Solving a Generalized Constrained Longest Common Subsequence Problem

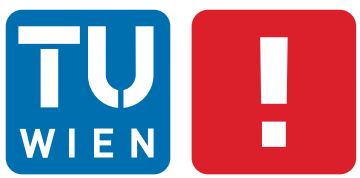
– Exact and Heuristic Methods –

A trial talk for OPTIMA 20

M. Djukanovic[1], C. Berger, G. Raidl[1], C. Blum[2]

[1]Institute of Logic and Computation, TU Wien, Vienna, Austria,
[2] Artificial Intelligence Research Institute (IIIA) Spanish National Research Council (CSIC)

September 15, 2020



Introduction

A *string* is a finite sequence of characters over some (finite) alphabet Σ.

Strings are commonly as: models for presenting DNA molecules, proteins, RNA molecules, texts, etc.

Bioinformatics and strings:

- finding similarities between molecules: understanding of biological processes (diseases, developmental defects, etc.)
- (discrete) optimization problems

Longest Common Subsequence (LCS) Problem

A *subsequence* of string s is any sequence of characters obtained by deleting zero or more characters from s.

LCS Problem:

- Input: set of strings $S = \{s_1, \ldots, s_m\}, m \in \mathbb{N}$, alphabet Σ
- Objective: find a *subsequence* of *maximum* length that is *common* for all strings from S
- $\mathcal{NP}$–hard problem for arbitrary S

Constrained Longest Common Subsequence (CLCS)

CLCS Problem:

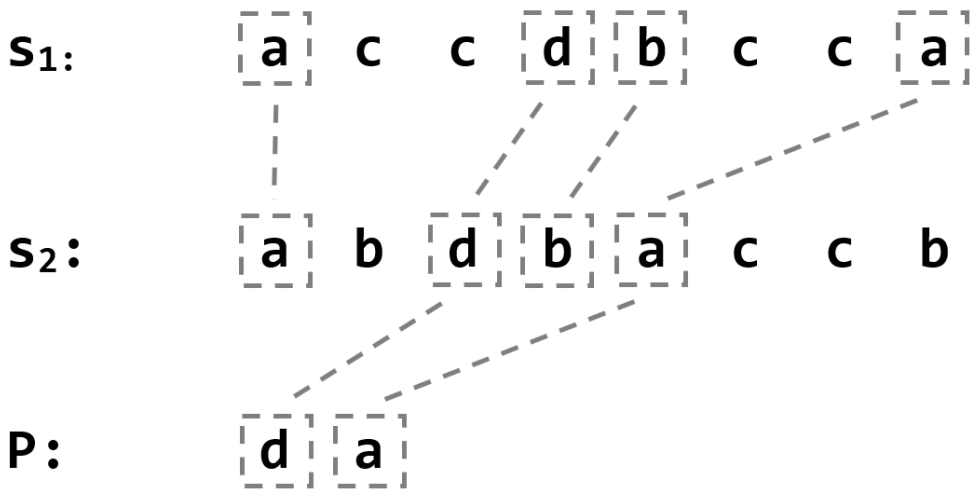
- Input: a set of strings $S = \{s_1, \ldots, s_m\}$, $m \in \mathbb{N}$, and alphabet Σ, and a pattern string P.
- Objective: find a *subsequence* of *maximum* length that is *common* for all strings from S and has P as its subsequence.

Constrained Longest Common Subsequence (CLCS)

CLCS Problem:

- Input: a set of strings $S = \{s_1, \ldots, s_m\}$, $m \in \mathbb{N}$, and alphabet Σ, and a pattern string P.
- Objective: find a *subsequence* of *maximum* length that is *common* for all strings from S and has P as its subsequence.

Example:



Literature Overview

Practical relevance: identifying homology between biological sequences which have a specific or putative structure in common

2–CLCS problem:

- Introduced by Tsai (2003), several algorithms exist
- *Polynomially* solvable by dynamic programming in $O(|s_1| \cdot |s_2| \cdot |P|)$
- A few sparse DP approaches in $O(\mathcal{M} \cdot |P|)$ time

m–CLCS problem:

- $\mathcal{NP}$-hard if m arbitrary
- Approximation algorithm by Gotthilf et al. (2008)
  - appr. factor of $\sqrt{l_{min} \cdot |\Sigma|}$ where $l_{min} = \min\{|s_1|, \ldots, |s_m|\}$

Outline of the paper:

- **Greedy Heuristic:**
  Constructive heuristic to quickly obtain solutions of reasonable quality;
- **Beam Search:**
  Algorithm to tackle large instances heuristically;
- **A* Search:**
  Exact method to solve small-to-medium instances to proven optimality.

State Graph for the m–CLCS problem

Instance:

s_1 = **bcaacbdba**
s_2 = **cbccadcbbd**
P = **cbb**

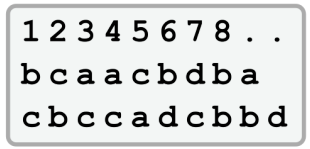
((1,1),0,0)

Node v = $(p^{L,v}, l^v, u^v)$

Search Space for the *m*–CLCS problem

Instance:

s_1 = **bcaacbdba**
s_2 = **cbccadcbbd**
P = **cbb**



((1,1),0,0)

Node v = $(p^{L,v}, l^v, u^v)$

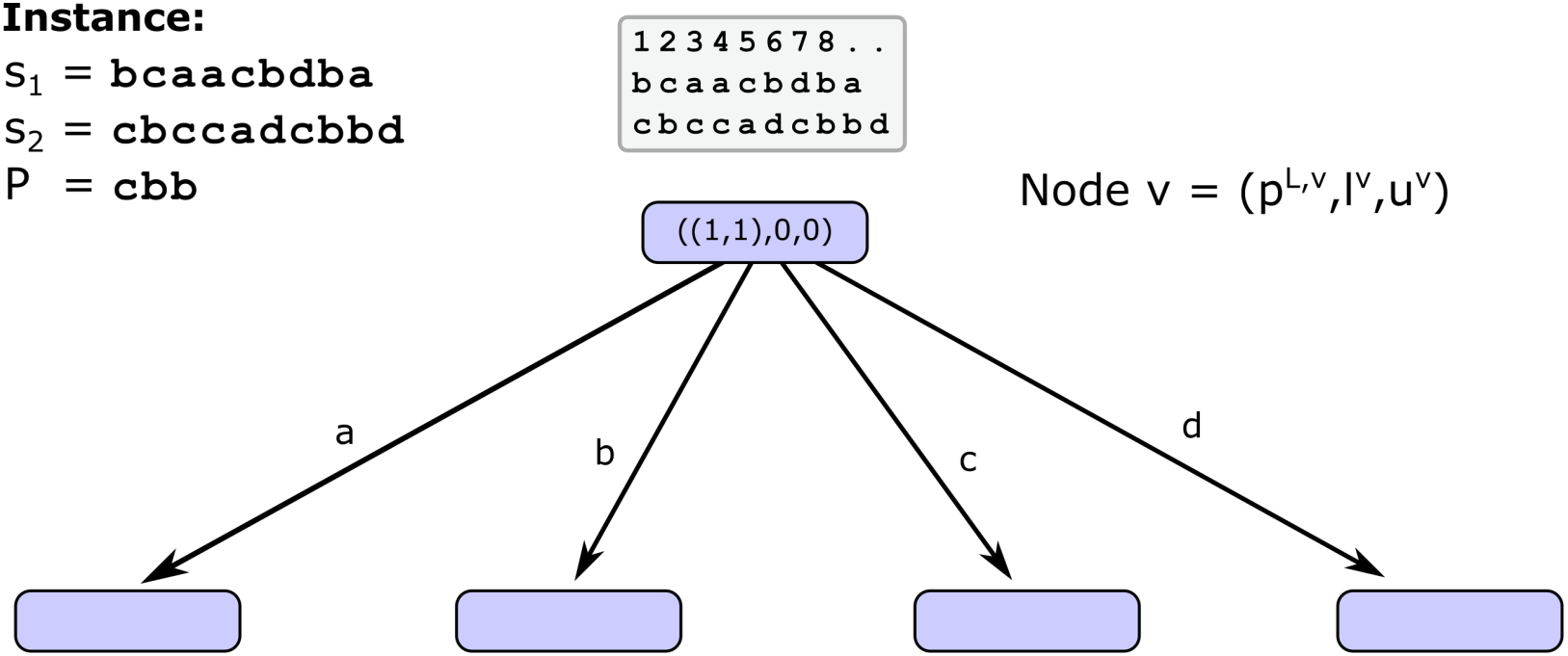
Search Space for the *m*–CLCS problem

Instance:

s_1 = **bcaacbdba**

s_2 = **cbccadcbbd**

P = **cbb**



Node v = $(p^{L,v}, l^v, u^v)$

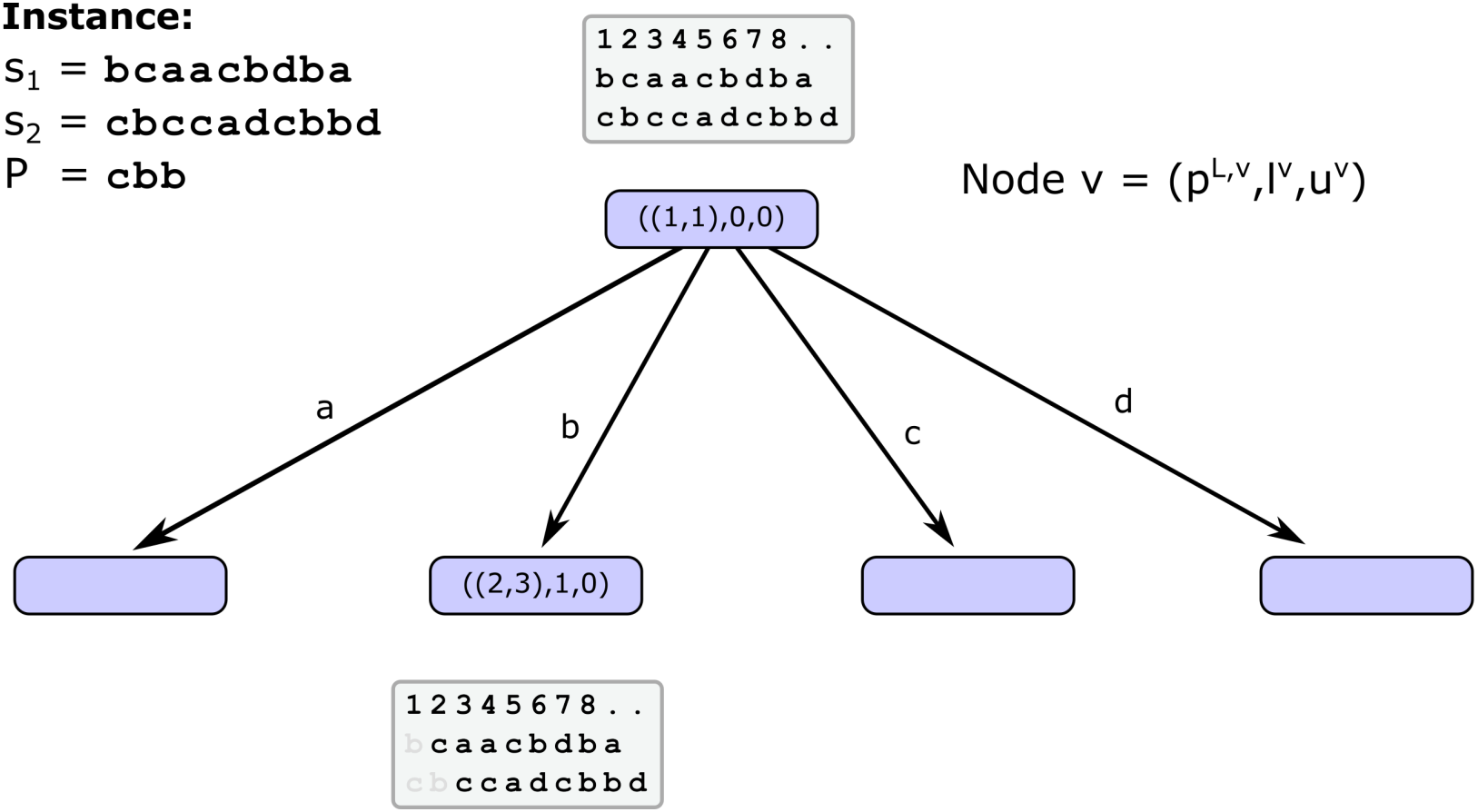
Search Space for the *m*–CLCS problem

Instance:

s_1 = **bcaacbdba**

s_2 = **cbccadcbbd**

P = **cbb**



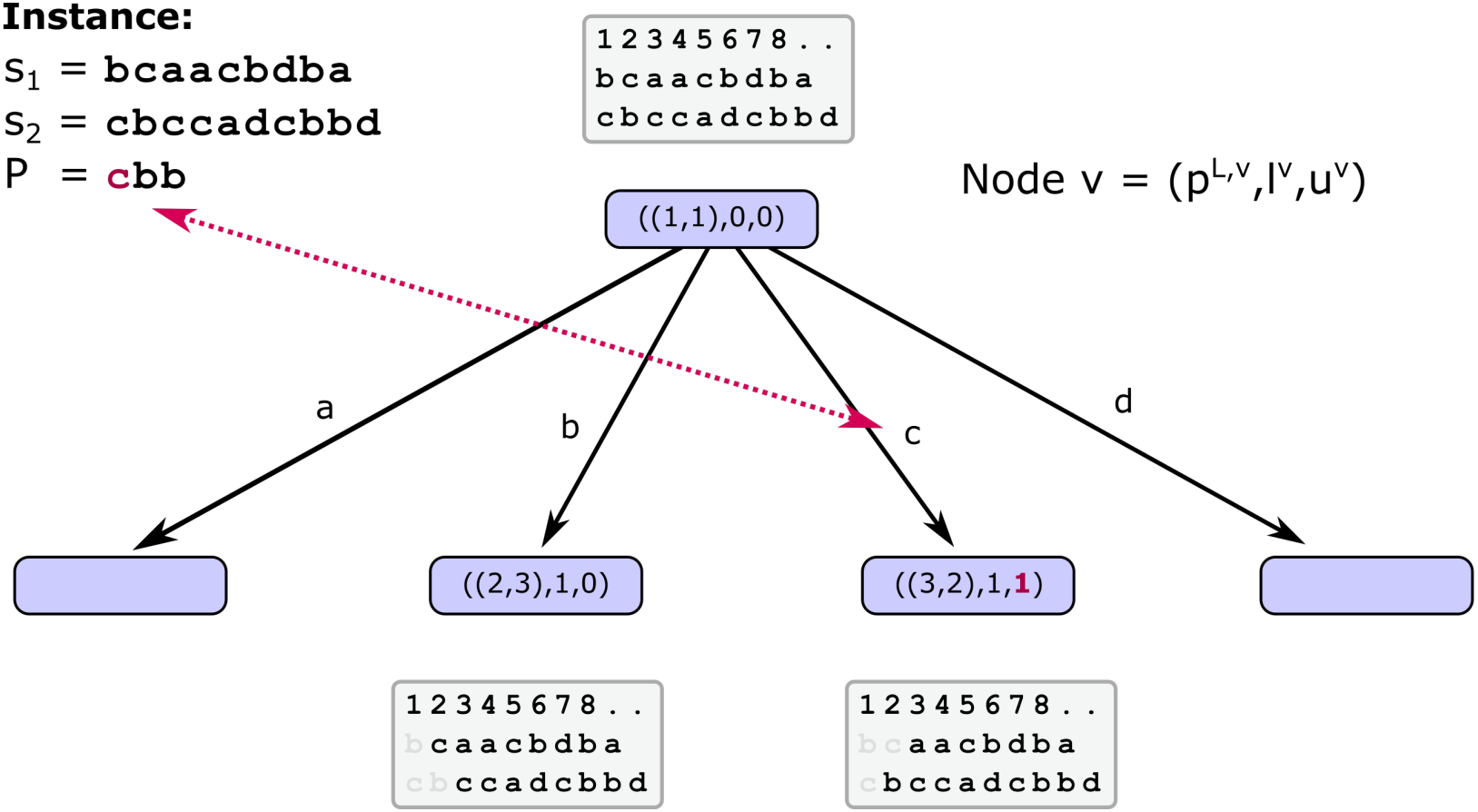
Search Space for the m–CLCS problem

Instance:

s_1 = **bcaacbdba**

s_2 = **cbccadcbbd**

P = **cbb**



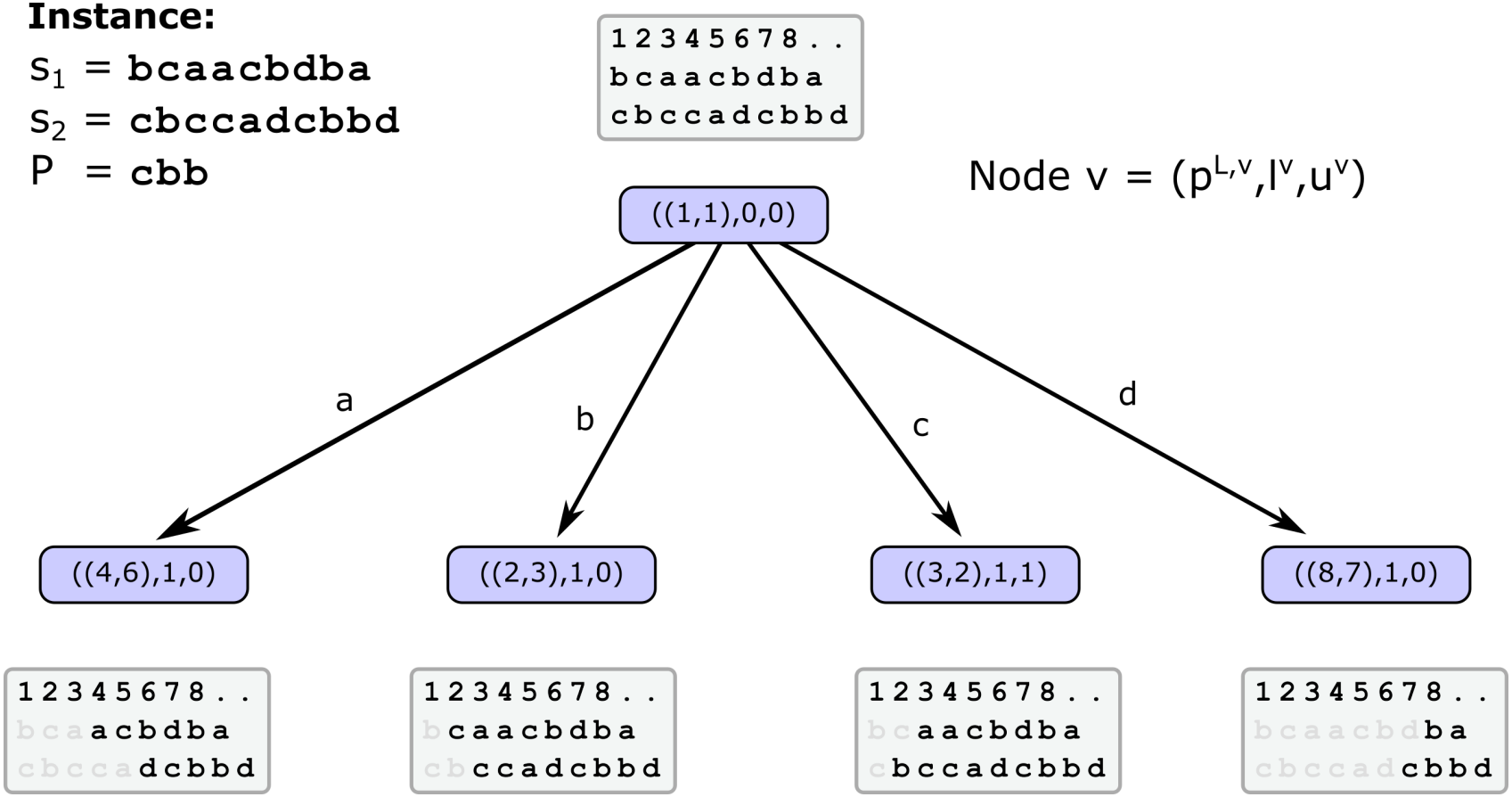
Search Space for the *m*–CLCS problem

Instance:

s_1 = **bcaacbdba**

s_2 = **cbccadcbbd**

P = **cbb**



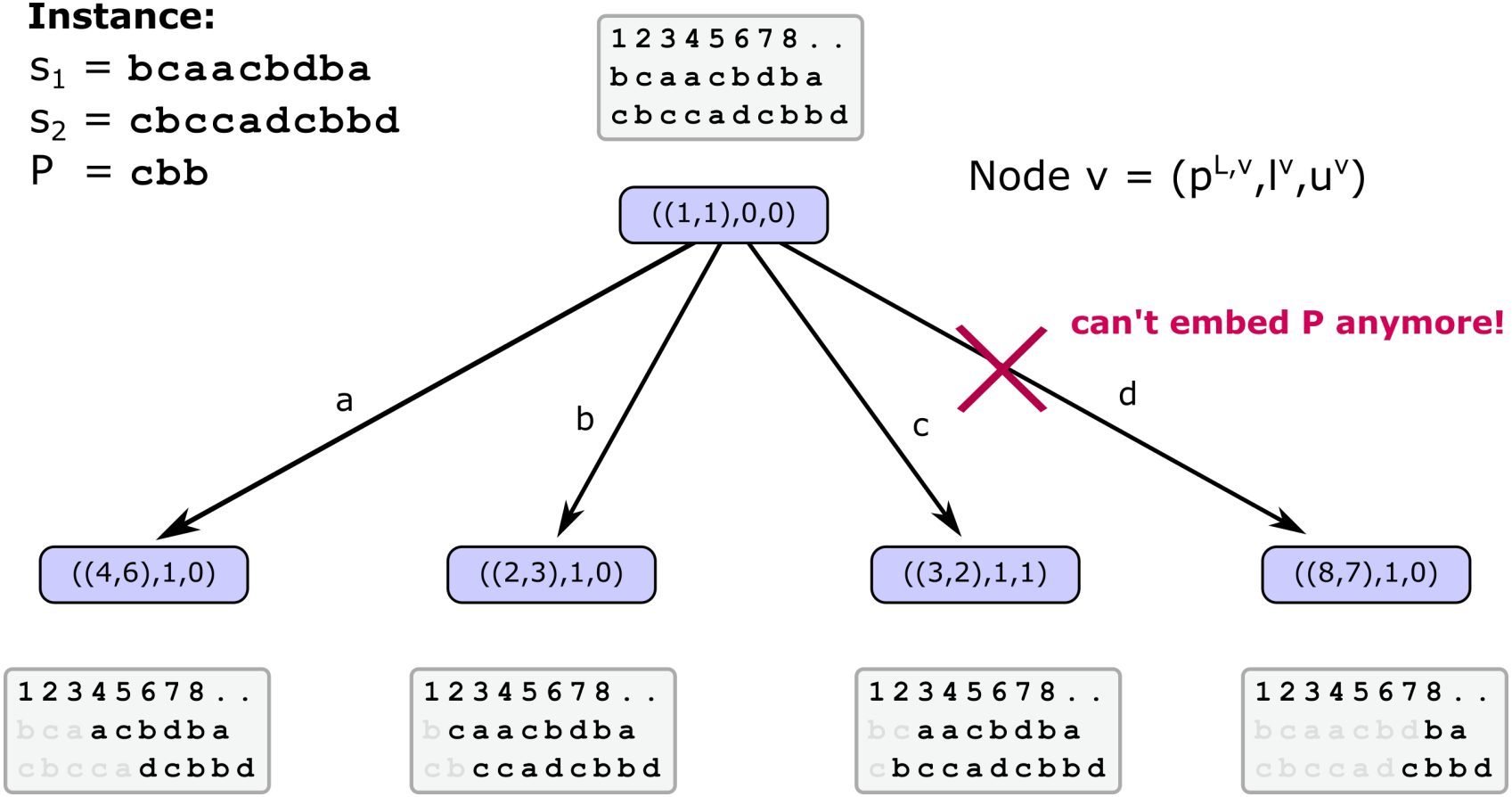
Search Space for the m–CLCS problem

Instance:

s_1 = **bcaacbdba**

s_2 = **cbccadcbbd**

P = **cbb**



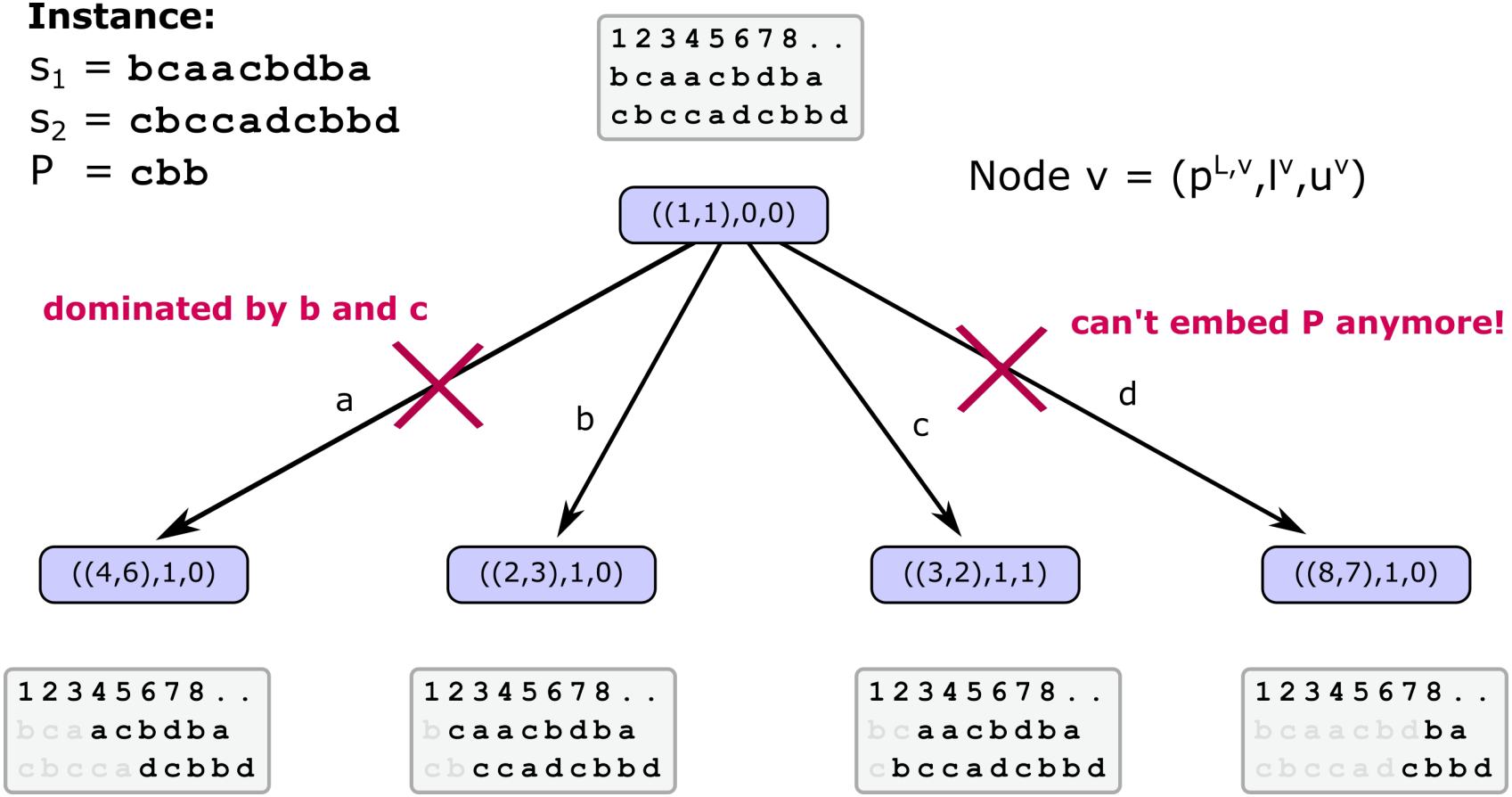
Search Space for the m–CLCS problem

Instance:

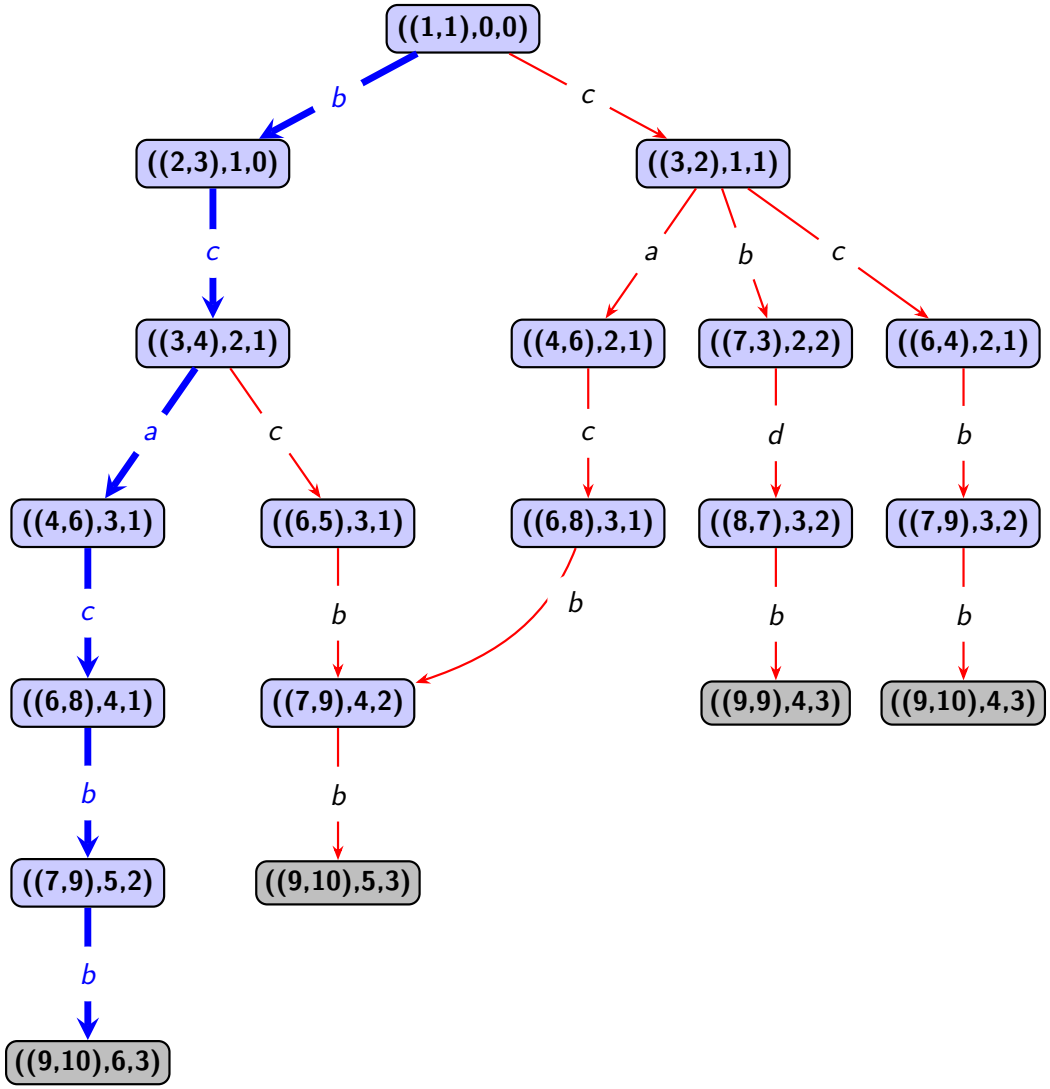
s_1 = **bcaacbdba**

s_2 = **cbccadcbbd**

P = **cbb**



Full State Graph Example



Instance: $s_1 = \texttt{bcaacbdba}$, $s_2 = \texttt{cbccadcbbd}$, $P = \texttt{cbb}$

Greedy Heuristic for the m–CLCS problem

Constructive heuristic based on Best-Next heuristic:

- starts with an empty partial solution $s^P = \varepsilon$
- at each step, the most-promising letter is appended to s^P:

$$g(p^{\mathrm{L},v}, u^v, c) = pen(u^v, c) + \sum_{i=1}^{m} \frac{Succ[i, p_i^{\mathrm{L},v}, c] - p_i^{\mathrm{L},v} + 1}{|s_i| - p_i^{\mathrm{L},v} + 1}$$

$$\text{where } pen(u^v, c) = \frac{1}{|P| - u^v + \mathbf{1}_{P[u^v+1]=c}}$$

- stops once no more letters can be added

Beam Search (BS) for the m–CLCS problem

Incomplete breadth-first-search, at each level of the search tree:

- all nodes are expanded in all possible ways
- at most β nodes are kept w.r.t. an heuristic rule

Further reduction of nodes through (optional):

- Pruning: compare UB of nodes with so-far best found solution
- Filtering: check dominance-relation (based on reduced filtering, controlled by k_{best} param.)

Heuristic Guidance of BS

Four options were developed:

- Upper Bounds on the length of a LCS
- Probability–Based Heuristic
- Expected Length Calculation Heuristic
- Pattern Ratio Heuristic

Upper Bounds for m–CLCS Problem

- UB_1 bound: based on number of occurrences of each letters in each optimal solution

- UB_2 bound: DP–based upper bound calculating LCS for each 2 consecutive strings s_i and s_{i+1} and taking the minimum of the values

BS: Heuristic Guidance

Probability–Based Heuristic:

- $\Pr(p, q)$: probability that any string of length p is a subsequence of a random string of length q (Mousavi and Tabataba, 2012)
- Assuming independence among input strings, nodes are evaluated by

$$H(v) = \prod_{i=1}^{m} \Pr(p, |s_i| - p_i^{\mathrm{L},v} + 1)$$

- p is heuristically determined at each level of BS by:

$$p = p^{\min} + \min_{v \in V_{\mathrm{ext}}} \left\lfloor \frac{\min_{i=1,\ldots,m} \left\{ |s_i| - p_i^{\mathrm{L},v} + 1 \right\} - p^{\min}}{|\Sigma|} \right\rfloor$$

with $p^{\min} = \min_{v \in V_{\mathrm{ext}}} \{|P| - u^v\}$

BS: A Novel Heuristic Guidance

Expected Length Calculation Heuristic:

- State-of-the-art for LCS problem, extended towards CLCS problem
- an approximation of the expected length of a CLCS by

$$\mathrm{Ex}(v) = l_{\min} - \sum_{k=|P|-u^v+1}^{l_{\min}} \left(1 - \left(\prod_{i=1}^{m} \Pr(k, |s_i| - p_i^{\mathrm{L},v} + 1)\right) \cdot \Pr(|P| - u^v, k)\right)^{|\Sigma|^k}$$

with $l_{\min}(v) = \min_{i=1,\ldots,m}\{|s_i| - p_i^{\mathrm{L},v} + 1\}$

Calculated in expected $O(m \log n)$ time (divide-and-conquer)

BS: A Novel Heuristic Guidance

Expected Length Calculation Heuristic:

- State-of-the-art for LCS problem, extended towards CLCS problem
- an approximation of the expected length of a CLCS by

$$\mathrm{Ex}(v) = l_{\min} - \sum_{k=|P|-u^v+1}^{l_{\min}} \left(1 - \left(\prod_{i=1}^{m} \Pr(k, |s_i| - p_i^{\mathrm{L},v} + 1)\right) \cdot \Pr(|P| - u^v, k)\right)^{|\Sigma|^k}$$

with $l_{\min}(v) = \min_{i=1,\ldots,m}\{|s_i| - p_i^{\mathrm{L},v} + 1\}$

Calculated in expected $O(m \log n)$ time (divide-and-conquer)

Pattern Ratio Heuristic:

- evaluate nodes by

$$R(v) = \frac{l_{\min}}{|P| - u^v + 1}$$

and an additional tie-breaking criterion (Euclidean norm)

A* Search for the *m*–CLCS problem

- Finding paths in weighted graphs, by Hart et al. (1968)
- Best-first-search: always the *most-promising* node is expanded by considering all its possible successor nodes
- Maintains a list of open (not-yet-expanded) nodes, ranked acc. to $f(v) = g(v) + h(v)$
  - $g(v)$: cost of best known path from start node to v
  - $h(v)$: estimated cost from v to a goal node
- If $h(v)$ is admissible, an optimal solution is found when a goal node is selected for expansion
- We set $g(v) := l^v$, $h(v) := \min\{\mathrm{UB}_1(v), \mathrm{UB}_2(v)\}$

Experimental Studies

Settings:

- Implementation in $C++$, GCC 7.4 with highest available optimization level
- Single-threaded mode on an Intel Xeon E5-2640 with 2.40GHz, 16 GB of memory

Benchmarks: We generated 10 instances for each combination of

- $n \in \{100, 500, 1000\}$
- $m \in \{10, 50, 100\}$
- $|\Sigma| \in \{4, 20\}$
- and $(|P|/n) \in \left\{\frac{1}{50}, \frac{1}{20}, \frac{1}{10}, \frac{1}{4}, \frac{1}{2}\right\}$

which gives us 900 problem instances.

Computational Results for the m–CLCS Problem

We compare

- Approx: approximation algorithm from Gotthilf (2008)
- Greedy heuristic

- Four beam search configurations:
  - BS-Prob
  - BS-Ex
  - BS-Pat
  - BS-Ub

Comparison of BS Configurations

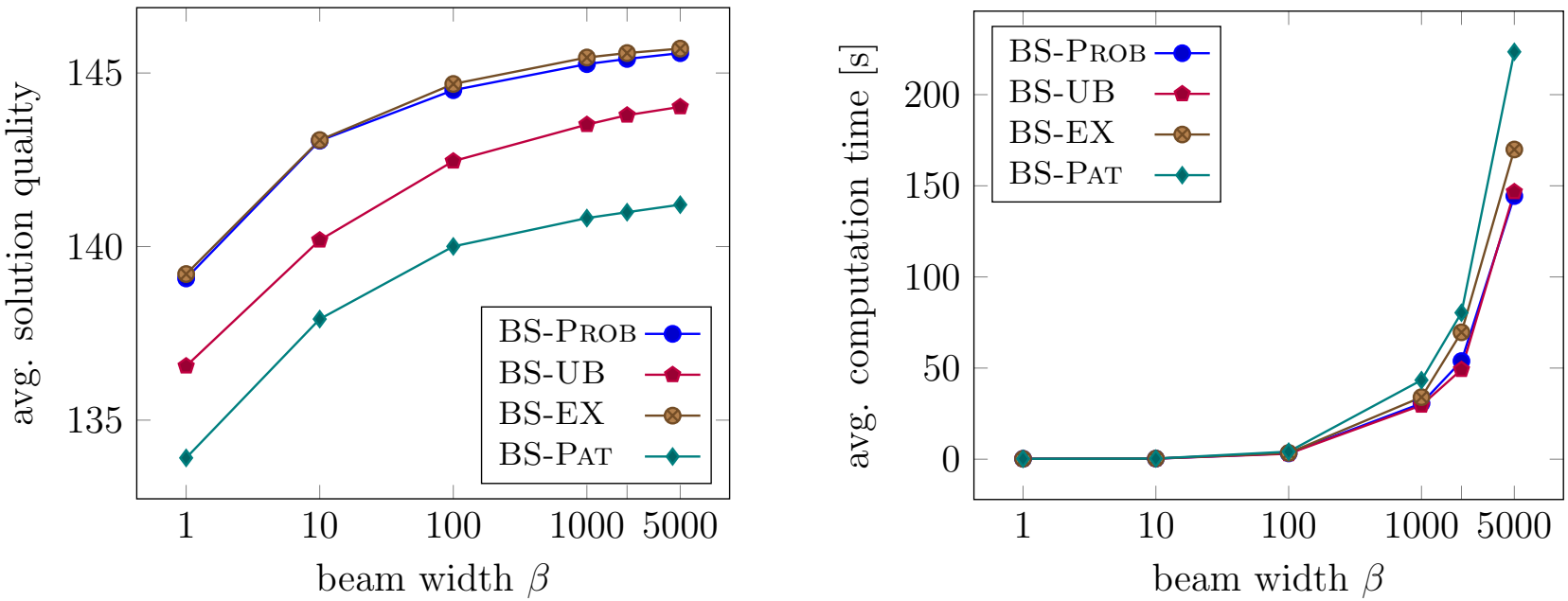


Figure: Results of BS with $k_{\text{best}} = 100$ and varying β

$\Longrightarrow$ We choose $\beta = 2000$, $k_{\text{best}} = 100$, and Prune().

Computational Results for the m–CLCS Problem

Table: Instances with $p' = \frac{|P|}{n} = \frac{1}{50}$

			Approx		Greedy		BS-Ub		BS-Prob		BS-Ex		BS-Pat		A*	
$\|\Sigma\|$	m	n	$\overline{\|s\|}$	$\bar{t}$	$\overline{\|s\|}$	$\bar{t}$	$\overline{\|s\|}$	$\bar{t}$	$\overline{\|s\|}$	$\bar{t}$	$\overline{\|s\|}$	$\bar{t}$	$\overline{\|s\|}$	$\bar{t}$	#	$\bar{t}$
4	10	100	20.9	<0.1	28.4	<0.1	34.2	22.9	**34.3**	20	**34.3**	20.8	33.8	26.2	7	290.4
4	10	500	117.8	<0.1	159.6	<0.1	180.4	149.1	183.6	157.3	**184.8**	143.2	177.7	174.7	0	-
4	10	1000	239.2	0.1	327.6	0.1	363.5	284.7	372.4	372.3	**376.3**	434.2	354.7	428.2	0	-
4	50	100	17.4	<0.1	20.3	<0.1	24.1	15.5	**24.2**	12.1	**24.2**	16.7	24	22	0	-
4	50	500	109.3	0.1	125.3	0.1	137.3	106	140.4	138.1	**141.8**	131.8	136.3	147.2	0	-
4	50	1000	228.9	0.5	263.6	0.5	279.8	257.9	288.7	231.1	**290.4**	340.0	277.2	251.7	0	-
4	100	100	17.0	<0.1	18.9	<0.1	**21.9**	16.1	**21.9**	16.3	**21.9**	14	21.6	19.4	0	-
4	100	500	108.1	0.2	118.5	0.2	128.4	135	131	118.2	**132.0**	115.2	127.6	160.2	0	-
4	100	1000	225.1	0.9	248.7	0.8	262.4	287.6	270.5	236.6	**272.1**	329.9	261.6	282	0	-
20	10	100	4.3	<0.1	6.2	<0.1	***7.9**	0.1	***7.9**	0.1	***7.9**	0.1	***7.9**	0.1	10	<0.1
20	10	500	23.8	<0.1	40.7	<0.1	48.9	104.5	49.7	137	**50.4**	183.8	41.9	221.7	0	-
20	10	1000	48.9	0.1	82.5	0.1	97.7	246.8	102.0	280.7	**104.9**	344.3	85.6	551.4	0	-
20	50	100	2.8	<0.1	***3.1**	<0.1	***3.1**	<0.1	***3.1**	<0.1	***3.1**	<0.1	***3.1**	<0.1	10	<0.1
20	50	500	20.0	0.1	24	0.1	28.3	49	**28.8**	46.8	**28.8**	100.3	26	135.5	0	-
20	50	1000	42.6	0.5	53.7	0.5	59.6	152.5	61.4	158.1	**62.3**	245.4	55.1	211.2	0	-
20	100	100	2.3	<0.1	***2.4**	<0.1	***2.4**	<0.1	***2.4**	<0.1	***2.4**	<0.1	***2.4**	<0.1	10	<0.1
20	100	500	18.5	0.3	22	0.3	24.7	60.9	**25.2**	62.6	25.0	118.5	22.8	82.7	0	-
20	100	1000	41.1	1	49	0.8	52.8	166.2	54.7	188.6	**55.0**	334.8	50	342.7	0	-

*The algorithm reached the optimal solution values from A* search

Computational Results for the m–CLCS Problem

Table: Instances with $p' = \frac{|P|}{n} = \frac{1}{4}$

			Approx		Greedy		BS-Ub		BS-Prob		BS-Ex		BS-Pat		A*	
$\|\Sigma\|$	m	n	$\overline{\|s\|}$	$\bar{t}$	$\overline{\|s\|}$	$\bar{t}$	$\overline{\|s\|}$	$\bar{t}$	$\overline{\|s\|}$	$\bar{t}$	$\overline{\|s\|}$	$\bar{t}$	$\overline{\|s\|}$	$\bar{t}$	#	$\bar{t}$
4	10	100	28.6	<0.1	32.3	<0.1	***34.5**	1.1	***34.5**	0.9	***34.5**	1.0	***34.5**	1.5	10	0.2
4	10	500	134.3	<0.1	159.8	<0.1	179.3	45.6	**182.4**	48.8	181.1	98.0	168.6	97	1	660.8
4	10	1000	264.7	0.1	317.2	0.1	350.3	76.8	**361.7**	108	361.4	249.4	330.8	220.2	0	-
4	50	100	26.4	<0.1	26.9	<0.1	***27.5**	<0.1	***27.5**	<0.1	***27.5**	<0.1	***27.5**	<0.1	10	<0.1
4	50	500	130.1	0.1	139.5	0.1	146.2	33.6	**148.3**	28	146.3	19.9	142.7	55.9	0	-
4	50	1000	257.4	0.5	277.3	0.5	291.9	73.6	**296.4**	63.6	289.5	41.1	284.2	107.6	0	-
4	100	100	25.9	<0.1	26.2	<0.1	***26.5**	<0.1	***26.5**	<0.1	***26.5**	<0.1	***26.5**	<0.1	10	<0.1
4	100	500	128.9	0.2	135.8	0.2	140.4	24.6	**140.8**	34.8	140.3	17.4	137.3	45.9	0	-
4	100	1000	256.4	0.8	270.7	0.7	279.7	56.4	**282.5**	73.4	279.0	40.4	273.3	122	0	-
20	10	100	***25.0**	<0.1	***25.0**	<0.1	***25.0**	<0.1	***25.0**	<0.1	***25.0**	<0.1	***25.0**	<0.1	10	<0.1
20	10	500	***125.0**	<0.1	***125.0**	<0.1	***125.0**	<0.1	***125.0**	<0.1	***125.0**	<0.1	***125.0**	<0.1	10	<0.1
20	10	1000	***250.0**	0.1	***250.0**	0.1	***250.0**	0.1	***250.0**	0.1	***250.0**	0.1	***250.0**	0.1	10	0.1
20	50	100	***25.0**	<0.1	***25.0**	<0.1	***25.0**	<0.1	***25.0**	<0.1	***25.0**	<0.1	***25.0**	<0.1	10	<0.1
20	50	500	***125.0**	0.1	***125.0**	0.1	***125.0**	0.2	***125.0**	0.2	***125.0**	0.1	***125.0**	0.1	10	0.1
20	50	1000	***250.0**	0.5	***250.0**	0.4	***250.0**	0.4	***250.0**	0.5	***250.0**	0.5	***250.0**	0.5	10	0.5
20	100	100	***25.0**	<0.1	***25.0**	<0.1	***25.0**	<0.1	***25.0**	<0.1	***25.0**	<0.1	***25.0**	<0.1	10	<0.1
20	100	500	***125.0**	0.3	***125.0**	0.3	***125.0**	0.3	***125.0**	0.3	***125.0**	0.2	***125.0**	0.2	10	0.3
20	100	1000	***250.0**	1	***250.0**	1	***250.0**	1.1	***250.0**	1.1	***250.0**	0.8	***250.0**	1.1	10	1.0

*The algorithm reached the optimal solution values from A* search

Summary

- A* search:
  - the first exact algorithm to solve m–CLCS problem
  - A* works extremely well when $|P|$ larger
- Beam search to solve large instances heuristically:
  - reach optimal solutions on many tested instances
  - BS-Prob best when $|P|$ is long
  - BS-Ex best when $|P|$ is short

Future Work

- Test our algorithm on real world data sets
  - already done for 2–CLCS
- Develop anytime algorithms for m–CLCS problem
- Solve the generalized m–CLCS problem with many constraints

Thank you for your attention!