On Solving the Multiple Variable Gapped Longest Common Subsequence Problem

Marko Djukanović[1,2,6] Nikola Balaban[2] Christian Blum[3]
Aleksandar Kartelj[4] Sašo Džeroski[5] Žiga Zebec[6]

[1]University of Nova Gorica, Nova Gorica, Slovenia
[2]Faculty of Natural Sciences and Mathematics, University of Banja Luka, Banja Luka, B&H
[3]Artificial Intelligence Research Institute (IIIA-CSIC), Barcelona, Spain
[4]Faculty of Mathematics, University of Belgrade, Belgrade, Serbia
[5]Jožef Stefan Institute, Ljubljana, Slovenia
[6]Institute of Information Sciences (IZUM), Maribor, Slovenia

– EUROCAST 2026: 20th International Conference on Computer Aided Systems Theory, February 23-27, 2026, Las Palmas de Gran Canaria, Spain –

I FEEL SLOVENIA
Sofinancira Evropska unija
REPUBLIKA SLOVENIJA MINISTRSTVO ZA VISOKO ŠOLSTVO, ZNANOST IN INOVACIJE Masarykova cesta 16, SI - 1000 Ljubljana
SMASH machine learning for science and humanities postdoctoral program

Introduction

- Objects we deal with: sequences (strings) over finite alphabet Σ
  - DNA/RNA over $\{\mathtt{A}, \mathtt{T}, \mathtt{G}, \mathtt{C/U}\}$
  - Proteins over 20 (canonical) amino acids: $\{\mathtt{A}, \mathtt{C}, \mathtt{D}, \mathtt{E}, \mathtt{P}, \mathtt{Q}...\}$
- **One of central tasks in computational biology:**
  - sequence comparison, finding common motifs between sequences
  - compare structurally $\Rightarrow$ something about functionality
- **Subsequences**: reveal structural similarities $\rightarrow$ **Longest commmon subsequence problem** variants (studied 50 years already)

Longest common subsequence problem (LCSP)

Definition (LCSP)

Input: Given an arbitrary set of sequences $S = \{s_1, \ldots, s_m\}$
Task: Find a subsequence s common for all sequences from S with maximum possible length ($|s|$).

Example

Input: $S = \{\texttt{AATTGC}, \texttt{ATTAC}\}$
LCS solution: $s = \texttt{ATTC}$

LCS: Literature & Problem Variants

- For $m = 2$, polynomially solvable: Dynamic programming, Hunt-Szymanski, Hirschenberg, ...
- When m arbitrary large – $\mathcal{NP}$-hard:
  - approximation, meta-heuristic, but also exact approaches

Problem-related practical variants:

- Repetition-free, filled LCS problem, ...
- Gapped LCS problem

The gapped LCS problem (Peng and Yang, 2012)

Definition (A gap sequence)

Given is a sequence s and an assigned function $G_s\colon \{1,\ldots,|s|\} \mapsto \mathbb{N}$. A pair (s, G_s) is called a gap sequence.

Definition (A gapped subsequence)

Sequence $\tilde{s}$ is a gapped subsequence of (s, G_s) iff

- $\tilde{s}$ is a subsequence of s
- the gap constraint G_s fulfilled w.r.t. *positions of embedding* (letters of) $\tilde{s}$ in s:
  - suppose $i_1 < \ldots < i_{|\tilde{s}|}$ are those **positions**
  - $(\forall\, j = 2,\ldots,|\tilde{s}|)\; i_j - i_{j-1} \leq G_s(i_j) + 1$ (consecutive positions respect gap distances = allowed *number of letters inbetween*)

Problem definition

Example

$s = \texttt{AATTGC}$, $G_s(\cdot) = 1$

- $\tilde{s} = \texttt{ATG}$, the embedding: AATTGC (valid gapped subsequence)
- $\tilde{s} = \texttt{ATG}$, the embedding: AATTGC (invalid gapped subsequence)

Definition (The multiple (variable) gapped LCS problem – MVGLCSP)

Input: Given is a set of gapped sequences $\{(s_1, G_{s_1}), \ldots, (s_m, G_{s_m})\}$.
Task: Find the longest subsequence $\tilde{s}$ so that

- $\tilde{s}$ is common subusequence of each s_i
- $\tilde{s}$ fulfills all gap constraints G_{s_i} $(i = 1, \ldots, m)$

Note: when $G_{s_i} = n$ (the length of longest sequence) $\Rightarrow$ VGLCSP = LCSP.

Literature & Motivation for VGLCSP

- Peng and Yang (2012, 2014): studied $m = 2$ version by **three dynamic programming** (DP) approaches (basic and two advanced)
- Manea et al. (2024): complexity analysis investigated
- NP-hard under arbitrary large m

Motivation:

- Genetics and molecular biology: DNA/protein comparison analysis where structural distances between residues must be respected
- Time-series analysis: in settings where events are required to occur within specified temporal delays (Lainscsek et al. (2015))

Methodology

- Based on the significant extension of the **state space graph formulation** for LCS problem (Djukanovic et al. (2020))
- **Gap constraints**: incorporated to cut-off invalid extensions (edges) among the LCS extensions immediately
- **Many root/source nodes** in the state graph (the position of leading letter in each common gapped subsequence free to select)

Root-based state graph formulation: rough idea

- Each **state** $v = (p^L, l^v)$: one or more feasible partial solutions where
  - a vector of positions p^L refer to the positions of suffixes of input sequences relevant to further expand these sols (subproblem)
  - the length of current partial solution l^v
- **Expansion** of v: extend (concatenate) partial solutions feasibly (and non-dominantly) by one letter in all possible ways, respecting gap constraints
- **Non-expandable nodes**: complete solutions;

Decision: select appropriate (matches) p^L for start (empty solution)

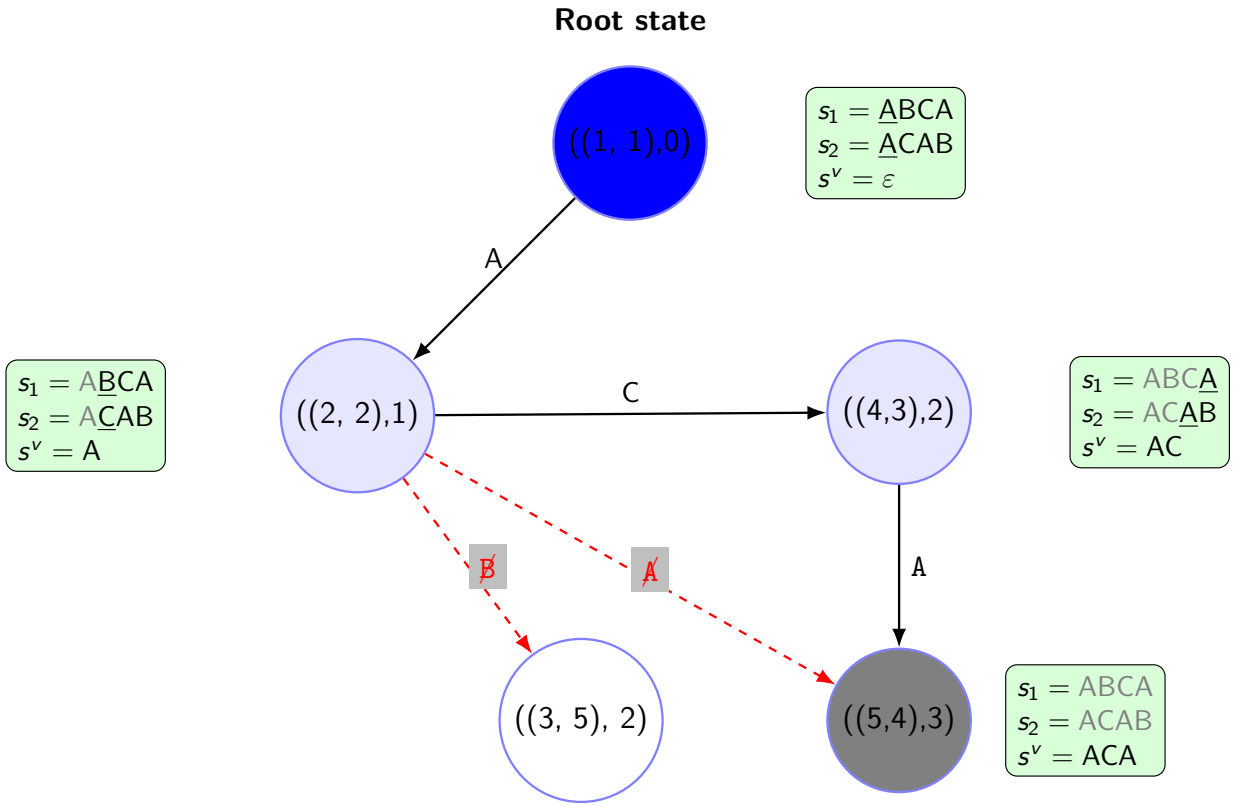
$\Rightarrow$ possibly **(exponentially) many root nodes**

Space(p^L): root-based state (sub) graph induced by root node $(p^L, 0)$.

Root-based state graph formulation: example

Given $s_1 =$ A BCA, $s_2 =$ A CAB, assuming $G_1 = G_2 = 1$, match $p^L = (1, 1)$;

Space$((1, 1))$: partial solutions start with A at $(1, 1)$

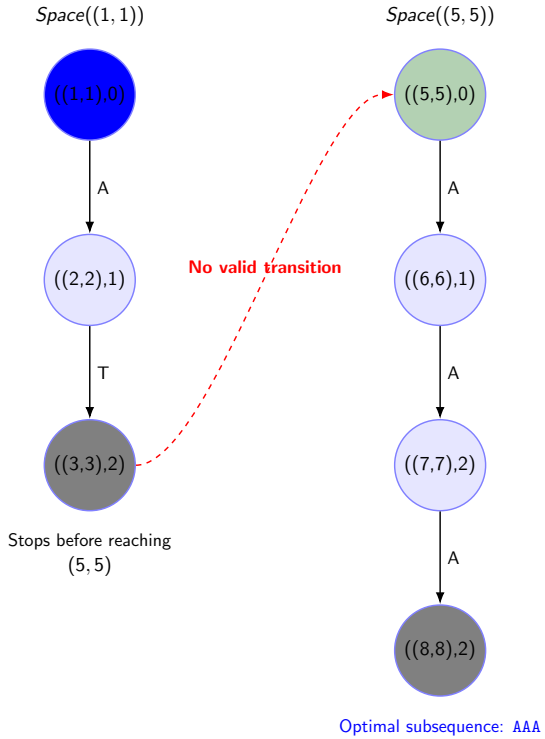


Space$((1, 1)) \Longrightarrow$ ACA

An issue with the root-state-space formulation: disconnected components

Example

$S = \{s_1 = \texttt{ATGG}\boxed{\texttt{A}}\texttt{AA},\ s_2 = \texttt{ATCC}\boxed{\texttt{A}}\texttt{AA}\}$, with $G_{s_1} = G_{s_2} = 1$.



Observation: $((5,5),*) \notin Space((1,1))$, The optimal solution: AAA.

Iterative multi-source beam search (IMSBS) strategy

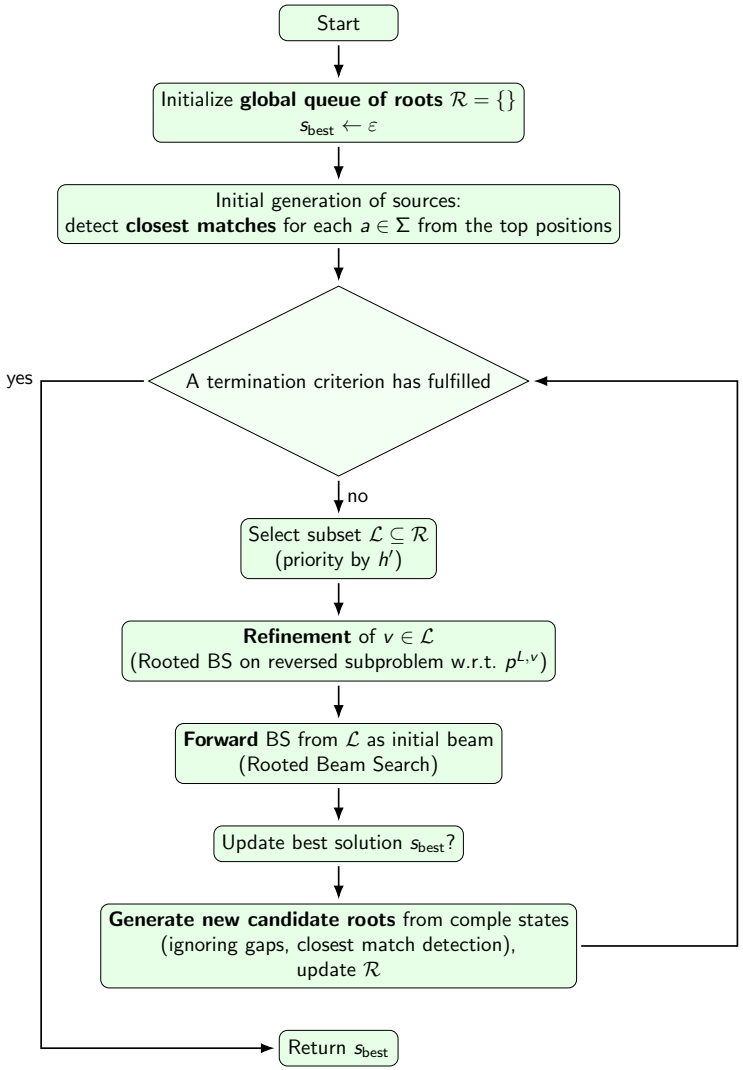
- Explicitly enumerating root states computationally prohibitive ($O(n^m)$ time)

- **IMSBS** (rough idea of exploring space of root nodes):

  1. **Exploitation** of Space(p^L): given to beam search (BS) — limited breadth-first-search (BFS) strategy; parameter β controls the number of nodes at each level to further pursue

  2. **Exploration** of promising regions (Space(p^L)) of the state space
     - **Iteratively identify** a set of promising candidate root states during BS
     - allow some (non-) root nodes: effective *refinement* (find distant root ancestors to reach that node)

Workflow of the IMSBS



Heuristic guidances in BS

Three LCS heuristic guidances (from literature):

1. UB_1: "Look-ahead" for the remaining sequence length
2. UB_2: *Character Frequency Alignment* score
   - sum up the maximum possible occurrences of each letters in remaining subsequences
3. h_{prob}: *probability-based heuristic* guidance (matrices of probabilities (Mousavi and Tabataba, 2012))

Experimental Evaluation: general m-case

- BS: a baseline beam search, executes a single iteration of IMSBS (utilizing a huge β)
- IMSBS-GREEDY: fix beam-width $\beta = 1$ for the forward BS, perform a large number of IMSBS iterations
  - impact of iterations on the overall performance of IMSBS
- IMSBS: a tuned version; configured to use an average runtime comparable to that of BS

The IMSBS framework implemented in **Python** 3.11.

Benchmark set RANDOM

For each combination of instance parameters

- $n \in \{50, 100, 200, 500\}$
- $m \in \{2, 3, 5, 10\}$
- $|\Sigma| \in \{2, 4\}$

10 random problem instances are generated (sequences uniformly at random).

Gap constraints generated from $G_s(\cdot) \in \mathcal{U}(\{\lfloor 0.5 \cdot |\Sigma| \rfloor, \ldots, \lfloor 1.5 \cdot |\Sigma| \rfloor\})$.

$\Longrightarrow$ A total of **320 problem instances** is generated.

Parameter tuning of IMSBS

We fixed less sensible params (preliminary tests): beam width in backward BS, $|\mathcal{L}|$, h'

Tuned parameters:

- β (in BS-forward)
- Heuristic guidance in BS (forward)
  - $\{\text{UB}_1, \text{UB}_2, h_{prob}\}$

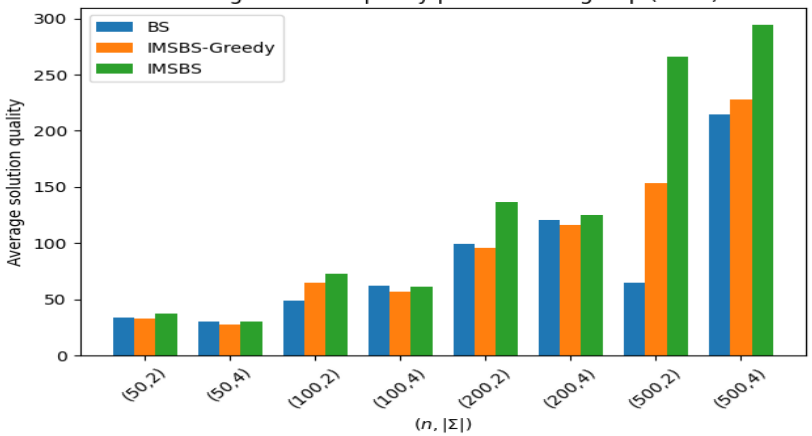
Grid search used for tuning; avg. solution quality over all (320) instances!

The results of tuning

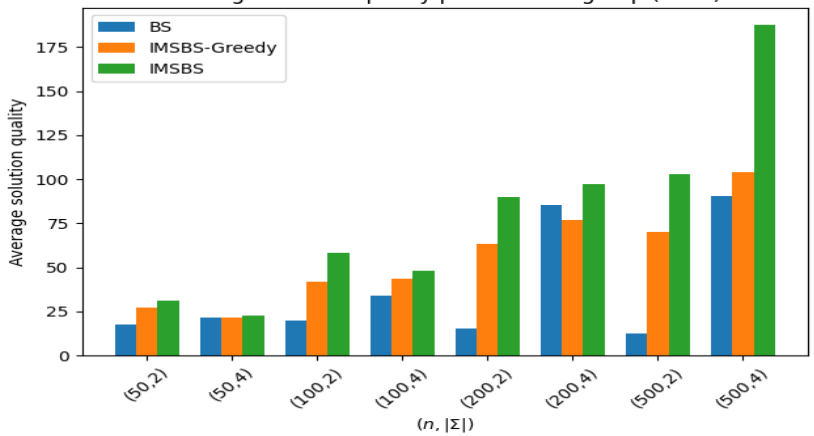
- **(Baseline)** $\textsc{Bs} \Longrightarrow \beta = 10,000$ and $h = h_{\text{prob}}$
- $\textsc{Imsbs} \Longrightarrow h = \text{UB}_2$, $\beta = 500$, and $\#iter\_imsbs = 100$
- $\textsc{Imsbs-Greedy}$: $\beta = 1$, $\#iter\_imsbs = 10,000$

Numerical results

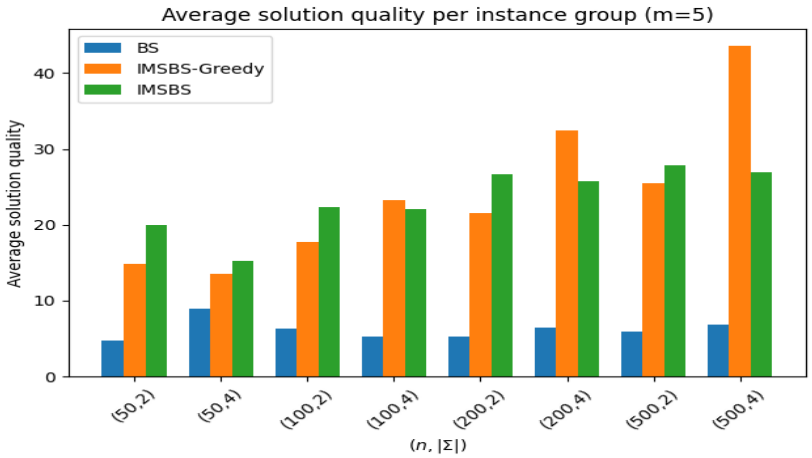
$m = 2$



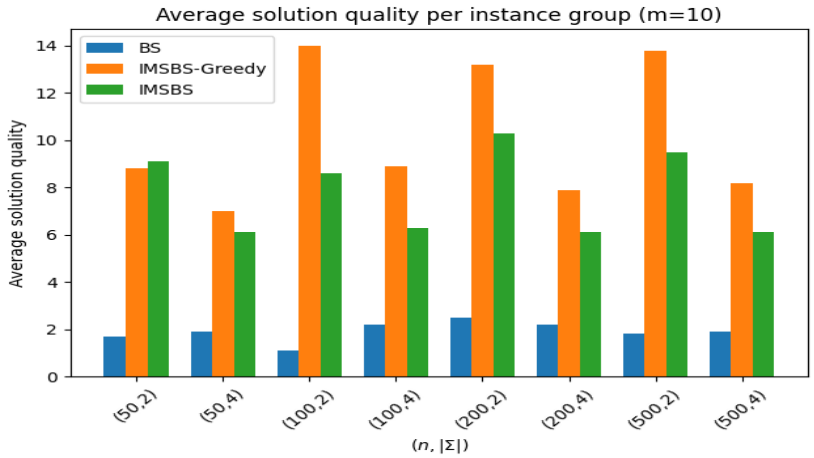
$m = 3$



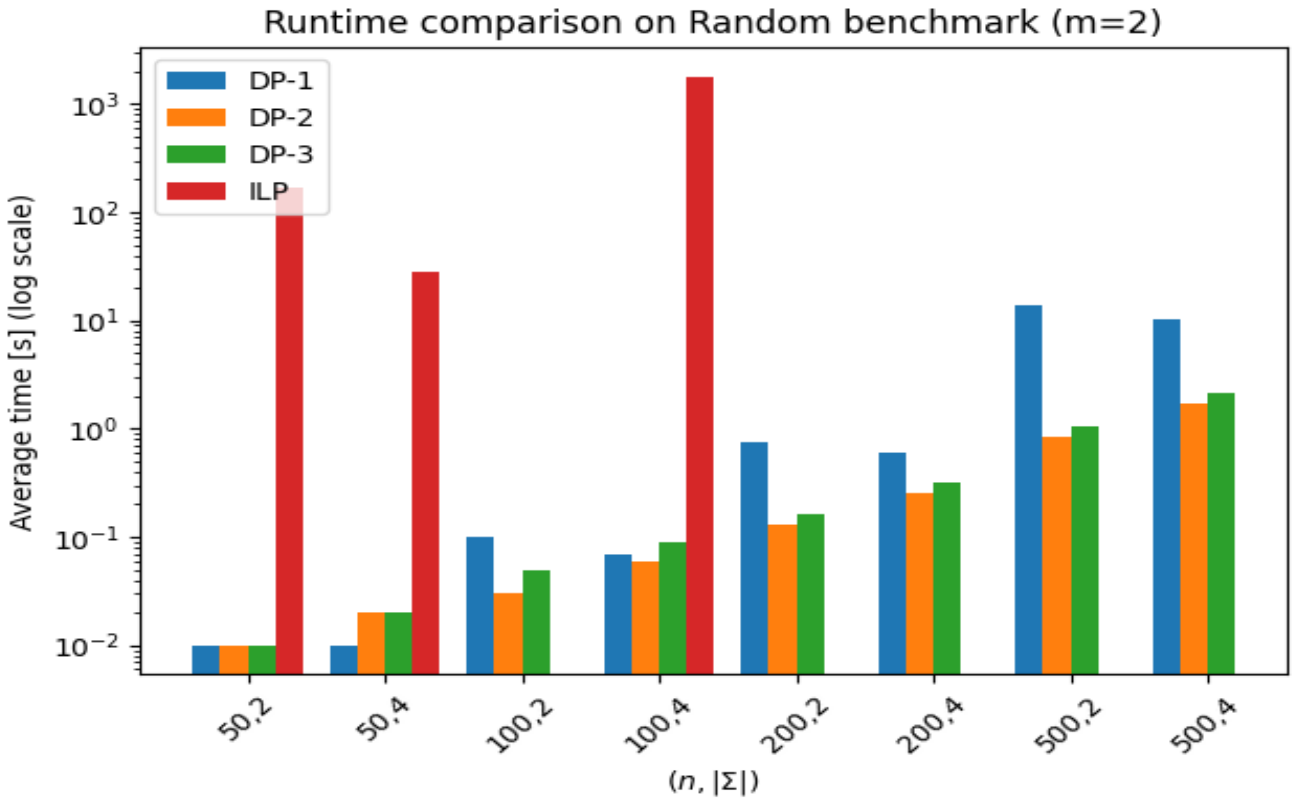
$m = 5$



$m = 10$

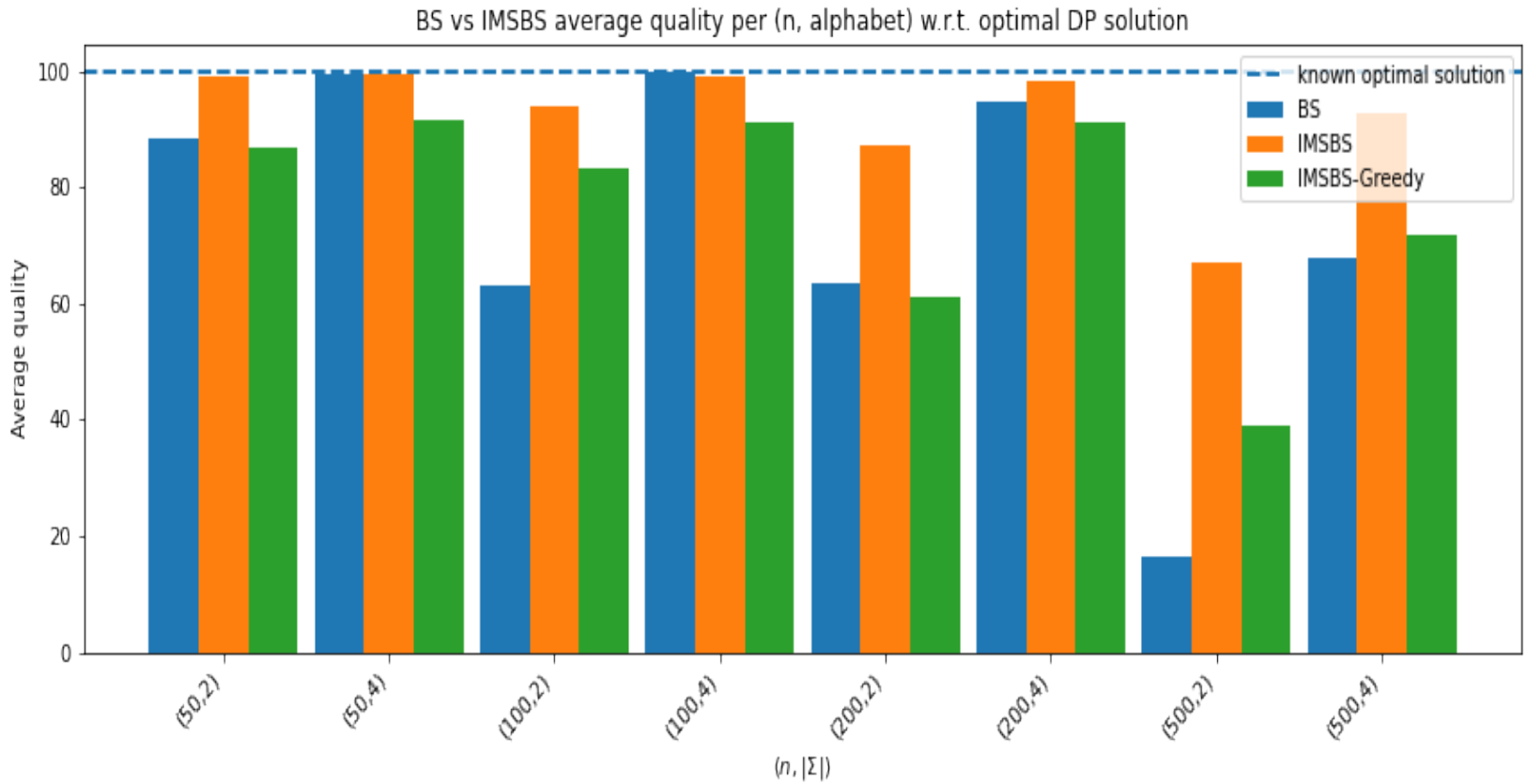


Numerical results for $m = 2$



- **The first known empirical study for the $m = 2$ (80 instances)**
- ILP: proposed in this work

The $m = 2$ case: heuristic performance vs. optimal solution (gaps in percent)



Conclusions and Future Work

- Proposed a **general heuristic framework** IMSBS to solve the multiple VGLCS problem
- Combines consecutive Beam search calls inducing source node generation and refinement of sources for further iterations
  - balance between intensification and diversification
- Empirical studies conducted for the first time on the **synthetic instances**: IMSBS outperforms the baseline Beam search

Future work:

- **Real-world** instance-case scenario
- Lack of more advanced heuristic guidance: **Data-driven/ML heuristic** involving various local and global features (NN-based)

Thank you for your attention!