A*-based Construction of Multivalued Decision Diagrams

Matthias Horn[1], Johannes Maschler[2], Günther R. Raidl[2], Elina Rönnberg[3]

[1]Algorithmics group, Delft University of Technology, The Netherlands, m.g.horn@tudelft.nl

[2] Institute of Logic and Computation, TU Wien, Austria, {raidl|maschler}@ac.tuwien.ac.at

[3] Department of Mathematics, Linköping University, Sweden, elina.ronnberg@liu.se

October 21, 2022

Overview

- Compile relaxed multivalued decision diagrams (MDDs) with a modified A* algorithm
- Problems exhibit a sequencing and selection aspect
  - elements from a ground set must be selected in a specific order
- Tested on two NP-hard maximization problems
  - longest common subsequence (LCS) problem
  - prize-collecting variant of a scheduling problem

Journal Article

- main work published in Computers & Operations Research 126.105125 (2021)

A^* Search (Hart et al., 1968)

Informed search algorithm for path planning in possibly huge graphs

- uses a heuristic function h to guide the search
- maintains an open list Q of nodes sorted according to priorities

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

Initially: $Q = \{\mathrm{r}\}$ (root node r)
Repeat:

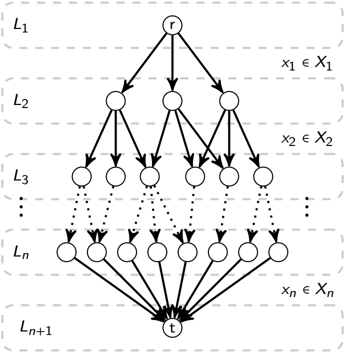
- pop node $u \in Q$ with best $f(u)$
- if $u = \mathrm{t}$ (target node t) then terminate
- expand u: determine successor nodes

Exact approach if h is a dual bound

A*-based Construction of Multivalued Decision Diagrams

Decision Diagrams (DDs)

Exact DDs

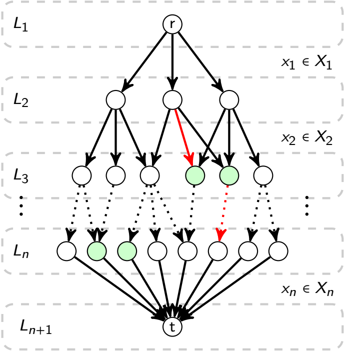


- represent precisely the set of feasible solutions of a combinatorial optimization problem (COP)
- longest path:
  corresponds to optimal solution

⚠ tend to be exponential in size ⇒ approximate exact DD

Decision Diagrams (DDs)

Relaxed DDs



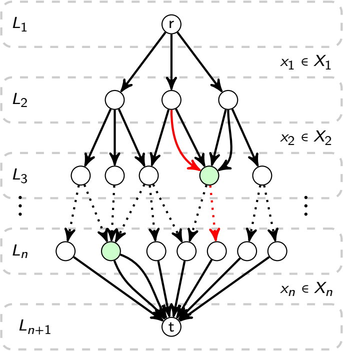
- represent superset of feasible solutions of a COP
- length of longest path: corresponds to an upper bound

superimpose (merge) nodes of exact DD

discrete relaxation of solution space

Decision Diagrams (DDs)

Relaxed DDs



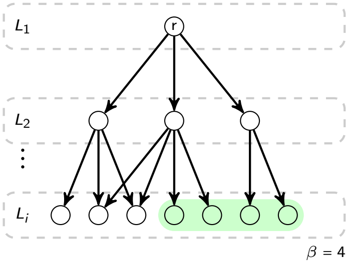
- represent superset of feasible solutions of a COP
- length of longest path: corresponds to an upper bound

superimpose (merge) nodes of exact DD

discrete relaxation of solution space

Compilation of Decision Diagrams

Top-Down Construction



Construction Principle

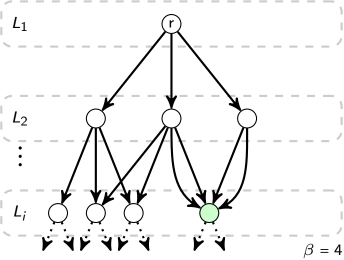
- compiled layer by layer, start with r
- size of each layer is controlled by β
- rank states by heuristic function

Relaxed DD

- merge worst states if $|L_i| > \beta$

Compilation of Decision Diagrams

Top-Down Construction



Construction Principle

- compiled layer by layer, start with r
- size of each layer is controlled by β
- rank states by heuristic function

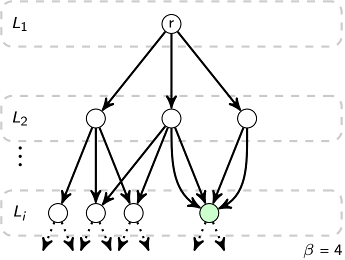
Relaxed DD

- merge worst states if $|L_i| > \beta$

Drawbacks

Compilation of Decision Diagrams

Top-Down Construction



Construction Principle

- compiled layer by layer, start with r
- size of each layer is controlled by β
- rank states by heuristic function

Relaxed DD

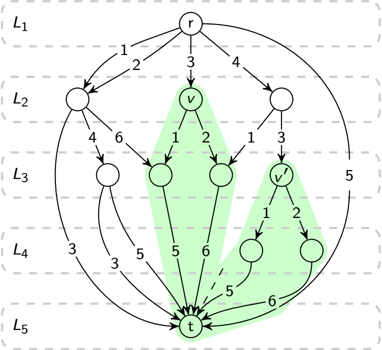
- merge worst states if $|L_i| > \beta$

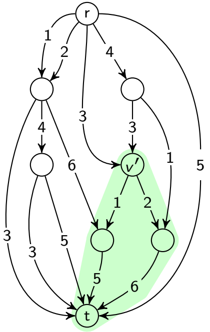
Drawbacks

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

Compilation of Relaxed MDDs

Example: Isomorphic Substructures





Compilation of Relaxed MDDs

A* Construction (A*C)

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 ϕ
  - if ϕ would be exceeded, worst ranked nodes are merged.

- Key characteristics:
  - naturally avoids multiple nodes for identical states
  - avoids multiple copies of isomorphic substructures
  - expansions/selections of nodes: guided by an auxiliary UB function

Longest Common Subsequence (LCS) Problem

Given: m input strings $S = \{s_1, s_2, \ldots, s_m\}$ and alphabet Σ

Task: find the longest common subsequence (LCS)

Subsequence

Obtained by possible deleting characters from an input string.

Common subsequence (CS)

A subsequence which is common to all input strings.

Example: $m = 3$, $\Sigma = \{\mathtt{A},\mathtt{B},\mathtt{C},\mathtt{D}\}$

s_1: A B C D A A B C C

s_2: B A D C B A A B C

s_3: C B A B B D A B C

LCS: BDABC

Applications and Related Work

Wide range of applications

- computational biology: strings represent RNA or DNA segments
- similarity measure of strings, data compression, ...

Deeply investigated over the last decades

Exact approaches

- based on dynamic programming (DP) (Gusfield, 1997)
  - solved in polynomial time $O(n^m)$ for fixed m, otherwise NP-hard
- based on dominant point methods and/or parallelization (..., Li et al., 2016; Peng and Wang, 2017)

Heuristic approaches

- greedy heuristics, large neighborhood search, beam search, ...
- A*-based algorithm (Djukanovic et al., 2020)

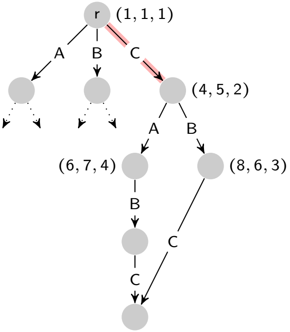
(Relaxed) MDDs for the LCS Problem

A*-based Compilation

s_1: A B C D A A B C C

s_2: B A D C B A A B C

s_3: C B A B B D A B C



Upper bound(s) Z^{ub}:

- literature, e.g. DP-based

Each arc $\alpha \in A$ is

- associated with a character $c(\alpha) \in \Sigma$
- path originating from r identifies a (infeasible) common subsequence

Each node $u \in V$ is/has an

- associated with a position vector $\mathrm{p}(u) = (p_1(u), \ldots, p_m(u))$
- represents subproblem $S[\mathrm{p}(u)]$
- outgoing arc for each feasible and non-dominated character

Merge operation for set of nodes U

- ?

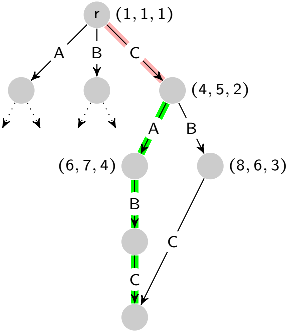
(Relaxed) MDDs for the LCS Problem

A*-based Compilation

s_1: A B C D A A B C C

s_2: B A D C B A A B C

s_3: C B A B B D A B C



Upper bound(s) Z^{ub}:

- literature, e.g. DP-based

Each arc $\alpha \in A$ is

- associated with a character $c(\alpha) \in \Sigma$
- path originating from r identifies a (infeasible) common subsequence

Each node $u \in V$ is/has an

- associated with a position vector $\mathrm{p}(u) = (p_1(u), \ldots, p_m(u))$
- represents subproblem $S[\mathrm{p}(u)]$
- outgoing arc for each feasible and non-dominated character

Merge operation for set of nodes U

- ?

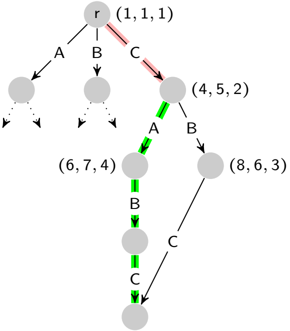
(Relaxed) MDDs for the LCS Problem

A*-based Compilation

s_1: A B C D A A B C C

s_2: B A D C B A A B C

s_3: C B A B B D A B C



Each arc $\alpha \in A$ is

- associated with a character $c(\alpha) \in \Sigma$
- path originating from r identifies a (infeasible) common subsequence

Each node $u \in V$ is/has an

- associated with a position vector $\mathrm{p}(u) = (p_1(u), \ldots, p_m(u))$
- represents subproblem $S[\mathrm{p}(u)]$
- outgoing arc for each feasible and non-dominated character

Upper bound(s) Z^{ub}:

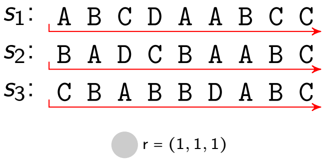
- literature, e.g. DP-based

Merge operation for set of nodes U

- $\oplus(U) = (\min_{u \in U} p_i(u))_{i=1,\ldots,m}$

Exact MDDs for the LCS Problem

A*-based Compilation

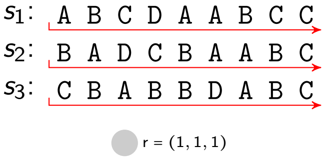


Priority function

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

Exact MDDs for the LCS Problem

A*-based Compilation



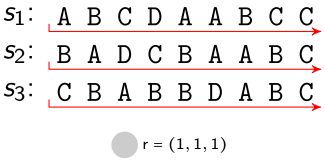
Priority function

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

Evaluate $f(\mathrm{r})$

Exact MDDs for the LCS Problem

A*-based Compilation



Priority function

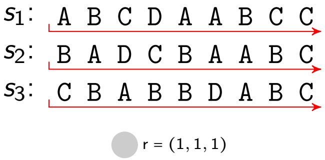
- $f(u) = Z^{\mathrm{lp}}(u) + Z^{\mathrm{ub}}(u)$

Evaluate $f(\mathrm{r})$

- $Z^{\mathrm{lp}}(\mathrm{r}) = 0$

Exact MDDs for the LCS Problem

A*-based Compilation



Priority function

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

Evaluate $f(\mathsf{r})$

- $Z^{\mathrm{lp}}(\mathsf{r}) = 0$
- $Z^{\mathrm{ub}}(\mathsf{r}) = ?$

How to estimate UB?

Exact MDDs for the LCS Problem

A*-based Compilation

s_1: A B C D A A B C C

s_2: B A D C B A A B C

s_3: C B A B B D A B C

r = (1, 1, 1)

Priority function

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

Evaluate $f(\mathsf{r})$

- $Z^{\mathrm{lp}}(\mathsf{r}) = 0$
- $Z^{\mathrm{ub}}(\mathsf{r}) = ?$

How to estimate UB?

- How often does A appear?

Exact MDDs for the LCS Problem

A*-based Compilation

s_1: A B C D A A B C C

s_2: B A D C B A A B C

s_3: C B A B B D A B C

7 r = (1, 1, 1)

Priority function

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

Evaluate $f(\mathrm{r})$

- $Z^{\mathrm{lp}}(\mathrm{r}) = 0$
- $Z^{\mathrm{ub}}(\mathrm{r}) = 7$

How to estimate UB?

- How often does A appear?

	s_1	s_2	s_3	min
A	3	3	2	2
B	2	3	4	2
C	3	2	2	2
D	1	1	1	1
			Σ	7

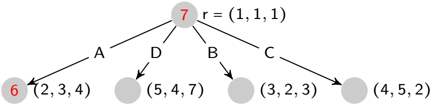
Exact MDDs for the LCS Problem

A*-based Compilation

s_1: A B C D A A B C C

s_2: B A D C B A A B C

s_3: C B A B B D A B C



Priority function

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

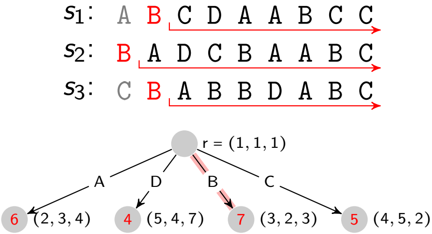
Evaluate $f(2,3,4)$

- $Z^{\mathrm{lp}}(2,3,4) = 1$
- $Z^{\mathrm{ub}}(2,3,4) = 5$

	s_1	s_2	s_3	min
A	2	2	1	1
B	2	2	3	2
C	3	2	1	1
D	1	1	1	1
			Σ	5

Exact MDDs for the LCS Problem

A*-based Compilation

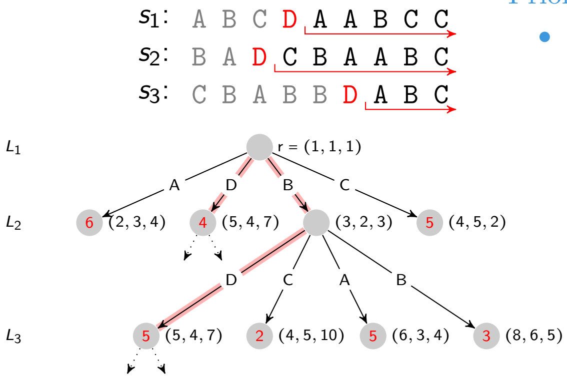


Priority function

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

Exact MDDs for the LCS Problem

A*-based Compilation

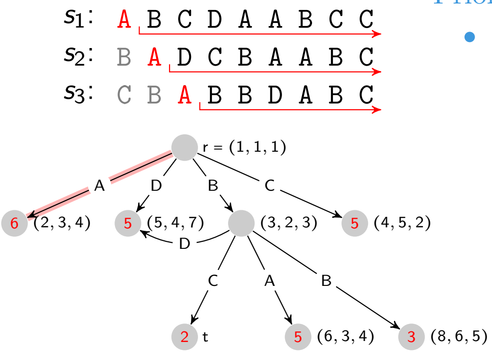


Priority function

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

Exact MDDs for the LCS Problem

A*-based Compilation

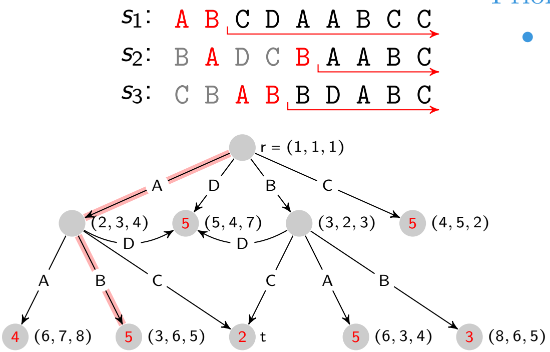


Priority function

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

Exact MDDs for the LCS Problem

A*-based Compilation



Priority function

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

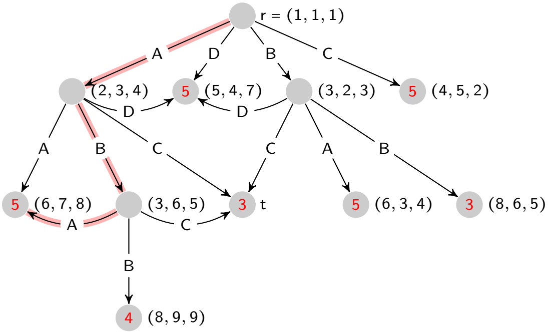
Exact MDDs for the LCS Problem

A*-based Compilation

s_1: A B C D A A B C C

s_2: B A D C B A A B C

s_3: C B A B B D A B C



Priority function

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

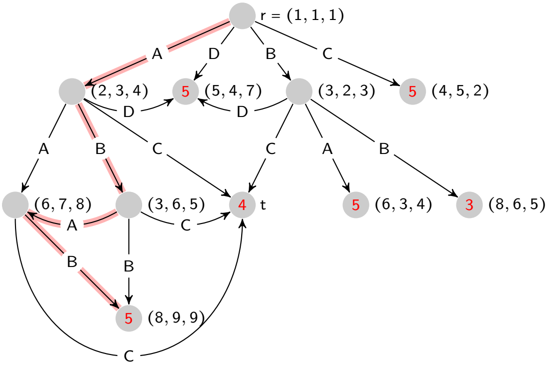
Exact MDDs for the LCS Problem

A*-based Compilation

s_1: A B C D A A B C C

s_2: B A D C B A A B C

s_3: C B A B B D A B C



Priority function

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

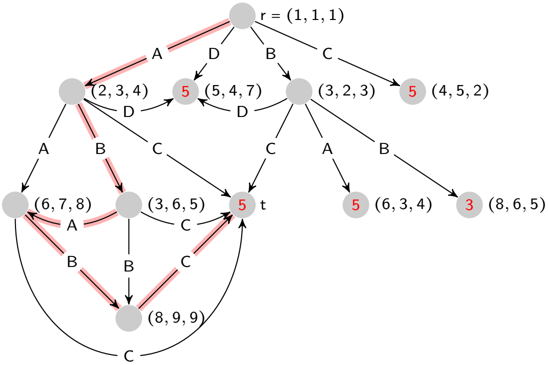
Exact MDDs for the LCS Problem

A*-based Compilation

s_1: A B C D A A B C C

s_2: B A D C B A A B C

s_3: C B A B B D A B C



Priority function

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

Compilation of Relaxed MDDs

Modified A* Search

Classical informed search algorithm for path planning

- uses a heuristic function Z^{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: $Q = \{\mathrm{r}\}$
Repeat:

- pop node $u \in Q$ with maximum $f(u)$
- if $u = \mathrm{t}$ then (terminate) $Z^{\mathrm{ub}}_{\mathrm{min}} := Z^{\mathrm{lp}}(\mathrm{t})$ is a feasible upper bound
- expand u: determine successor nodes
- if $|Q| > \phi$ then reduce Q by merging nodes
- if Q empty then terminate (complete relaxed DD)

Relaxed MDDs for the LCS Problem

A*-based Compilation

s_1: A B C D A A B C C

s_2: B A D C B A A B C

s_3: C B A B B D A B C

7 r = (1, 1, 1)

Priority function

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

Relaxed DD

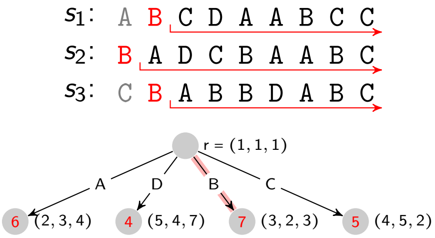
- merge if $|Q| > \phi$
- Example $\phi = 5$

Current size

- $|Q| = 1$

Relaxed MDDs for the LCS Problem

A^*-based Compilation



Priority function

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

Relaxed DD

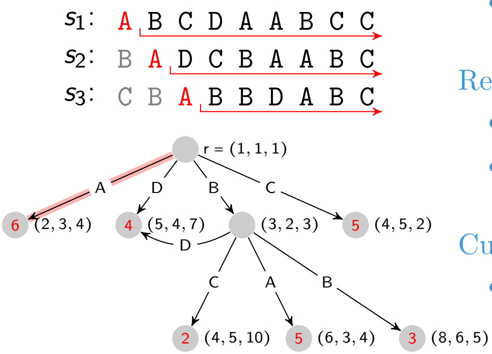
- merge if $|Q| > \phi$
- Example $\phi = 5$

Current size

- $|Q| = 4$

Relaxed MDDs for the LCS Problem

A*-based Compilation



Priority function

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

Relaxed DD

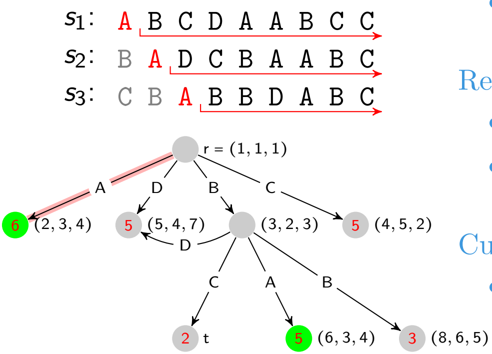
- merge if $|Q| > \phi$
- Example $\phi = 5$

Current size

- $|Q| = 6 > \phi = 5$

Relaxed MDDs for the LCS Problem

A*-based Compilation



Priority function

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

Relaxed DD

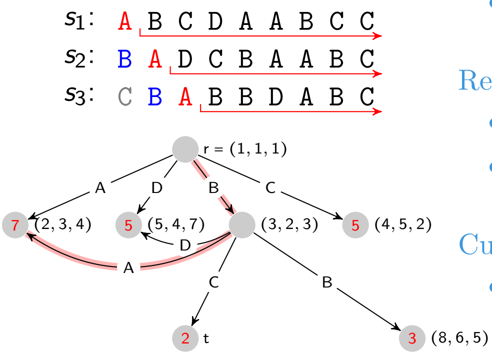
- merge if $|Q| > \phi$
- Example $\phi = 5$

Current size

- $|Q| = 6 > \phi = 5$

Relaxed MDDs for the LCS Problem

A*-based Compilation



Priority function

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

Relaxed DD

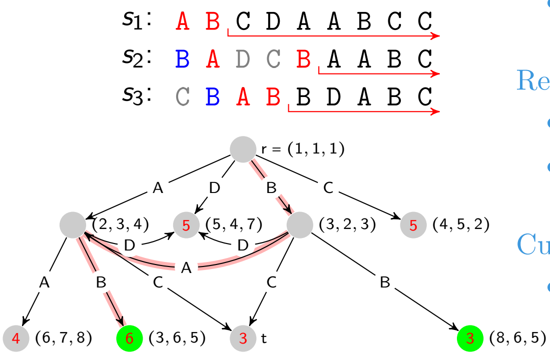
- merge if $|Q| > \phi$
- Example $\phi = 5$

Current size

- $|Q| = 5$

Relaxed MDDs for the LCS Problem

A*-based Compilation



Priority function

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

Relaxed DD

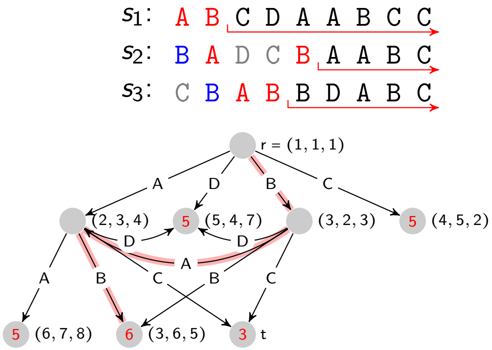
- merge if $|Q| > \phi$
- Example $\phi = 5$

Current size

- $|Q| = 6$

Relaxed MDDs for the LCS Problem

A*-based Compilation



Priority function

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

Relaxed DD

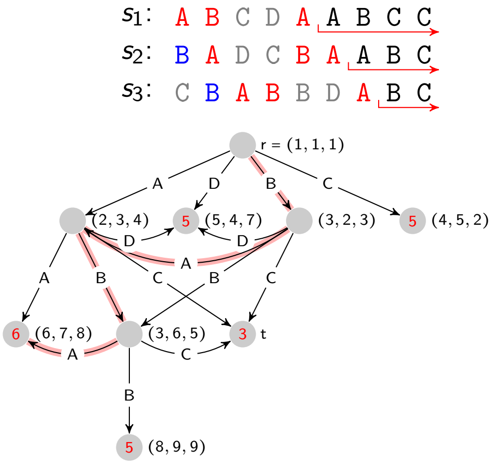
- merge if $|Q| > \phi$
- Example $\phi = 5$

Current size

- $|Q| = 5$

Relaxed MDDs for the LCS Problem

A*-based Compilation



Priority function

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

Relaxed DD

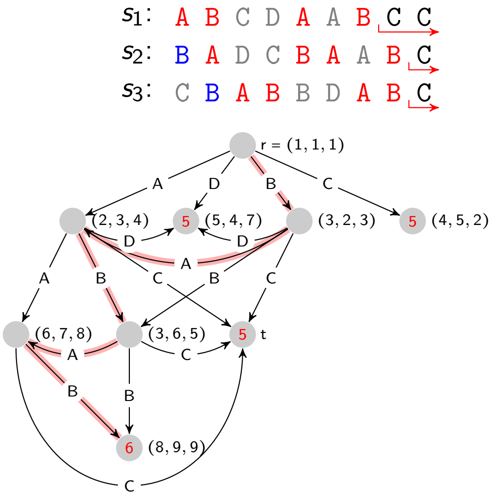
- merge if $|Q| > \phi$
- Example $\phi = 5$

Current size

- $|Q| = 5$

Relaxed MDDs for the LCS Problem

A*-based Compilation



Priority function

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

Relaxed DD

- merge if $|Q| > \phi$
- Example $\phi = 5$

Current size

- $|Q| = 4$

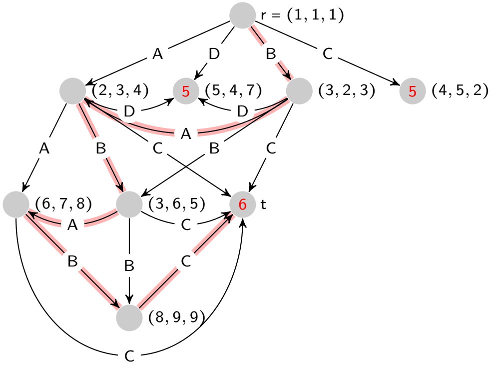
Relaxed MDDs for the LCS Problem

A*-based Compilation

s_1: A B C D A A B C C

s_2: B A D C B A A B C

s_3: C B A B B D A B C



Priority function

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

Relaxed DD

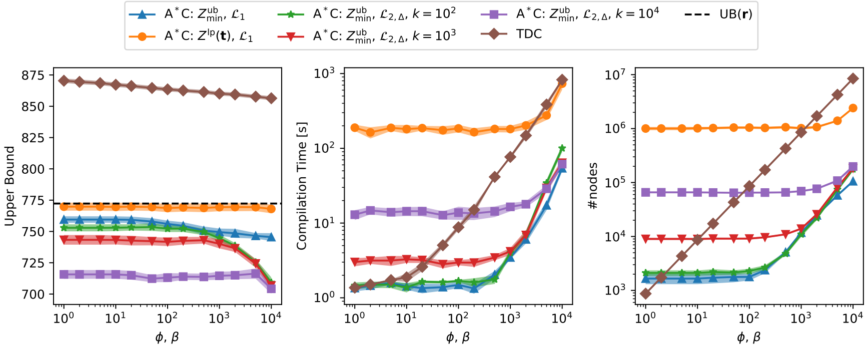
- merge if $|Q| > \phi$
- Example $\phi = 5$

Current size

- $|Q| = 3$

Results

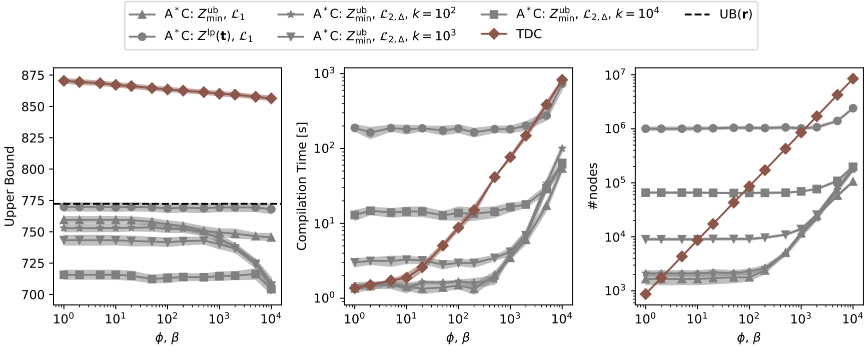
Impact of Parameters ϕ and β



- Benchmark set: BB, $m = 100$, $n = 1000$, $|\Sigma| = 8$, ten instances
- Parameter ϕ (A*C): open list size is limited by ϕ
- Parameter β (TDC): layer size is limited by β

Results

Impact of Parameters ϕ and β



- Benchmark set: BB, $m = 100$, $n = 1000$, $|\Sigma| = 8$, ten instances
- Parameter ϕ (A*C): open list size is limited by ϕ
- Parameter β (TDC): layer size is limited by β

Results

State-of-the-art Comparison

Current state-of-the-art results: Djukanovic, Raidl, Blum (2020)

- Title: Finding longest common subsequences: New anytime A^* search results
- Journal: Applied Soft Computing

- Upper bounds obtained from relaxed DDs compiled with A^*C are
  - stronger in 25.1% of the cases and
  - stronger or equally strong in 31.3% of the cases

Conclusions

- A* based construction (A*C) algorithm for relaxed MDDs
  - restrict the number of nodes in the open list
  - requires no concept of layers
- Considered two NP-hard optimization problems
  - prize collecting scheduling problem
  - longest common subsequence problem
- Experimental Results showed that
  - A*C provides more compact relaxed MDDs that
  - are significantly stronger
  - in shorter time than relaxed MDDs obtained from TDC