A Policy-Based Learning Beam Search for Combinatorial Optimization

Marc Huber
Institute of Logic and Computation, TU Wien, Vienna, Austria
mhuber@ac.tuwien.ac.at

joint work with Rupert Ettrich and Günther Raidl

Second Vienna Workshop on Computational Optimization
March 16, 2023

1. Introduction

- Historically, components of heuristic algorithms to solve combinatorial problems are manually designed by a human expert.
  - Suboptimal.
  - Expensive.
- Learning to search: replace hard-coded heuristic components with machine learning models that assist in lower-level decisions.

Goal: Utilize machine learning to learn policy functions that guide beam search efficiently toward more promising solutions.

1. Introduction

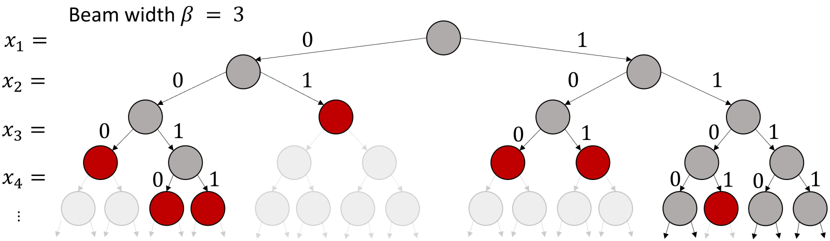
Beam Search (BS): Incomplete tree search algorithm

- Determines at each level the β most promising nodes to pursue further via evaluation function

$$f(v) = g(v) + h(v),$$

where

- $g(v)$: cost from root node r to node v.
- $h(v)$: heuristic estimated cost from node v to goal node t.



2. Related Work

Learning Beam Search Policies via Imitation Learning:

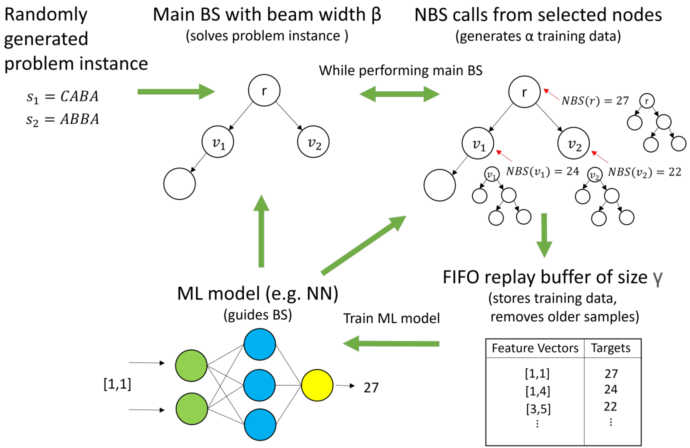
[Negrinho et al., 2018]

- Presented a meta algorithm that learns BS policies for structured prediction problems by imitation learning.
  - Learns a scoring function for BS to match the ranking induced by given oracle costs.
  - Proposed and analyzed several loss functions and data collection strategies that consider the beam also at train time.
- Pure theoretical work.

2. Related Work

Learning Beam Search (LBS) [Huber and Raidl, 2021]:

- Multilayer perceptron (MLP) used as heuristic $h(v)$.
- MLP is trained offline in a reinforcement learning manner on many representative randomly generated problem instances.



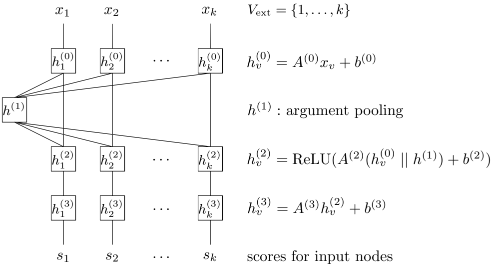
3. Policy-Based Learning Beam Search (P-LBS)

- **Builds upon our earlier LBS framework.**
- **Policy function = MLP:** applied to all the expanded nodes at a current BS level together.
  - ⇒ Relies not on the prediction of actual cost-to-go values!
- **Four-layer MLP architecture for P-LBS:**

Notation: V_{ext} = set of all nodes encountered at one BS level.

x_v = feature vector of node $v \in V_{\text{ext}}$.

$A^{(i)}$ = shared weight matrix; $b^{(i)}$ bias vector.



3. Policy-Based Learning Beam Search (P-LBS)

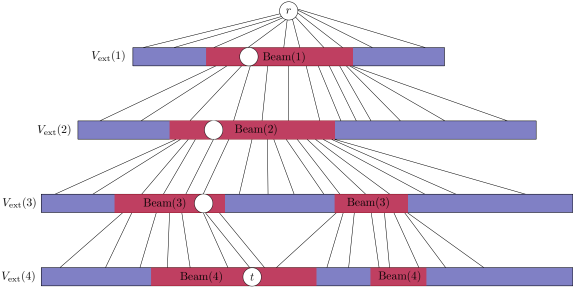
Abstract training procedure:

1. Initialize MLP randomly.
2. Create random problem instance.
3. Perform BS guided by the (retrained) MLP.
4. Store all nodes encountered on $\alpha \in \mathbb{N}$ randomly selected levels during BS in a FIFO replay buffer.
5. Train MLP on FIFO replay buffer data.
6. Repeat steps 2-5 until a stopping criterion is fulfilled.

3. Policy-Based Learning Beam Search (P-LBS)

Two different approaches to label training data:

1. *beam-unaware*: label nodes that lie on the $r - t$ path obtained by BS with ones and all other nodes with zero.



Exemplary training data labeling using *beam-unaware*.

2. *beam-aware*: perform NBS on each node $v \in V_{\text{ext}}$ to obtain estimated values for the oracle cost.

3. Policy-Based Learning Beam Search (P-LBS)

Adam optimizer is used to update network weights with respect to different loss functions.

<u>Notation:</u>

$c = (c_v)_{v \in V_{\mathrm{ext}}}$	$=$	vector of all target values of the nodes in V_{ext}.
$s = (s_v)_{v \in V_{\mathrm{ext}}}$	$=$	vector of all scores obtained by evaluating the MLP for V_{ext}.
$\hat{\sigma}$	$=$	permutation of V_{ext} that sorts scores in s such that $s_{\hat{\sigma}(1)} \geq s_{\hat{\sigma}(2)} \geq \ldots \geq s_{\hat{\sigma}(\lvert V_{\mathrm{ext}} \rvert)}$.
σ^*	$=$	permutation of V_{ext} that sorts target values in c such that $c_{\sigma^*(1)} \geq c_{\sigma^*(2)} \geq \ldots \geq c_{\sigma^*(\lvert V_{\mathrm{ext}} \rvert)}$.

3. Policy-Based Learning Beam Search (P-LBS)

Example:

$$V_{\text{ext}} = \{v_1, v_2, v_3, v_4, v_5\}.$$

$$\beta = 2.$$

$$\mathrm{MLP}(x_{v_1}, x_{v_2}, \ldots, x_{v_5}) = (s_{v_1}, s_{v_2}, s_{v_3}, s_{v_4}, s_{v_5}).$$

$$(\mathrm{NBS}(v_i))_{i=1,\ldots,5} = (c_{v_1}, c_{v_2}, c_{v_3}, c_{v_4}, c_{v_5}).$$

$$s_{\hat{\sigma}(1)} \geq s_{\hat{\sigma}(2)} \geq \ldots \geq s_{\hat{\sigma}(5)} = s_{v_5} \geq s_{v_2} \geq s_{v_1} \geq s_{v_4} \geq s_{v_3}.$$

$$c_{\sigma^*(1)} \geq c_{\sigma^*(2)} \geq \ldots \geq c_{\sigma^*(5)} = c_{v_2} \geq c_{v_1} \geq c_{v_5} \geq c_{v_4} \geq c_{v_3}.$$

Loss functions proposed by [Negrinho et al., 2018]:

- perceptron first (pf):

$$\ell(s, c) = \max(0, s_{\hat{\sigma}(1)} - s_{\sigma^*(1)}).$$

3. Policy-Based Learning Beam Search (P-LBS)

Example:

$$V_{\text{ext}} = \{v_1, v_2, v_3, v_4, v_5\}.$$

$$\beta = 2.$$

$$\text{MLP}(x_{v_1}, x_{v_2}, \ldots, x_{v_5}) = (s_{v_1}, s_{v_2}, s_{v_3}, s_{v_4}, s_{v_5}).$$

$$(\text{NBS}(v_i))_{i=1,\ldots,5} = (c_{v_1}, c_{v_2}, c_{v_3}, c_{v_4}, c_{v_5}).$$

$$s_{\hat{\sigma}(1)} \geq s_{\hat{\sigma}(2)} \geq \ldots \geq s_{\hat{\sigma}(5)} = s_{v_5} \geq s_{v_2} \geq s_{v_1} \geq s_{v_4} \geq s_{v_3}.$$

$$c_{\sigma^*(1)} \geq c_{\sigma^*(2)} \geq \ldots \geq c_{\sigma^*(5)} = c_{v_2} \geq c_{v_1} \geq c_{v_5} \geq c_{v_4} \geq c_{v_3}.$$

Loss functions proposed by [Negrinho et al., 2018] cont'd:

- cost-sensitive margin last (cml):

$$\ell(s, c) = (c_{\sigma^*(1)} - c_{\hat{\sigma}(\beta)}) \max(0, 1 + s_{\hat{\sigma}(\beta)} - s_{\sigma^*(1)}).$$

3. Policy-Based Learning Beam Search (P-LBS)

Example:

$$V_{\text{ext}} = \{v_1, v_2, v_3, v_4, v_5\}.$$

$$\beta = 2.$$

$$\text{MLP}(x_{v_1}, x_{v_2}, \ldots, x_{v_5}) = (s_{v_1}, s_{v_2}, s_{v_3}, s_{v_4}, s_{v_5}).$$

$$(\text{NBS}(v_i))_{i=1,\ldots,5} = (c_{v_1}, c_{v_2}, c_{v_3}, c_{v_4}, c_{v_5}).$$

$$s_{\hat{\sigma}(1)} \geq s_{\hat{\sigma}(2)} \geq \ldots \geq s_{\hat{\sigma}(5)} = s_{v_5} \geq s_{v_2} \geq s_{v_1} \geq s_{v_4} \geq s_{v_3}.$$

$$c_{\sigma^*(1)} \geq c_{\sigma^*(2)} \geq \ldots \geq c_{\sigma^*(5)} = c_{v_2} \geq c_{v_1} \geq c_{v_5} \geq c_{v_4} \geq c_{v_3}.$$

Loss functions proposed by [Negrinho et al., 2018] cont'd:

- log loss neighbors (lln):

$$\ell(s, c) = -s_{\sigma^*(1)} + \log\left(\sum_{i=1}^{|V_{\text{ext}}|} \exp(s_i)\right).$$

3. Policy-Based Learning Beam Search (P-LBS)

Example:

$$
\begin{aligned}
V_{\text{ext}} &= \{v_1, v_2, v_3, v_4, v_5\}. \\
\beta &= 2. \\
\mathrm{MLP}(x_{v_1}, x_{v_2}, \ldots, x_{v_5}) &= (s_{v_1}, s_{v_2}, s_{v_3}, s_{v_4}, s_{v_5}). \\
(\mathrm{NBS}(v_i))_{i=1,\ldots,5} &= (c_{v_1}, c_{v_2}, c_{v_3}, c_{v_4}, c_{v_5}). \\
s_{\hat{\sigma}(1)} \geq s_{\hat{\sigma}(2)} \geq \ldots \geq s_{\hat{\sigma}(5)} &= s_{v_5} \geq s_{v_2} \geq s_{v_1} \geq s_{v_4} \geq s_{v_3}. \\
c_{\sigma^*(1)} \geq c_{\sigma^*(2)} \geq \ldots \geq c_{\sigma^*(5)} &= c_{v_2} \geq c_{v_1} \geq c_{v_5} \geq c_{v_4} \geq c_{v_3}.
\end{aligned}
$$

Loss functions proposed by [Negrinho et al., 2018] cont'd:

- upper bound (ub):

$$\ell(s, c) = \max(0, \delta_{\beta+1}, \ldots, \delta_{|V_{\text{ext}}|}),$$

where $\delta_j = (c_{\sigma^*(1)} - c_{\sigma^*(j)})(s_{\sigma^*(j)} - s_{\sigma^*(1)})$ for $j = \beta + 1, \ldots, |V_{\text{ext}}|$.

3. Policy-Based Learning Beam Search (P-LBS)

Example:

$$
\begin{aligned}
V_{\text{ext}} &= \{v_1, v_2, v_3, v_4, v_5\}. \\
\beta &= 2. \\
\mathrm{MLP}(x_{v_1}, x_{v_2}, \dots, x_{v_5}) &= (s_{v_1}, s_{v_2}, s_{v_3}, s_{v_4}, s_{v_5}). \\
(\mathrm{NBS}(v_i))_{i=1,\dots,5} &= (c_{v_1}, c_{v_2}, c_{v_3}, c_{v_4}, c_{v_5}). \\
s_{\hat{\sigma}(1)} \geq s_{\hat{\sigma}(2)} \geq \dots \geq s_{\hat{\sigma}(5)} &= s_{v_5} \geq s_{v_2} \geq s_{v_1} \geq s_{v_4} \geq s_{v_3}. \\
c_{\sigma^*(1)} \geq c_{\sigma^*(2)} \geq \dots \geq c_{\sigma^*(5)} &= c_{v_2} \geq c_{v_1} \geq c_{v_5} \geq c_{v_4} \geq c_{v_3}.
\end{aligned}
$$

Loss functions proposed by us:

- cost-sensitive margin beam (cmb):

$$\ell(s, c) = \sum_{i=1}^{\beta-1} \max(0, c_{\sigma^*(i)} - c_{\hat{\sigma}(\beta)}) \max(0, 1 + s_{\hat{\sigma}(\beta)} - s_{\sigma^*(i)}).$$

3. Policy-Based Learning Beam Search (P-LBS)

Further loss functions proposed by [Negrinho et al., 2018]:

- perceptron last (pl):

$$\ell(s, c) = \max(0, s_{\hat{\sigma}(\beta)} - s_{\sigma^*(1)}).$$

- margin last (ml):

$$\ell(s, c) = \max(0, 1 + s_{\hat{\sigma}(\beta)} - s_{\sigma^*(1)}).$$

- log loss beam (llb):

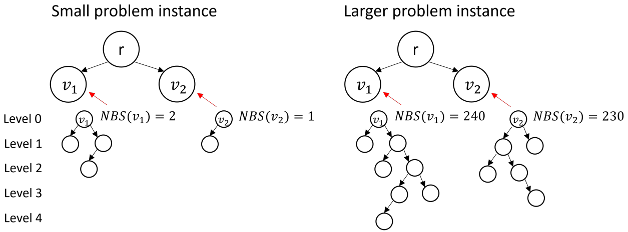
$$\ell(s, c) = -s_{\sigma^*(1)} + \log\left(\sum_{i \in I} \exp(s_i)\right),$$

where $I = \{\sigma^*(1), \hat{\sigma}(1), \ldots, \hat{\sigma}(\beta)\}$.

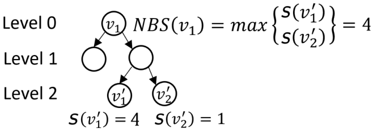
3. Policy-Based Learning Beam Search (P-LBS)

Bootstrapping for beam-aware data labeling:

- NBS calls are time expensive.



$\Rightarrow$ Stop NBS execution at level depth $d \in \mathbb{N}$ and use the so far trained MLP to obtain suitable training targets.



4. Experimental Evaluation

Longest Common Subsequence (LCS) problem:

- **Input:** set of m input strings $\mathcal{S} = \{S_1, \ldots, S_m\}$ over alphabet Σ, each of length $n = |S_i|_{i=1,\ldots,m}$.
- **Output:** longest string that appears as subsequence in any string of $\mathcal{S}$.
- **Example:** LCS of strings `AGACT`, `GTAAC`, and `GTACT` is `GAC`.

LCS benchmark instances:

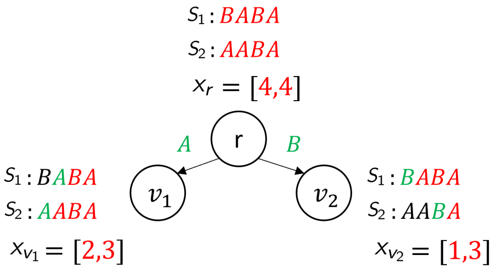
- **`rat`** instance set [Shyu and Tsai, 2009].
  - 20 single instances composed of sequences from rat genomes.
- **`ES`** instance set [Easton and Singireddy, 2008].
  - Nine instance sets. Each set contains 50 random instances.

State-of-the-art:

- [Djukanovic et al., 2019], and [Huber and Raidl, 2021].

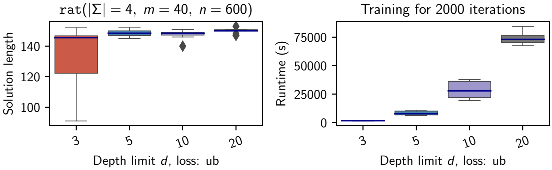
LCS problem: feature vectors for MLP

- Remaining string lengths $(x_v^i)_{i=1,\dots,m}$ *ordered ascending*, where $v \in V_{\mathrm{ext}}$.



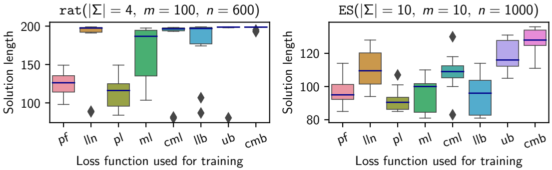
- $s := \mathrm{MLP}(x_{v_1}, x_{v_2}) = (3, 2)$, where $v_1, v_2 \in V_{\mathrm{ext}}$.

Bootstrapping:



Impact of depth limit d in NBS calls on the solution length of BS on a `rat` instance.

Loss functions:



Impact of the loss function in P-LBS on the solution lengths of BS on `rat` and `ES` instances.

5. Results on LCS benchmark instances

- Trained MLPs for each combination of $|\Sigma|$, m, and n occuring in benchmark instances on random instances using P-LBS with each loss function.
- Evaluated BS with trained MLPs on all instances from benchmark sets `rat` and `ES`.

⇒ BS with the trained MLPs with loss functions lln, cml, ub and cmb could achieve
  - in five out of 29 instance groups for $\beta = 50$,
  - and in two out of 29 for $\beta = 600$ new best solutions.

6. Conclusions and Future Work

- Presented a general P-LBS framework for learning BS policies to solve combinatorial optimization problems.
- Compared and evaluated different loss functions in the practical scenario of solving the LCS problem.
- Utilized bootstrapping to achieve reasonable scalability to larger problem instances.

Future Work:

- Weakness: disregarded in beam-unaware training the fact that multiple best goal nodes may exist.
  - ⇒ Adapt P-LBS so that all found equally good goal nodes and corresponding $r - t$ paths are considered.
- Utilize graph neural network as policy to get rid of the dependency of specific instance sizes.

Thank you for your attention!

Questions?

References I

Djukanovic, M., Raidl, G. R., and Blum, C. (2019).
A beam search for the longest common subsequence problem guided by a novel approximate expected length calculation.
In *Machine Learning, Optimization, and Data Science: 5th International Conference, LOD 2019, Siena, Italy, September 10–13, 2019, Proceedings*, page 154–167. Springer-Verlag.

Easton, T. and Singireddy, A. (2008).
A large neighborhood search heuristic for the longest common subsequence problem.
Journal of Heuristics, 14:271–283.

References II

Huber, M. and Raidl, G. R. (2021).
Learning beam search: Utilizing machine learning to guide beam search for solving combinatorial optimization problems.
In *Machine Learning, Optimization, and Data Science: 7th International Conference, LOD 2021, Grasmere, UK, October 4–8, 2021, Revised Selected Papers, Part II*, page 283–298. Springer-Verlag.

Negrinho, R., Gormley, M., and Gordon, G. J. (2018).
Learning beam search policies via imitation learning.
In Bengio, S. et al., editors, *Advances in Neural Information Processing Systems*, volume 31, pages 10652–10661. Curran Associates, Inc.

References III

Shyu, S. J. and Tsai, C.-Y. (2009).
Finding the longest common subsequence for multiple biological sequences by ant colony optimization.
Computers & Operations Research, 36(1):73–91.