiled layer by layer
- the size of each layer is limited by width $\beta$ (here $\beta = 3$)
- merge strategy: rank states of the current layer and merge the worst states



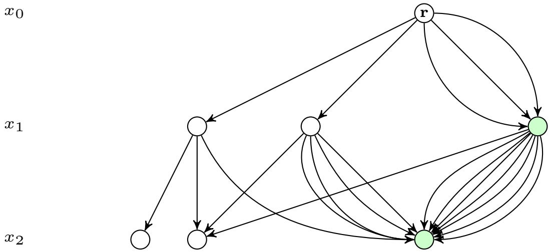
# Relaxed DD - Top Down Construction

- compiled layer by layer
- the size of each layer is limited by width $\beta$ (here $\beta = 3$)
- merge strategy: rank states of the current layer and merge the worst states



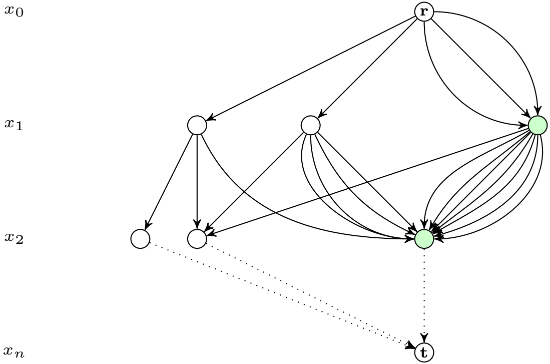
# Relaxed DD - Top Down Construction

- compiled layer by layer
- the size of each layer is limited by width $\beta$ (here $\beta = 3$)
- merge strategy: rank states of the current layer and merge the worst states



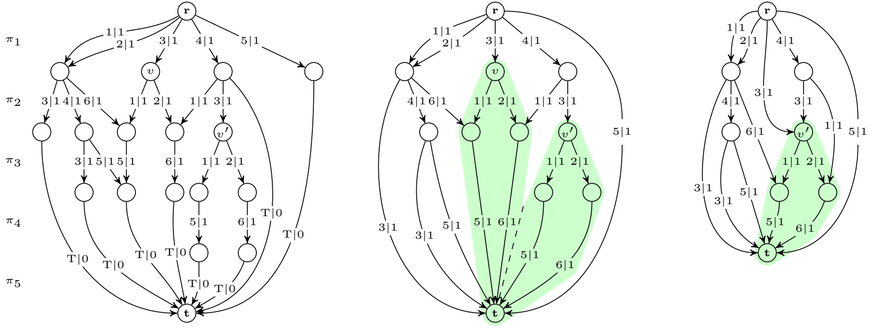
# Relaxed DD - Top Down Construction

- compiled layer by layer
- the size of each layer is limited by width $\beta$ (here $\beta = 3$)
- merge strategy: rank states of the current layer and merge the worst states



# Relaxed DD - A* Construction (A*C)

- Disadvantages of TDC
  - states can only be merged within layers
  - nodes on different layers may correspond to the same state
  - isomorphic substructures may appear




(a) (b) (c)

# Relaxed DD - $A^*$ Construction ($A^*C$)

- **Idea:** Switch from breadth-first search to best-first search!
  - layers do not play a role anymore
- Construct a DD by using a modified A* algorithm:
  - the size of the open list $|Q|$ is limited by parameter $\phi$
  - if $\phi$ would be exceeded, worst ranked nodes are merged.
- Key characteristics:
  - naturally avoids multiple nodes for identical states at different substructures and
  - consequently multiple copies of isomorphic substructures
  - node expansions and selection of nodes to be merged are guided by an auxiliary upper bound function

# A* Search

A$^*$: classical informed search algorithm for path planning in possibly huge graphs (Hart et al., 1968)

- uses a heuristic function, here an upper bound $Z^{\mathrm{ub}}$, to guide the search
- maintains an open list $Q$ of nodes sorted according to priorities

$$f(u) = Z^{\mathrm{lp}}(u) + Z^{\mathrm{ub}}(u)$$

Initially: $Q = \{\mathbf{r}\}$
Repeat:

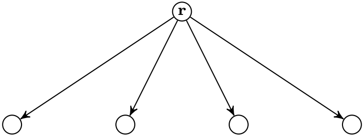
- pop node $u \in Q$ with maximum $f(u)$
- if $u = \mathbf{t}$ terminate
- expand $u$: determine successor nodes

# Relaxed DD - A* Construction

# Relaxed DD - A* Construction

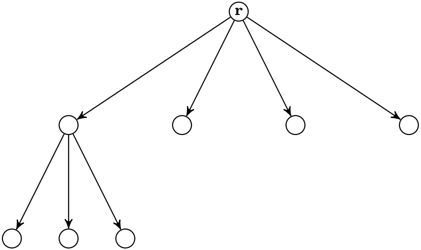

At each iteration:
pop node $u \in Q$ with maximum $f(u)$ and expand $u$

# Relaxed DD - A* Construction



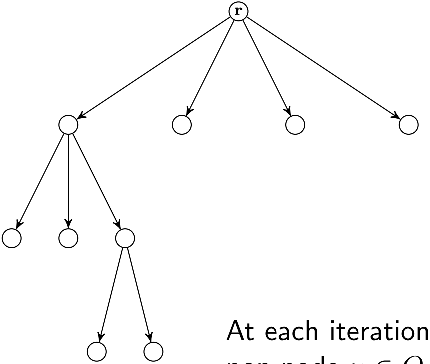

At each iteration:
pop node $u \in Q$ with maximum $f(u)$ and expand $u$

# Relaxed DD - A* Construction



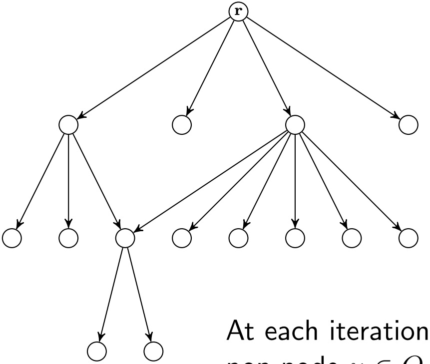

At each iteration:
pop node $u \in Q$ with maximum $f(u)$ and expand $u$

# Relaxed DD - A* Construction



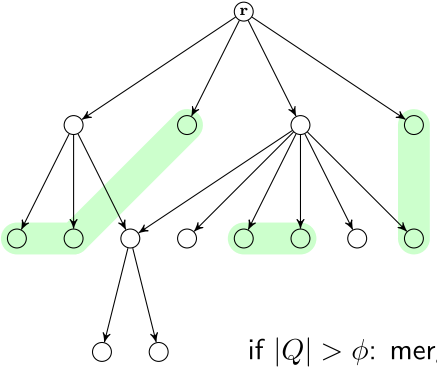

At each iteration:
pop node $u \in Q$ with maximum $f(u)$ and expand $u$

# Relaxed DD - A* Construction



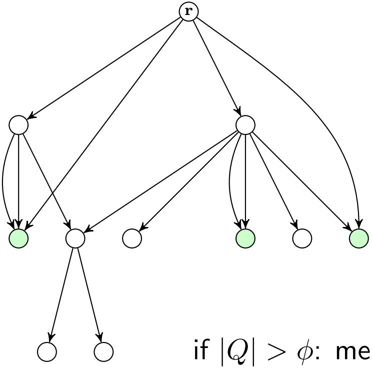

At each iteration:
pop node $u \in Q$ with maximum $f(u)$ and expand $u$

# Relaxed DD - A* Construction



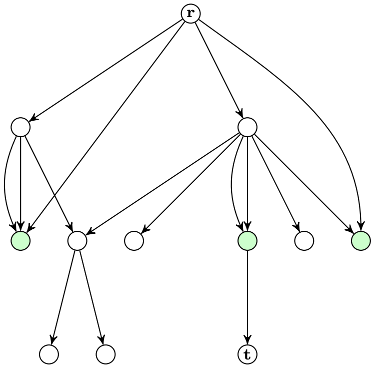

if $|Q| > \phi$: merge nodes

# Relaxed DD - A* Construction

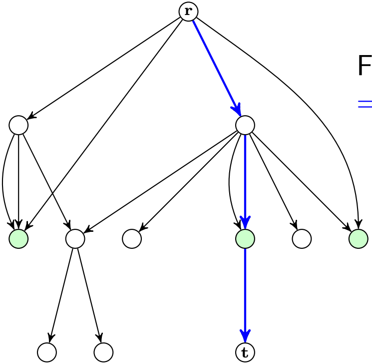



if $|Q| > \phi$: merge nodes

# Relaxed DD - A* Construction

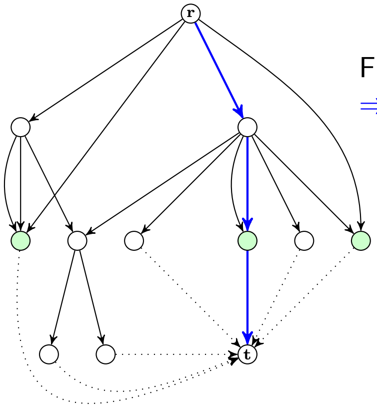


# Relaxed DD - A* Construction




First time target state expanded:
$\Rightarrow Z^{\text{ub}}_{\min} := Z^{\text{lp}}(\mathbf{t})$ is a feasible UB

# Relaxed DD - A* Construction

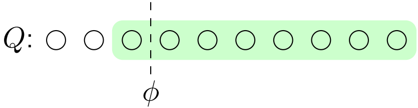



First time target state expanded:
$\Rightarrow Z^{\mathrm{ub}}_{\min} := Z^{\mathrm{lp}}(\mathbf{t})$ is a feasible UB

To get complete relaxed DD:
continue until $Q$ is empty

# Relaxed DD - A* Construction

## Merging Strategies

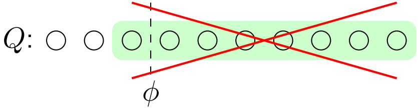



We cannot just merge all worst $|Q| - \phi + 1$ states into a single state:

- cycles can emerge
- there may be already expanded states in $Q$
- merging different states may introduce a large relaxation loss

# Relaxed DD - A* Construction

## Merging Strategies




We cannot just merge all worst $|Q| - \phi + 1$ states into a single state:

- ⚠ cycles can emerge
  - ➔ must be taken into consideration when merging nodes
- ⚠ there may be already expanded states in $Q$
  - ➔ merge only not yet expanded states
- ⚠ merging different states may introduce a large relaxation loss
  - ➔ try to merge only similar states

# Relaxed DD - A* Construction

## Merging Strategies

To efficiently reduce $Q$ and identify similar states we maintain

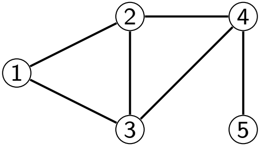
- we apply labeling function $L(u)$ and maintain
- a dictionary of collector nodes $V^{\mathrm{C}}$ indexed by $L(u)$.
- only ever merge nodes having the same label

While $|Q| > \phi$, iteratively consider not yet expanded states $u \in Q \setminus V^{\mathrm{C}}$ in increasing priority order:

- if $\exists v \in V^{\mathrm{C}} : L(v) = L(u)$ then merge $u$ into $v$
- otherwise $V^{\mathrm{C}} \leftarrow V^{\mathrm{C}} \cup \{u\}$

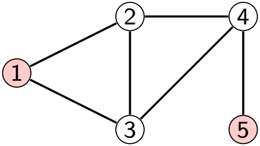
# Maximum Independent Set Problem (MISP)

- Definition
  - given: graph $G = (V, E)$, $V = \{1, 2, \ldots, n\}$
  - find the maximum independent set $I$, s.t. $I \subseteq V$ and no two vertices in $I$ are connected by an edge in $E$
  - let $N(j) = \{j' \mid (j, j') \in E\} \cup \{j\}$ be the neighborhood of $j$
- Example (from Bergman et al., 2016)



# Maximum Independent Set Problem (MISP)

- Definition
  - given: graph $G = (V, E)$, $V = \{1, 2, \ldots, n\}$
  - find the maximum independent set $I$, s.t. $I \subseteq V$ and no two vertices in $I$ are connected by an edge in $E$
  - let $N(j) = \{j' \mid (j, j') \in E\} \cup \{j\}$ be the neighborhood of $j$
- Example (from Bergman et al., 2016)




$I = \{1, 5\}$

# Maximum Independent Set Problem (MISP)

## Binary DDs (BDDs) for MISP

- well studied in literature
  - BDD-based branch and bound algorithm
  - branching on merged nodes
  - uses TDC to created relaxed and restricted BDDs
- advanced techniques
  - long arcs: *zero-suppressed BDD*
    skip over variables whose values are represented implicitly
  - dynamic variable ordering
    select next decision variable during compilation of BDD
- we aim to compile relaxed BBDs with A*C

# Maximum Independent Set Problem (MISP)

## BDD - Top Down Construction

### Dynamic Programming Formulation

- decision variables:
  $x_j \in \{0,1\}$, $j = 1, \ldots, n$, if $j$ is selected (=1) or not (=0)
- state space:
  $S_j = 2^{V_j}$ for $j = 2, \ldots, n$, $\hat{r} = V$, $\hat{t} = \emptyset$ and $V_j = \{j, \ldots, n\}$
- transition functions:

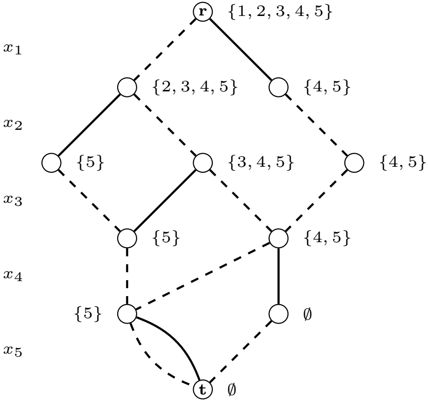
$$t_j(s^j, 0) = s^j \setminus \{j\}; \quad t_j(s^j, 1) = \begin{cases} s^j \setminus N(j), & \text{if } j \in s^j; \\ \hat{0}, & \text{if } j \notin s^j; \end{cases}$$

- cost functions: $h_j(s^j, 0) = 0$; $h_j(s^j, 1) = 1$;
- node merger: union

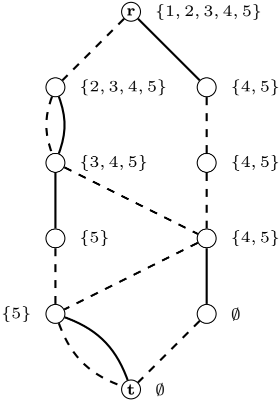
# Maximum Independent Set Problem (MISP)

## BDD - Top Down Construction

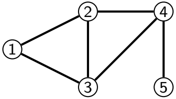
exact BDD




relaxed BDD, $\beta = 2$




MISP instance:



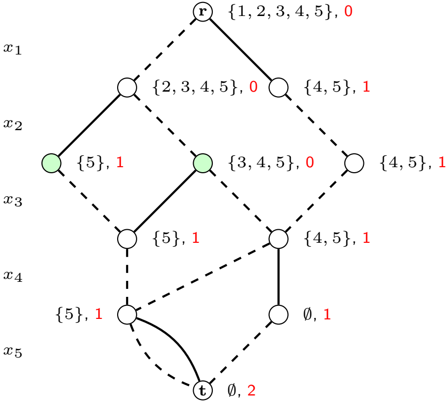

node ranking: $Z^{\mathrm{lp}}$, decreasing

$Z^{\mathrm{lp}}(v)$: longest path from $\mathbf{r}$ to node $v$

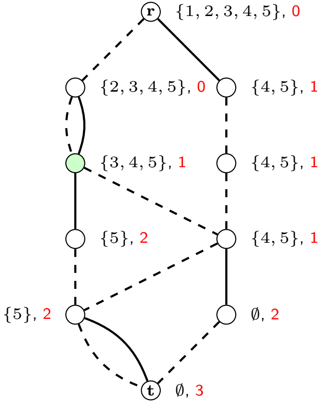
# Maximum Independent Set Problem (MISP)

## BDD - Top Down Construction

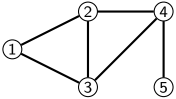
exact BDD




relaxed BDD, $\beta = 2$




MISP instance:



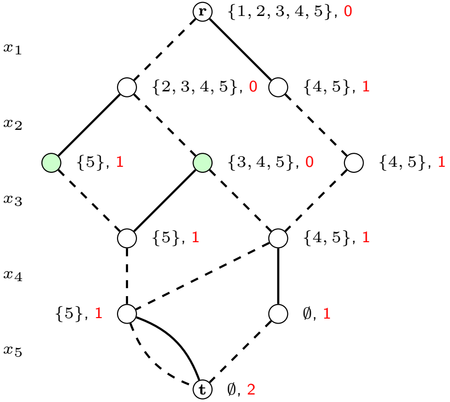

node ranking: $Z^{\text{lp}}$, decreasing

$Z^{\text{lp}}(v)$: longest path from $\mathbf{r}$ to node $v$

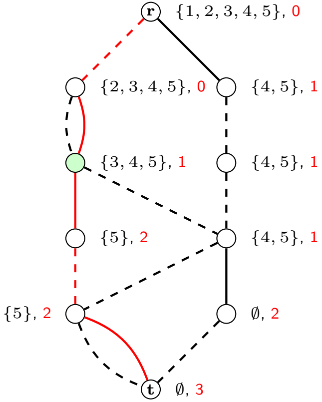
# Maximum Independent Set Problem (MISP)

## BDD - Top Down Construction

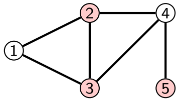
exact BDD




relaxed BDD, $\beta = 2$




MISP instance:



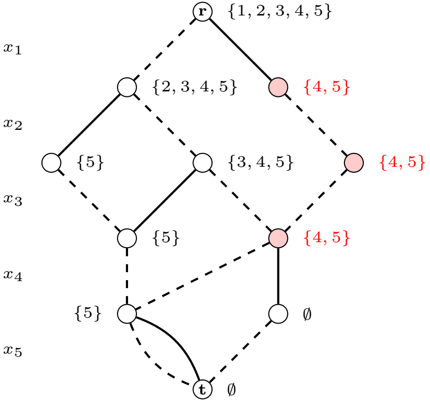

node ranking: $Z^{\text{lp}}$, decreasing

$Z^{\text{lp}}(v)$: longest path from $\mathbf{r}$ to node $v$

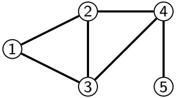
# Maximum Independent Set Problem (MISP)

## BDD - Top Down Construction

exact BDD




MISP instance:

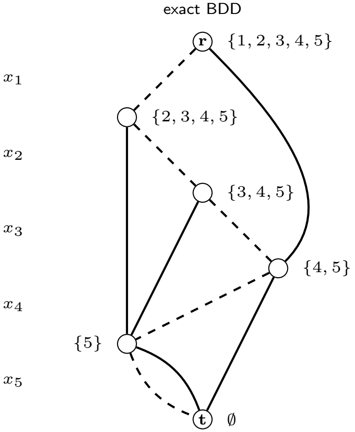



### Zero-suppressed BDD

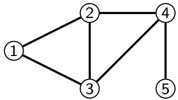
- long arcs: skip layers
- skipped variables are implicitly set to zero
- maintain open list of nodes
- current layer $j$: insert only nodes that contain $j$

# Maximum Independent Set Problem (MISP)

## BDD - Top Down Construction




MISP instance:



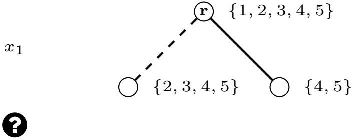

### Zero-suppressed BDD

- long arcs: skip layers
- skipped variables are implicitly set to zero
- maintain open list of nodes
- current layer $j$: insert only nodes that contain $j$

# Maximum Independent Set Problem (MISP)

## BDD - Top Down Construction

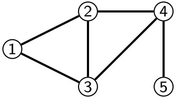
exact BDD




### Variable Ordering

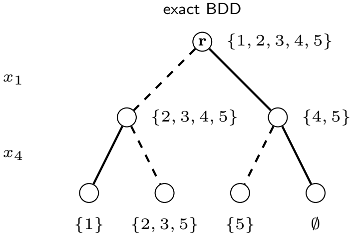
- Size of DD depends on variable ordering!
- select variable for each layer at run-time

MISP instance:



# Maximum Independent Set Problem (MISP)

## BDD - Top Down Construction

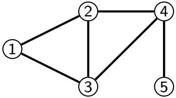



### Variable Ordering

- Size of DD depends on variable ordering!
- select variable for each layer at run-time

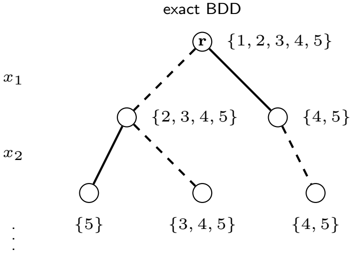
for each layer after merging:
select variable that belongs to the fewest number of states in current layer

MISP instance:



# Maximum Independent Set Problem (MISP)

## BDD - Top Down Construction

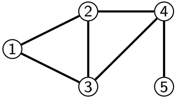



### Variable Ordering

- Size of DD depends on variable ordering!
- select variable for each layer at run-time

for each layer after merging:
select variable that belongs to the fewest number of states in current layer

MISP instance:



# Maximum Independent Set Problem (MISP)

## BDD - A* Construction

### Dynamic Programming Formulation

- decision variables:
  $x_j \in \{0,1\}$, $j = 1,\ldots,n$, if $j$ is selected (=1) or not (=0)
- (partial) variable ordering: $(l_i)_{i=1}^n$
  node $l_i \in V$ is considered at $i$-th position
- state space:
  $s \in 2^{V_j}$ for $j = 2,\ldots,n$, $\hat{r} = V$, $\hat{t} = \emptyset$ and $V_j = \{j,\ldots,n\}$
- next variable to consider for state $s$:
  $\mathrm{next}(s) := \arg\min_{j \in s} l_j^{-1}$ if any $j$ is assigned, $\top$ otherwise
- transition functions:
  $t(s,0) = s \setminus \{\mathrm{next}(s)\}$; $t(s,1) = s \setminus N(\mathrm{next}(s))$
- cost functions: $h(s,0) = 0$; $h(s,1) = 1$;
- node merger: union

# Maximum Independent Set Problem (MISP)

## BDD - A* Construction - Details

- Merging strategies
  - merge only nodes with same $\mathrm{next}(s)$ value
  - $L_0(s) = (\mathrm{next}(s))$
  - $L_1(s) = (\mathrm{next}(s), Z^{\mathrm{lp}}(s))$
  - $L_2(s) = (\mathrm{next}(s), Z^{\mathrm{ub}}(s))$
- Open list order for merging:
  - sort $Q$ according to $Z^{\mathrm{lp}}$, decreasing
- Variable ordering heuristic
  - if state $s$ is selected for expansion with $\mathrm{next}(s) = \top$
  - select variable that belongs to the fewest number of not yet expanded states in open list

# Maximum Independent Set Problem (MISP)

## BDD - A* Construction - Upper bounds

- Hansen Bound
  - $Z^{\text{ub}}_{\text{hansen}}(s) = \lfloor 1/2 + \sqrt{1/4 + n(s)^2 - n(s) - 2e(s)} \rfloor$
  - $n(s)$ := number of induced nodes
  - $e(s)$ := number of induced edges
- Borg Bound
  - $Z^{\text{ub}}_{\text{borg}}(s) = \lfloor ((\Delta(s) - 1) - n(s) + 1)/\Delta(s) \rfloor$
  - $\Delta(s)$ := max degree of nodes in $s$
  - must be a connected graph
- Minimum Degree Bound
  - $Z^{\text{ub}}_{\delta}(s) = n(s) - \delta(s)$
  - $\delta(s)$ := minimum degree of nodes in $s$

# Maximum Independent Set Problem (MISP)

## BDD - A* Construction - Upper bounds

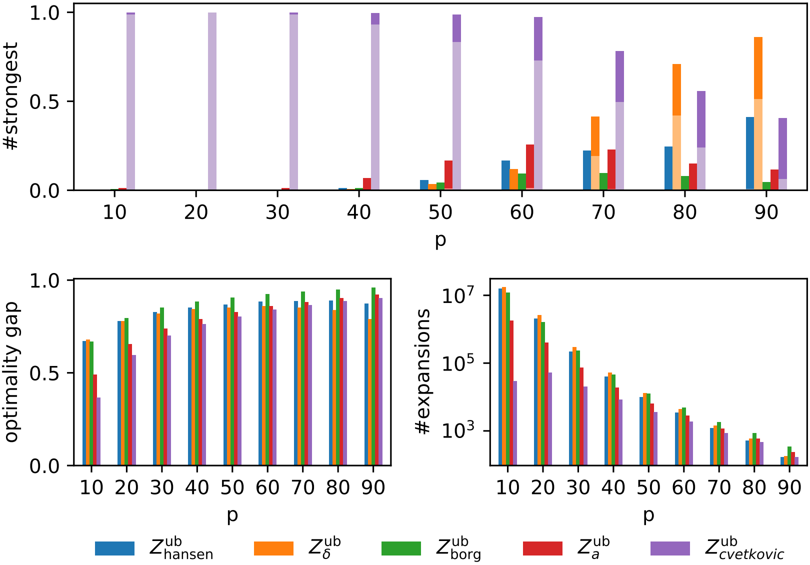
- Annihilation Number Bound
  - $Z_a^{\text{ub}}(s) = a(s) \ldots$ annihilation number
  - Let $d_1 \leq d_2 \leq \ldots \leq n(s)$ be the sequence of non-decreasing degrees of nodes in $s$. Then the annihilation $a(s)$ is the largest integer $0 \leq k \leq n(s)$ that satisfies $\sum_{i=1}^{k} d_i \leq \sum_{i=k+1}^{n(s)} d_i$.
- Cvetkovic Bound
  - $Z_{\text{cvetkovic}}^{\text{ub}}(s) = p_0(s) + \min\{p_-(s), p_+(s)\}$
  - $p_0(s) :=$ number of eigenvalues equal to zero
  - $p_-(s) :=$ number of eigenvalues smaller than zero
  - $p_+(s) :=$ number of eigenvalues greater than zero
- Strongest Bound
  - $Z^{\text{ub}}(s) = \min\{Z_{\text{hansen}}^{\text{ub}}(s), Z_{\text{borg}}^{\text{ub}}(s), Z_{\delta}^{\text{ub}}(s), Z_a^{\text{ub}}(s)\}$

# Instances

- MISP
  - random graphs with $n \in \{100, 250, 500, \ldots, 1750\}$ and density $p \in \{0.1, 0.2, \ldots, 0.9\}$
  - 10 graphs per $n$,$p$ configuration

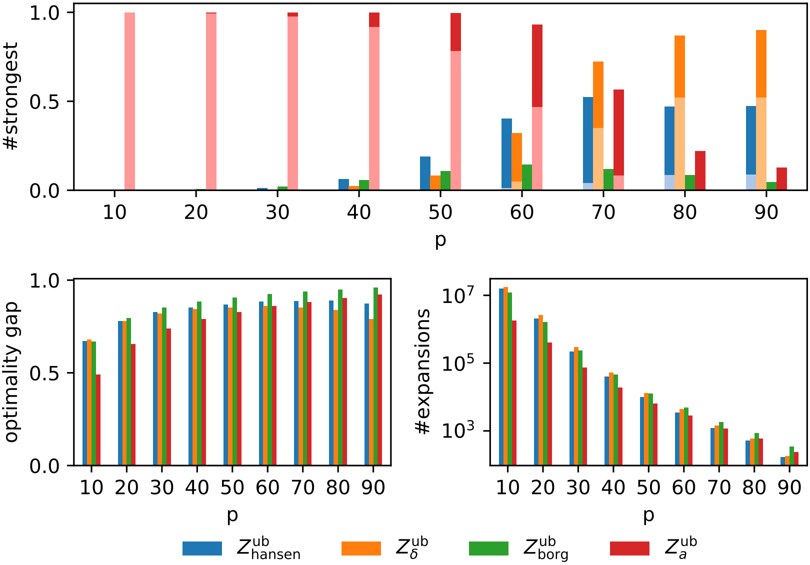
# A* Search

## Upper Bound Comparison, $n = 100$



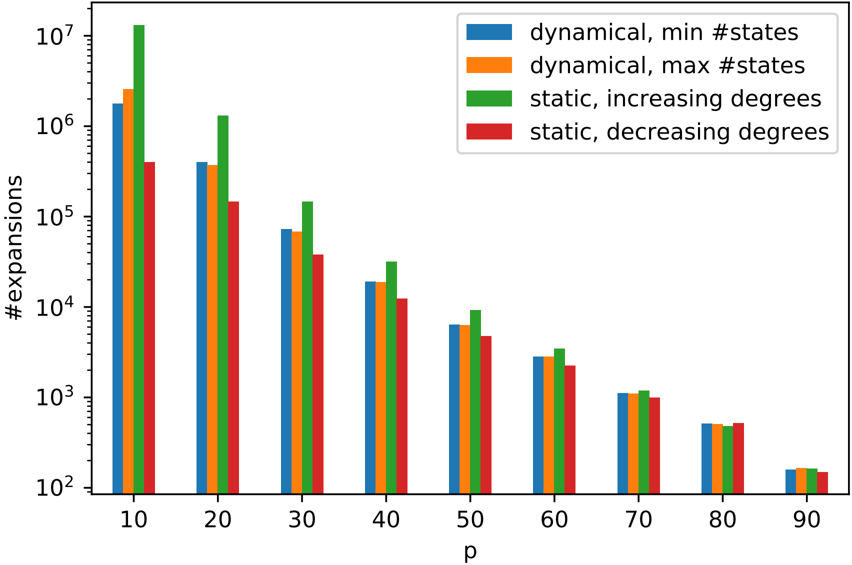
# A* Search

## Upper Bound Comparison, $n = 100$



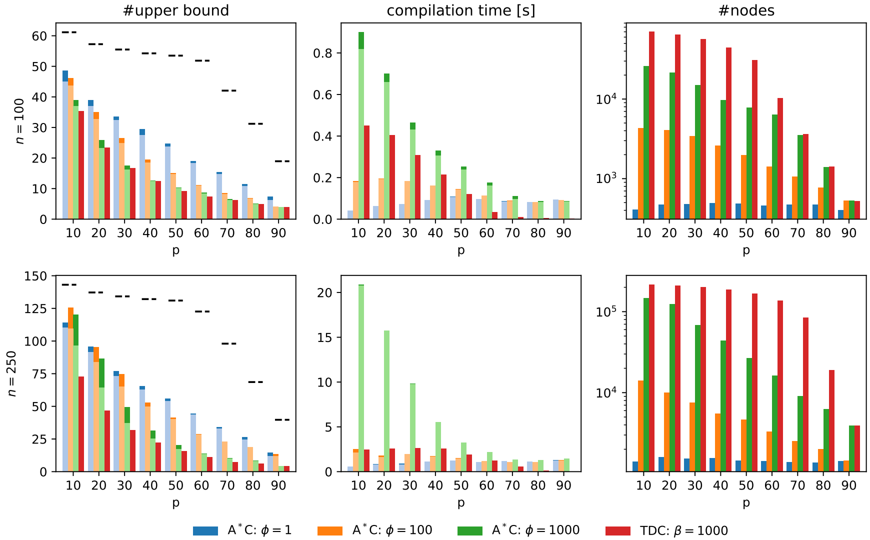
# A* Search

Variable Order, $n = 100$



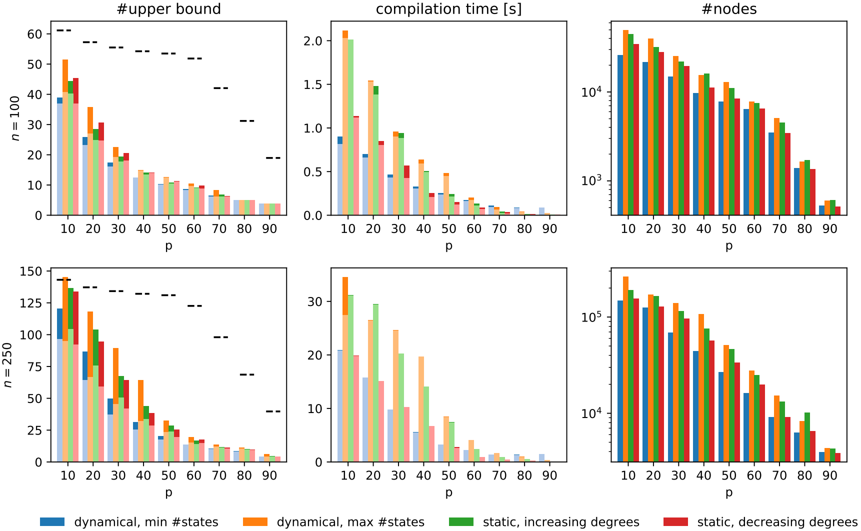
# A*C - Different values for $\phi$

## Variable order: dynamical+min #states, labeling function: $L_0(\cdot)$



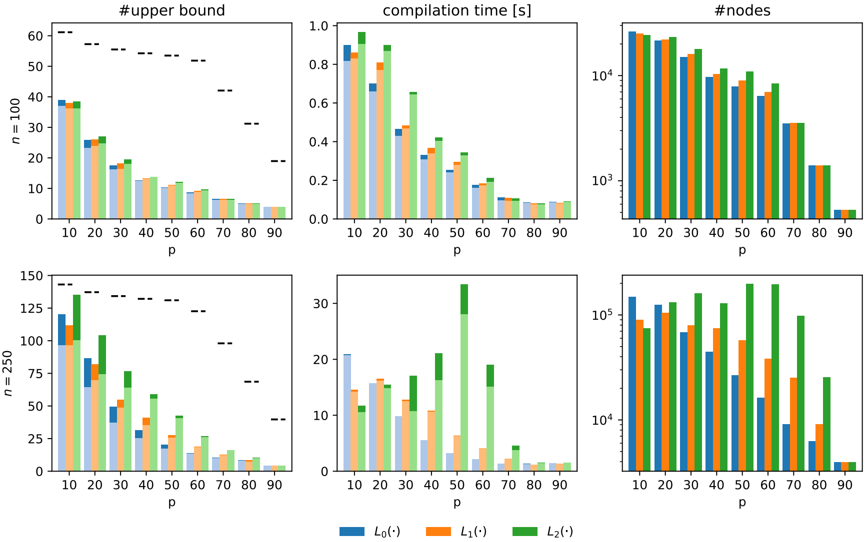
# A*C - Variable Order

$\phi = 1000$, labeling function: $L_0(\cdot)$



# A*C - Labeling Functions

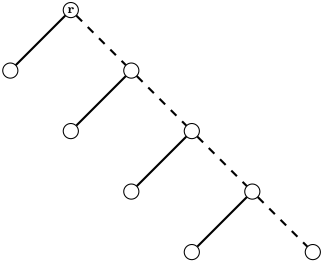
## $\phi = 1000$, variable order: dynamical+min #states



# Issues

⚠ Zero-arcs are preferred

- states expanded by zero-arcs do not change much
- such states seems promising according to priority function $f$
- leads to asymmetric expansion of nodes



# Issues

**⚠ Zero-arcs are preferred**

- states expanded by zero-arcs do not change much
- such states seems promising according to priority function $f$
- leads to asymmetric expansion of nodes

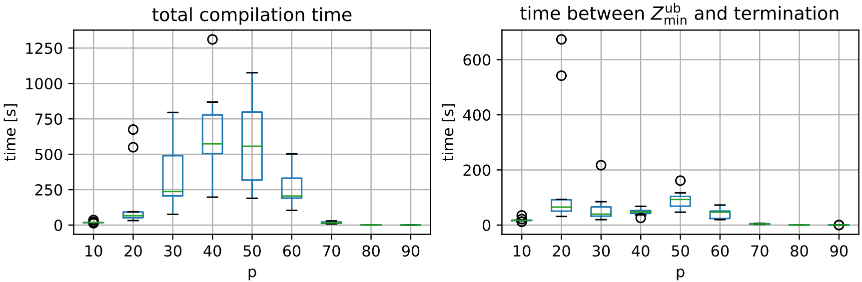
**Possible solutions:**

- weighted priority function $f_w(\cdot) = Z^{\mathrm{lp}}(\cdot) + wZ^{\mathrm{ub}}(\cdot)$, $0 \leq w \leq 1$
  - well known in A$^*$ search literature
  - $w = 0$: expand always state with largest $Z^{\mathrm{lp}}$ value
- switch to multi-valued decision diagrams
  - no zero-arcs anymore
  - high branching factor, layers are much larger

# Issues

⚠ Termination

- some rare incidents: A*C does not terminate in reasonable time
- in particular the time between obtaining $Z^{\mathrm{ub}}_{\mathrm{min}}$ and terminating can be time consuming




instance $n = 250$, A*C: $\phi = 1000$, $L_2(\cdot)$, variable order: decreasing degrees

# Issues

## Termination

- some rare incidents: A*C does not terminate in reasonable time
- in particular the time between obtaining $Z_{\min}^{\mathrm{ub}}$ and terminating can be time consuming

## Possible solutions:

- switch to TDC after $Z_{\min}^{\mathrm{ub}}$ has been obtained
- sliding window
  - A*C operates between layer $L_{\min}$ and $L_{\max}$, $L_{\max} - L_{\min} = k$
  - all nodes in layers $< L_{\min}$ are already expanded
  - if A*C selects a node in layer $L_{\max} + 1$ than all nodes in $L_{\min}$ are expanded and the window is shifted by one
- falling curtain
  - all nodes in layers $< L_{\min}$ are already expanded
  - after $k$ A* iterations, expand all nodes in layer $L_{\min}$
  - increment $L_{\min}$ by one

# Further Ideas

- Consider parent bounds
  - upper bounds from parent nodes minus arc length
- Use hysteresis for limiting open list size
  - start merging if $|Q| > \phi_{\max}$ until $|Q| \geq \phi_{\min}$
- Use TDC to compute upper bounds for A*C
  - $Z^{\mathrm{ub}}_{\min}$ would be as least as strong as UB obtained by TDC
  - A* search: compilation of restricted BDDs every $k$ iteration
    ➔ similar to our anytime A* algorithm

**Thank you for your attention**