Learning Beam Search

On the Utilization of Machine Learning Techniques for Guiding Beam Search to solve Combinatorial Optimization Problems

Marc Huber and Günther Raidl
{mhuber|raidl}@ac.tuwien.ac.at

Institute of Logic and Computation
TU Wien

May 18, 2021

Repetition: Idea of Learning Beam Search (LBS) and Related Work

Basic Idea:

- Meta-Algorithm that automatically learns a guidance-heuristic for Beam Search (BS) based on randomly generated instances

Related Work:

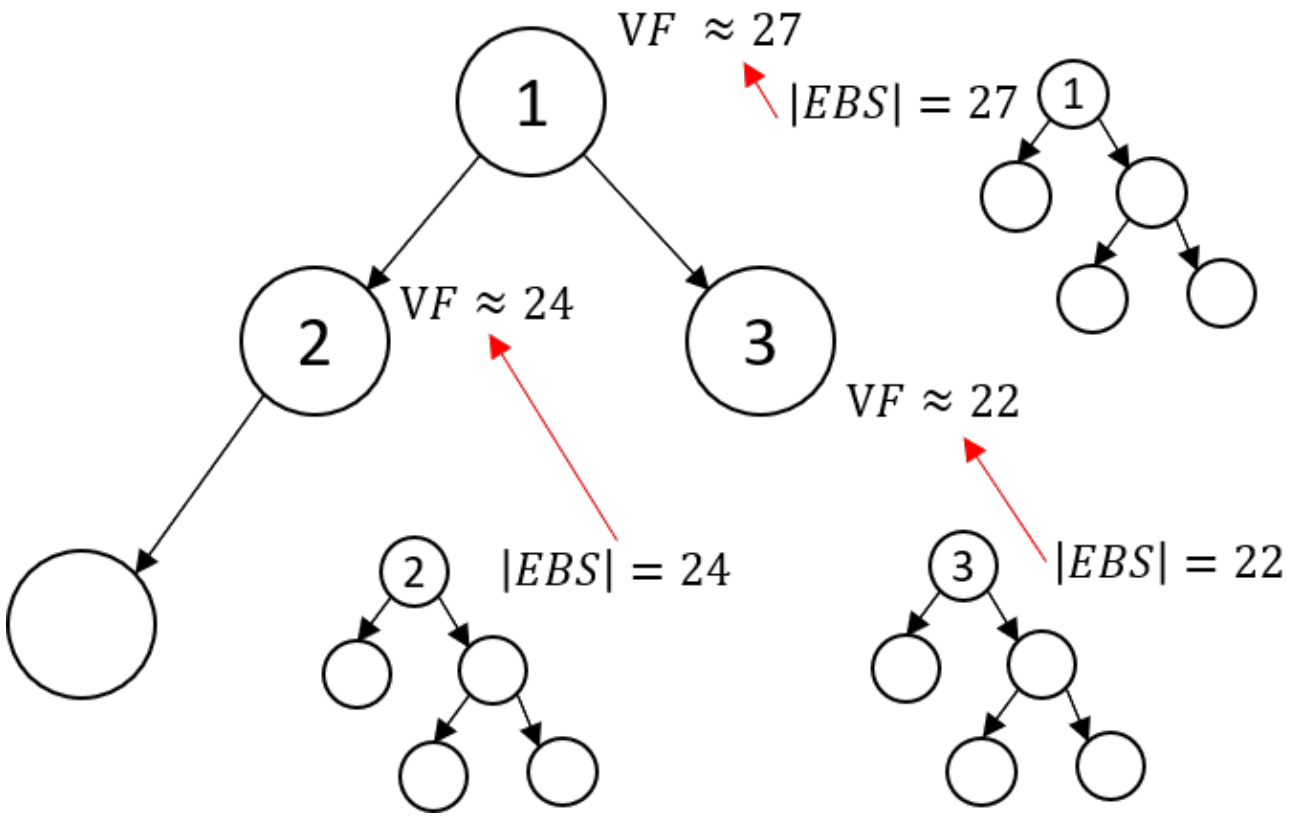
- Learning Beam Search Policies via Imitation Learning (2019)

  by R. Negrinho, M. R. Gormley and G. J. Gordon

  - Unifying Meta-Algorithm for Learning Beam Search policies using imitation learning
  - Meta-Algorithm captures existing learning algorithms and suggests new ones

Repetition: Basic Idea of Learning Beam Search (LBS)

- Perform a BS, guided by a learnable Value Function (VF)
- Obtain training data by performing independent so-called Embedded Beam Search (EBS)



Repetition: Procedure

LBS:

1. Initialize VF randomly
2. Create a random instance
3. Perform a BS, guided by the Value Function
4. Store observations and EBS values for each node of a LBS Tree with probability ϕ in a limited size FIFO
5. Train VF on FIFO Data
6. Repeat steps 2-5 until a stopping criterion is fulfilled

Probability ϕ

- Applying an EBS on every node would create too many training samples for a single instance
- Perform for each node an EBS and add samples to the FIFO only with a certain probability

$$\phi_t := \frac{t}{\gamma_{t-1}} \cdot \alpha, \tag{1}$$

where $t \geq 0$ denotes the current iteration, $\gamma_t > 0$ describes the sum of all created nodes up to iteration t, and $\alpha > 0$ is a strategy parameter.

Repetition: Hyperparameter

Table: Hyperparameter

Parameter	Description
iter	number of LBS iterations
min_obs_for_learn	minimum number of observations to start with learning
β	beam width of the main BS
β'	beam width of the EBS
α	factor for controlling number of generated samples per LBS iteration
replay_capacity	maximum size of the FIFO
batch_size	minimum batch size for learning
ML model	ML model used as VF

Considered Problems: LCS, CLCS and SCS

Given: set of m input strings $S = \{s_1, \ldots, s_m\}$ over alphabet Σ

- LCS: find a longest string that appears as subsequence in any string of S

  Example: $m = 2$, $|\Sigma| = 3$

  s_1: ABBA
  s_2: CABA
  $\Rightarrow$ ABA

- CLCS: extends LCS by a pattern string P that must appear as subsequence in a feasible solution

  Example: P=AB

  s_1: ABBA
  s_2: CABA
  $\Rightarrow$ ABA

Considered Problems: LCS, CLCS and SCS

Given: set of m input strings $S = \{s_1, \ldots, s_m\}$ over alphabet Σ

- SCS: find a shortest string that contains each $s \in S$ as a subsequence

  Example: $m = 2$, $|\Sigma| = 3$

  s_1: ABBA
  s_2: CABA
  $\Rightarrow$ CABBA

Considered Problems: LCS, CLCS and SCS

- Many applications in computational biology, text editing, etc.
  - e.g. to compare two DNA or protein sequences to learn how homologous they are
- Can be solved efficiently in time $\mathcal{O}(n^2)$ for $m = 2$ strings by dynamic programming (n: string length)
- NP-hard for general m
- Exact methods only suitable for small m or n
- State-of-the-art heuristic approaches for large m and n are based on Beam Search

LCS,CLCS and SCS State Space and Environment

- State space: directed acyclic graph $G = (V, A)$, in which
  - states (nodes) $v \in V$ are represented by position vectors $p^v = (p_i^v)_{i=1,\ldots,m}$ with $p_i^v \in 1, \ldots, |s_i| + 1$
  - the root node $r \in V$ has position vector $p^r = (1, \ldots, 1)$
  - Arcs in A represent actions (a valid non-dominated letter to a partial solution)
  - Terminal states are represented by the state $t \in V$ with $p^t = (|s_i| + 1)_{i=1,\ldots,m}$

LCS,CLCS and SCS State Space and Environment

Input for VF (=Observations):

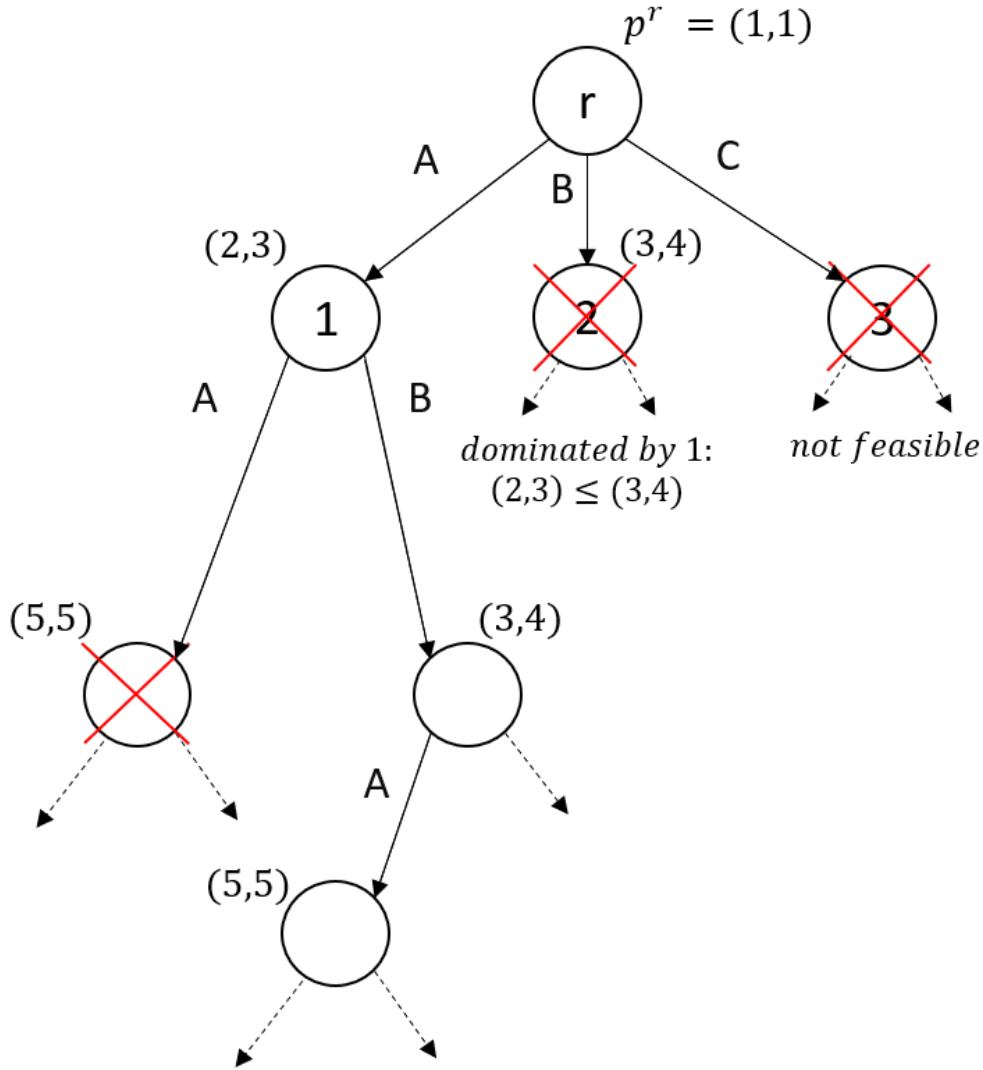
- Type 1: Remaining string lengths $|s_i| - p_i^v + 1$, $i = 1, \ldots, m$, sorted according to non-decreasing values
- LCS
  - Type 1 + minimum letter appearances $\min_{i=1,\ldots,m} |s_i[p_i^v, |s_i|]|_c$, $c \in \Sigma$
- CLCS
  - Type 1 extended by the remaining string length of the pattern string $|P| - p_{m+1}^v + 1$
  - Minimum letter appearances $\min_{i=1,\ldots,m} |s_i[p_i^v, |s_i|]|_c$, $c \in \Sigma$
- SCS
  - Type 1 + maximum letter appearances $\max_{i=1,\ldots,m} |s_i[p_i^v, |s_i|]|_c$, $c \in \Sigma$

LCS,CLCS and SCS State Space and Environment

Figure: LCS Example

LCS $s_1: ABBA$
$s_2: CABA$

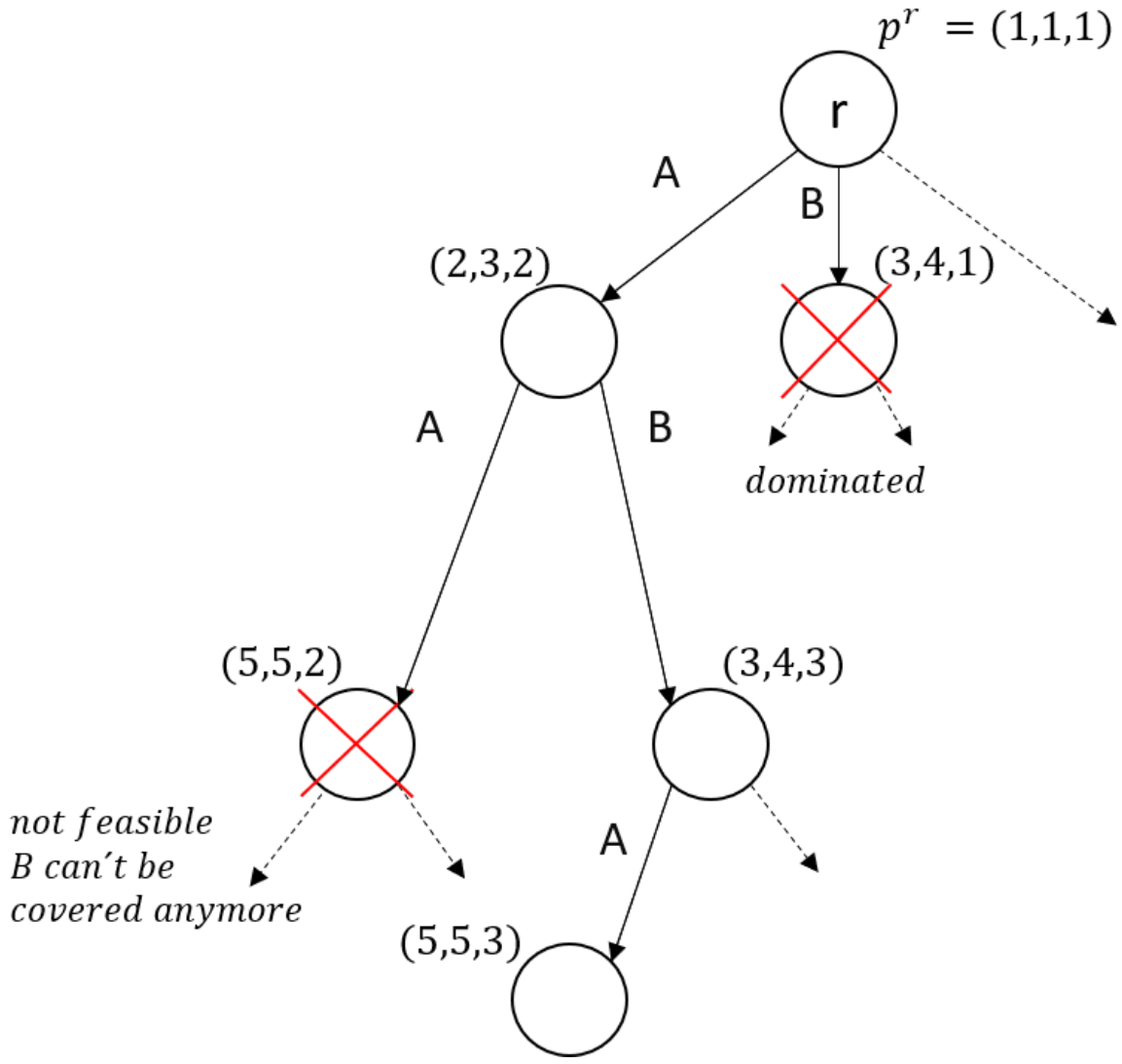
$solution: ABA$



LCS,CLCS and SCS State Space and Environment

Figure: CLCS Example

CLCS $p: AB$
$s_1: ABBA$
$s_2: CABA$

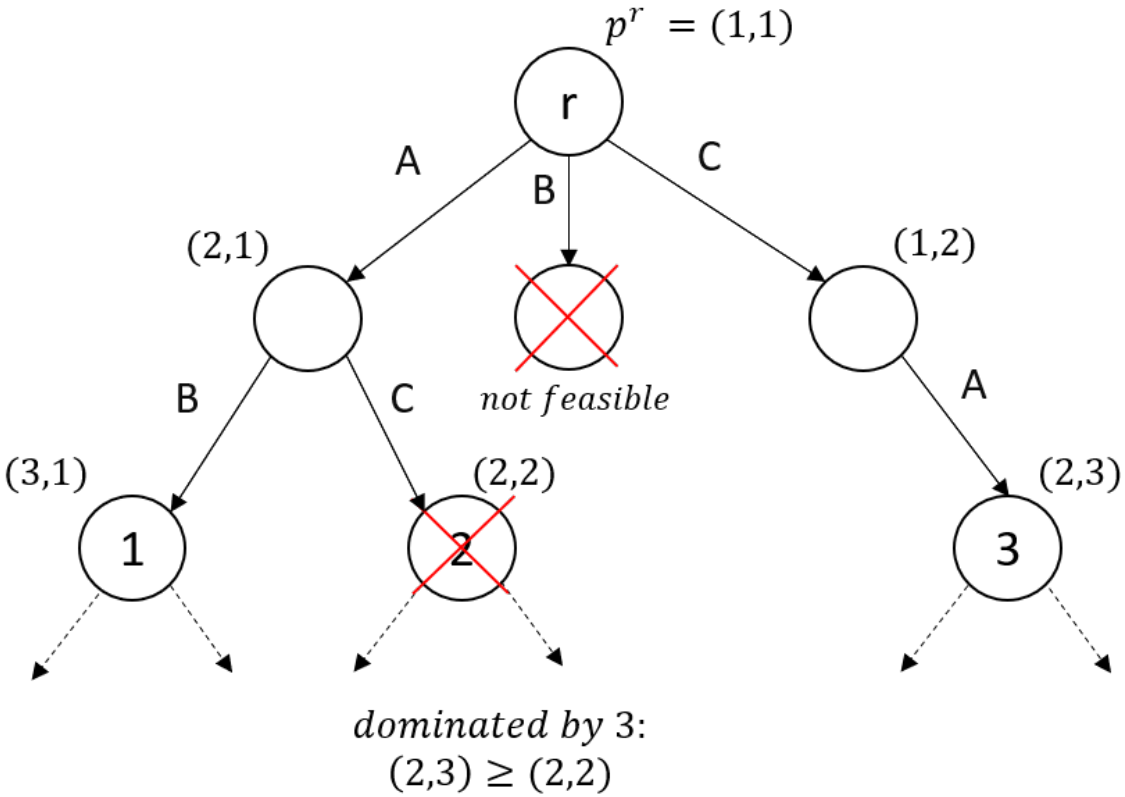


$solution: ABA$

LCS,CLCS and SCS State Space and Environment

Figure: SCS Example

SCS $s_1: ABBA$
$s_2: CABA$

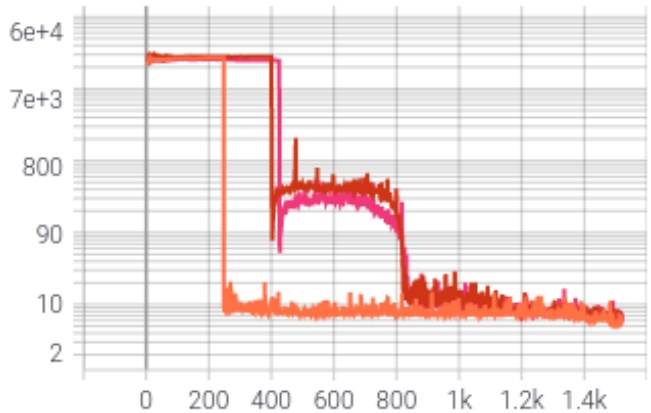


$solution: CABBA$

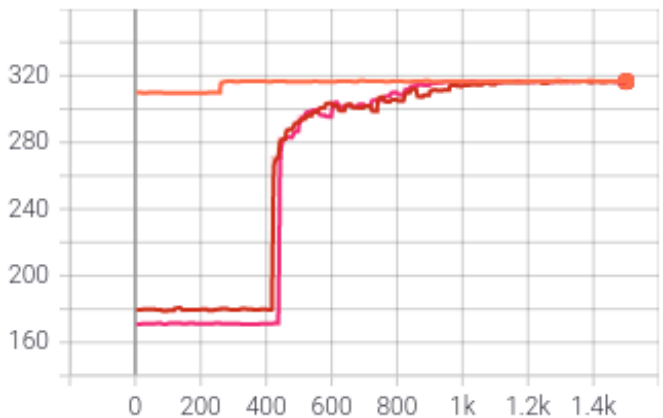
LCS: Evaluation on Random Testinstances

Instance: $n = 600$, $m = 3$, $|\Sigma| = 4$

Loss of VF on FIFO (Training Data) over LBS itertions



Independent interleaved test: average solution length of BS with VF over LBS iterations



LCS: Benchmark Instances

Six different benchmark sets

- BL instance set from Blum and Festa
- Random, Rat and Virus instance set from Shyu and Tsai
- ES instance set from Easton and Singireddy
- BB instance set from Blum and Blesa

Characteristics

- $n \in \{100, 600, 1000, 2500, 500\}$
- $m \in \{10, 15, 20, 25, 40, 50, 60, 80, 100, 150, 200\}$
- $|\Sigma| \in \{2, 4, 8, 10, 12, 20, 24, 25, 100\}$
- Total: 1910 instances

Test setting:

- Intel Xeon E5-2640 v4 2.40GHz, all runs single-threaded, 32GB

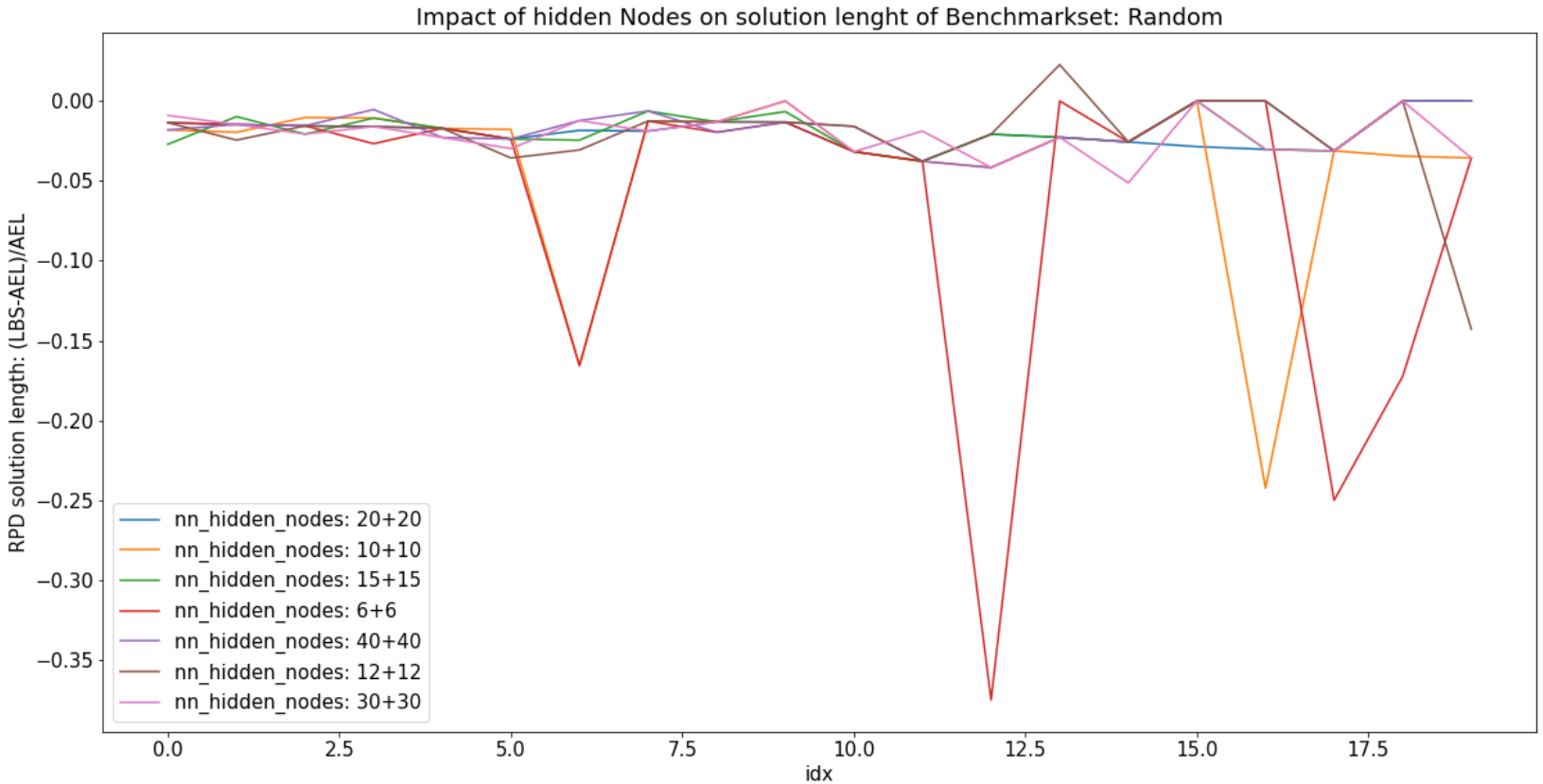
LCS Evaluation: Hyperparameter

Table: Hyperparameter

Parameter	Value
iter	1000
min_obs_for_learn	3000
$\beta = \beta'$	low-time: 50 (training+testing), high-time: 600 (testing only)
α	60
replay_capacity	5000
batch_size	32
ML model	NN with $20 + 20$ hidden nodes, ReLU activation functions except at the output layer

LCS Evaluation: Hyperparameter

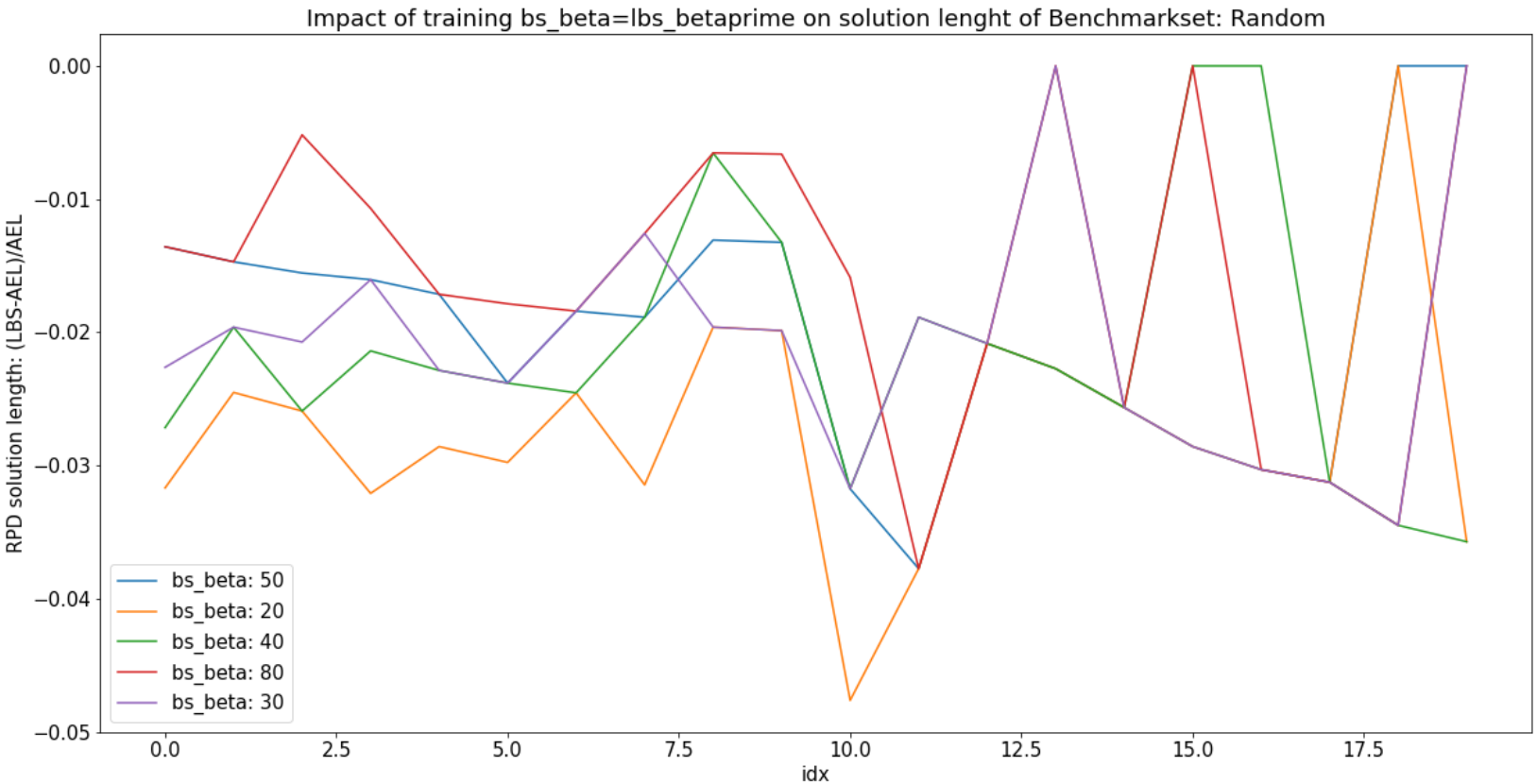
Figure: Impact of hidden nodes on solution length on set random



- NN with $<$ 15+15 hidden nodes leads to poor results
- better 20+20 or more hidden nodes for robustness and stability

LCS Evaluation: Hyperparameter

Figure: Impact of beta=beta' on solution length on set random



- $\beta < 50$ leads to poor results
- better set $\beta \geq 50$ for good results

LCS Results

Table: Solution length: LBS vs Literature

Table: Low-computation ($\beta = 50$)

set	#better	#equal	#worse	#total
BB	5	1	2	8
BL	1	1	13	15
ES	1	0	9	10
random	0	6	14	20
rat	6	9	5	20
virus	5	4	11	20

Table: Hi-computation ($\beta = 600$)

set	#better	#equal	#worse	#total
BB	3	4	1	8
BL	0	2	13	15
ES	0	0	0	0
random	0	11	9	20
rat	4	11	5	20
virus	3	10	7	20

CLCS: Benchmark Instances

Datasets generated by Marko, each one comprising 10 different instances

Characteristics

- $n \in \{10, 500, 1000\}$
- $\frac{n}{P} \in \{2, 4, 10, 20, 50\}$
- $m \in \{10, 50, 100\}$
- $|\Sigma| \in \{4, 20\}$

Table: Solution length: LBS ($\beta = 2000$) vs Literature

n_div_P	#better	#equal	#worse	#total
2	0	18	0	18
4	3	13	2	18
10	7	11	0	18
20	5	8	5	18
50	4	6	8	18

SCS: Benchmark Instances

- 6 DNA datasets and 3 protein datasets
  - each one comprising 10 different instances
  - $m \in \{100, 500\}$, $n \in \{100, 500, 1000\}$, $|\Sigma| \in \{4, 20\}$
- RandomSet
  - each instance is composed of eight strings ($m = 8$), four of them with 40 symbols, and the remaining four with 80 symbols, $|\Sigma| = \{2, 4, 8, 16, 24\}$

SCS: Benchmark Instances

Table: Benchmark set: DNA, Protein and Random

set	sigma	n	m	s_LBS100	t_LBS100	s_MA_BS	s_PBS	s_DR	s_IBS_SCS	t_IBS_SCS	s_MPBS	t_MPBS
DNA	4	100	100	273.200000	0.708000	nan	nan	284.400000	272.300000	1	267.420000	0.380000
			500	288.000000	2.842000	nan	nan	296.800000	288.100000	4	284.380000	2.460000
		500	100	1290.000000	3.595000	nan	nan	1364.400000	1284.600000	6	1273.700000	3.450000
			500	1364.500000	22.140000	nan	nan	1420.700000	1351.600000	22	1340.790000	11.940000
		1000	100	2589.800000	15.885000	nan	nan	2675.700000	2540.100000	16	2526.410000	12.550000
			500	nan	nan	nan	nan	2769.100000	2662.900000	48	2644.630000	22.660000
Protein	20	100	100	nan	nan	nan	nan	1008.700000	910.600000	16	874.220000	6.640000
			500	nan	nan	nan	nan	1199.800000	1118.100000	74	1084.900000	27.400000
		500	100	nan	nan	nan	nan	4846.300000	4374.900000	159	4280.380000	118.540000
random	2	80	8	111.400000	0.248000	110.700000	110.900000	nan	109.400000	1	108.800000	0.180000
	4	80	8	145.200000	0.539000	146.400000	145.400000	nan	142.400000	2	139.650000	0.430000
	8	80	8	184.800000	1.151000	192.600000	187.200000	nan	180.600000	3	176.670000	1.800000
	16	80	8	237.400000	1.848000	244.000000	241.900000	nan	235.600000	6	225.960000	7.090000
	24	80	8	270.000000	1.114000	281.200000	277.900000	nan	268.800000	8	254.330000	5.640000

Conclusions

- Loss of VF and average lengths converge well, self-learning works
- Better choose Hyperparameter not too small
- LBS
  - outperforms approaches in the literature on CLCS Problem
  - is competitive with approaches in the literature on LCS Problem
  - performs poor on SCS Problem

Future Work

- The actual values are not important, but just the resulting ranking of the nodes. Can we exploit this freedom by some better suited target values for the learning?
- Learn a function that receives as input two observations from two states and outputs a probability value indicating which of the two states is more promising to lead to an optimal solution
- Exploit distribution of instance for training

[1] Renato Negrinho and Matthew R. Gormley and Geoffrey J. Gordon, "Learning Beam Search Policies via Imitation Learning", 2019

[2] L. Bergroth, H. Hakonen and T. Raita, "A survey of longest common subsequence algorithms," Proceedings Seventh International Symposium on String Processing and Information Retrieval. SPIRE 2000, A Curuna, Spain, 2000, pp. 39-48, doi: 10.1109/SPIRE.2000.878178.

[3] M. Djukanovic, G. R. Raidl, and C. Blum. A beam search for the longest common subsequence problem guided by a novel approximate expected length calculation. In G. Nicosia, P. Pardalos, G. Giuffrida, R. Umeton, and V. Sciacca, editors, Proceedings of LOD 2019 – The 5th International Conference on Machine Learning, Optimization and Data Science, volume 11943 of LNCS, pages 154–167. Springer, 2020.