Wouter Kool,
Herke van Hoof
& Max Welling

Buy 4 REINFORCE Samples, Get a Baseline for Free!

Get the paper!

University of Amsterdam

ORTEC OPTIMIZE YOUR WORLD

AMLAB Amsterdam Machine Learning Lab

REINFORCE with replacement

- Multiple samples *for a single datapoint* (e.g. instance, source sentence)
- Other samples can be used as baseline (unbiased)

$$\nabla_{\boldsymbol{\theta}} \mathbb{E}_{y \sim p_{\boldsymbol{\theta}}(y)}[f(y)] \approx \frac{1}{k} \sum_{i=1}^{k} \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(y_i) \left(f(y_i) - \frac{1}{k-1} \sum_{j \neq i} f(y_j) \right)$$
$$= \frac{1}{k-1} \sum_{i=1}^{k} \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(y_i) \left(f(y_i) - \frac{1}{k} \sum_{j=1}^{k} f(y_j) \right)$$

REINFORCE without replacement

- Samples without replacement are *not independent*!
- Include importance weights, dependent on sampling threshold κ (unbiased)

$$\nabla_{\boldsymbol{\theta}} \mathbb{E}_{y \sim p_{\boldsymbol{\theta}}(y)}[f(y)] \approx \sum_{i \in S} \frac{p_{\boldsymbol{\theta}}(y^i)}{q_{\boldsymbol{\theta},\kappa}(y^i)} \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(y^i) f(y^i) = \sum_{i \in S} \frac{\nabla_{\boldsymbol{\theta}} p_{\boldsymbol{\theta}}(y^i)}{q_{\boldsymbol{\theta},\kappa}(y^i)} f(y^i)$$

- Include a 'baseline' $B(S) = \sum_{j \in S} \frac{p_{\boldsymbol{\theta}}(y^j)}{q_{\boldsymbol{\theta},\kappa}(y^j)} f(y^j)$ (unbiased)

$$\nabla_{\boldsymbol{\theta}} \mathbb{E}_{y \sim p_{\boldsymbol{\theta}}(y)}[f(y)] \approx \sum_{i \in S} \frac{\nabla_{\boldsymbol{\theta}} p_{\boldsymbol{\theta}}(y^i)}{q_{\boldsymbol{\theta},\kappa}(y^i)} \left(f(y^i) \left(1 - p_{\boldsymbol{\theta}}(y^i) + \frac{p_{\boldsymbol{\theta}}(y^i)}{q_{\boldsymbol{\theta},\kappa}(y^i)} \right) - B(S) \right)$$

- Normalized importance weights (biased, but low variance)

$$\nabla_{\boldsymbol{\theta}} \mathbb{E}_{y \sim p_{\boldsymbol{\theta}}(y)}[f(y)] \approx \sum_{i \in S} \frac{1}{W_i(S)} \cdot \frac{\nabla_{\boldsymbol{\theta}} p_{\boldsymbol{\theta}}(y^i)}{q_{\boldsymbol{\theta},\kappa}(y^i)} \left(f(y^i) - \frac{B(S)}{W(S)} \right)$$

$$W(S) = \sum_{i \in S} \frac{p_{\boldsymbol{\theta}}(y^i)}{q_{\boldsymbol{\theta},\kappa}(y^i)}$$

$$W_i(S) = W(S) - \frac{p_{\boldsymbol{\theta}}(y^i)}{q_{\boldsymbol{\theta},\kappa}(y^i)} + p_{\boldsymbol{\theta}}(y^i)$$

Stochastic Beam Search (Kool et al., 2019b)

Stochastic Beams and Where to Find Them

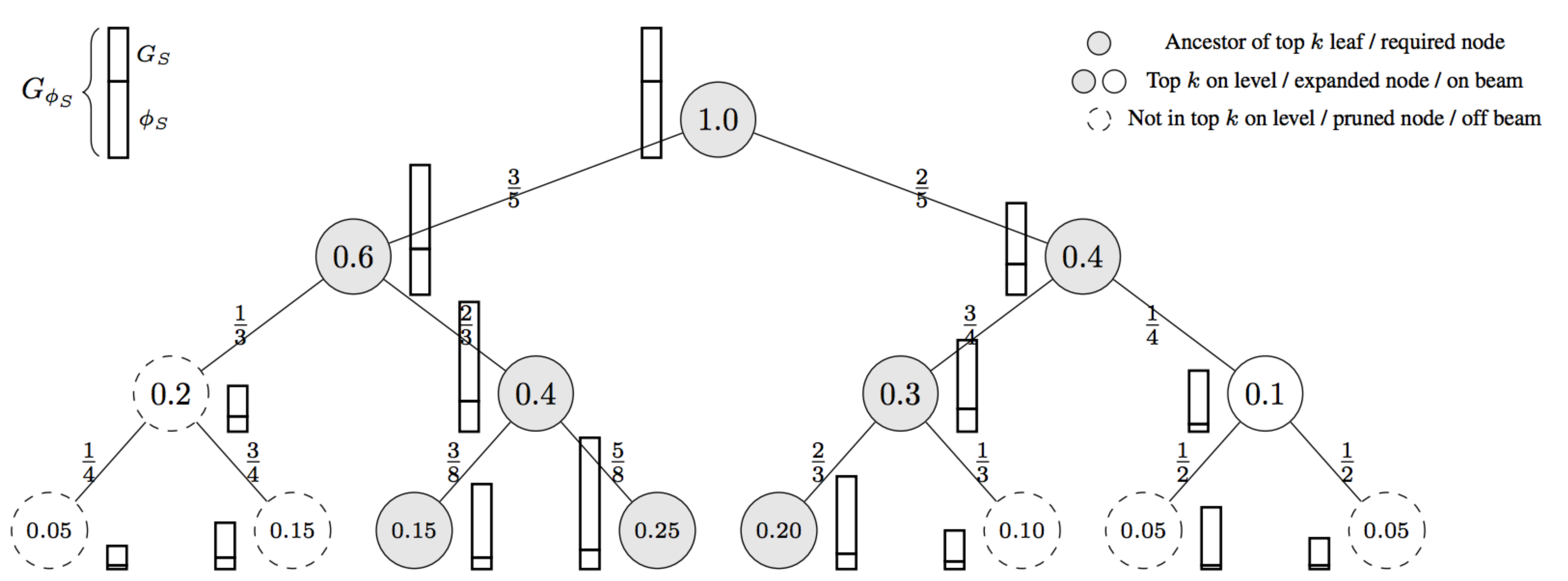


Figure 1. Example of the Gumbel-Top-k trick on a tree, with $k = 3$. The bars next to the leaves indicate the perturbed log-probabilities G_{ϕ_i}, while the bars next to internal nodes indicate the maximum perturbed log-probability of the set of leaves S in the subtree rooted at that node: $G_{\phi_S} = \max_{i \in S} G_{\phi_i} \sim \text{Gumbel}(\phi_S)$ with $\phi_S = \log \sum_{i \in S} \exp \phi_i$. The bar is split in two to illustrate that $G_{\phi_S} = \phi_S + G_S$. Numbers in the nodes represent $p_{\boldsymbol{\theta}}(\boldsymbol{y}^S) = \exp \phi_S = \sum_{i \in S} \exp \phi_i$, the probability of the partial sequence $\boldsymbol{y}^S$. Numbers at edges represent the conditional probabilities for the next token. The shaded nodes are ancestors of the top k leaves with highest perturbed log-probability G_{ϕ_i}. These are the ones we actually need to expand. In each layer, there are at most k such nodes, such that we are guaranteed to construct all top k leaves by expanding at least the top k nodes (ranked on G_{ϕ_S}) in each level (indicated by a solid border).

Method for sampling sequences without replacement

Come see at ICML | 2019 Thirty-sixth International Conference on Machine Learning

Deep RL meets Structured Prediction

ICLR 2019 workshop

Experiment

- Learn to predict tour (sequence) for TSP (Kool et al., 2019a)
- Estimators:
  - **Single sample** with a **batch** baseline
  - **Single sample** with **greedy rollout** baseline (Kool et al., 2019a)
  - **Multiple samples** with replacement (WR) with **local** baseline
  - **Multiple samples** without replacement (WOR) with **local** baseline

Idea 💡

- Take multiple samples per datapoint
- Encoder-decoders: run encoder only once
- Data- and computational efficiency

Travelling S(alesman|cientist) Problem (TSP)

Goal? Learn heuristic algorithms automatically!

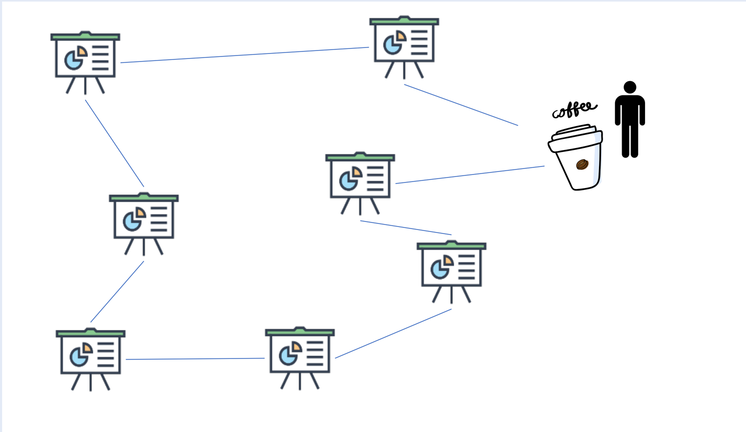
Why? Problem is (NP-)hard, development costly!

How? 'Translate' problem into solution…

Math?

- Instance $x = ((x_1, y_1), (x_2, y_2), \dots, (x_n, y_n))$
- Solution $y = (y_1, y_2, \dots y_n)$ e.g. $(3,1,2,4)$
- Model $p_{\boldsymbol{\theta}}(y|x) = \prod_{t=1}^{n} p_{\boldsymbol{\theta}}(y_t|s, y_{1:t-1})$
- Minimize expected tour length $f(y, x)$:

$$\min_{\boldsymbol{\theta}} E_{p_{\boldsymbol{\theta}}(y|x)}[f(y, x)]$$

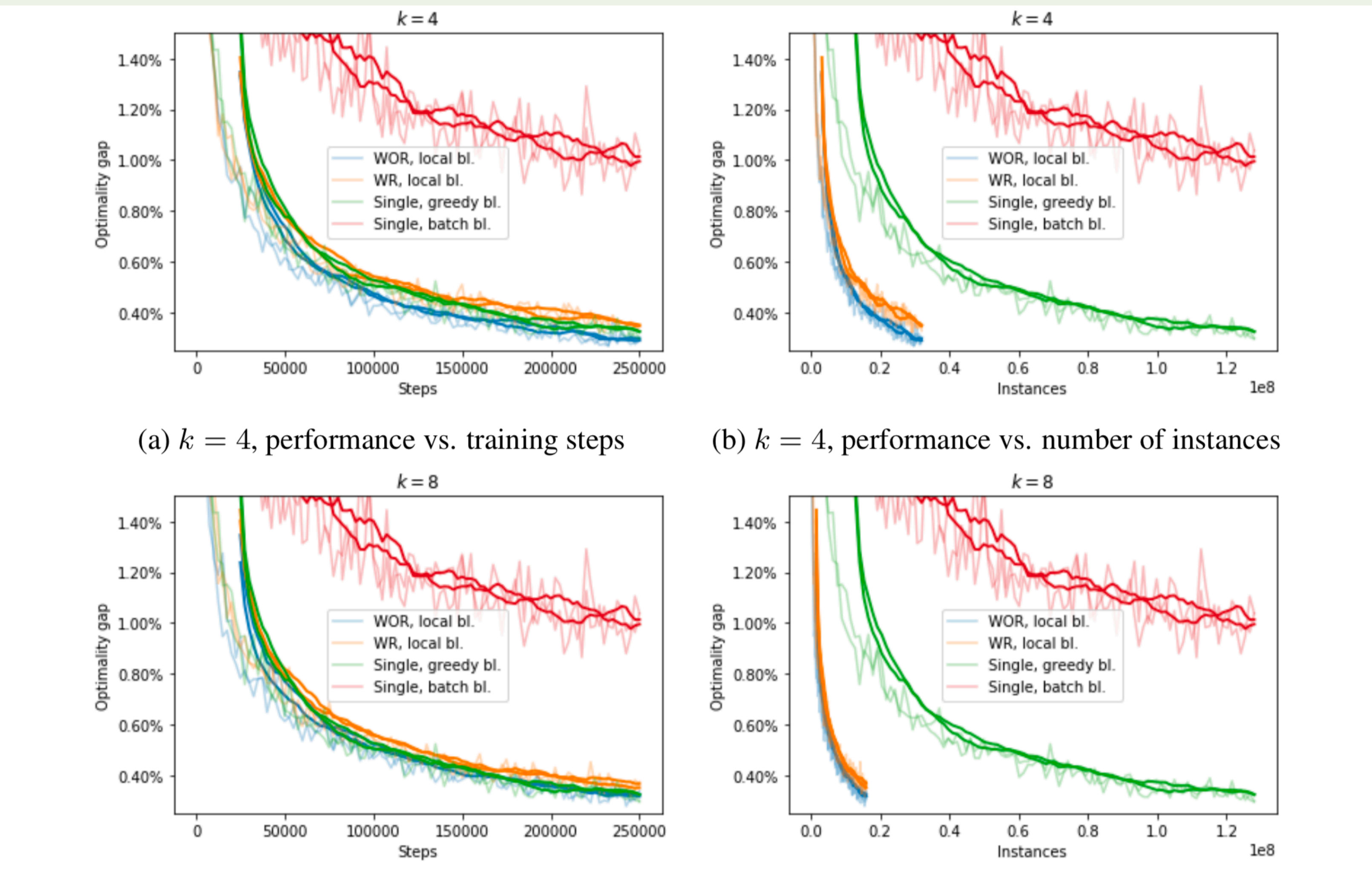


Travelling Scientist Problem

Also at ICLR | 2019 Seventh International Conference on Learning Representations

(Kool et al., 2019a)

Results for TSP (20 nodes)



(a) $k = 4$, performance vs. training steps (b) $k = 4$, performance vs. number of instances

(c) $k = 8$, performance vs. training steps (d) $k = 8$, performance vs. number of instances

Figure 1: Performance measured as validation set optimality gap during training. Raw results are light, smoothed results are darker (2 random seeds per setting). REINFORCE is used with replacement (WR) and without replacement (WOR) using $k = 4$ (top row) or $k = 8$ (bottom row) samples per instance, and a local baseline based on the k samples for each instance. We compare against REINFORCE using one sample per instance, either with a baseline that is the average of the batch, or the strong greedy rollout baseline by Kool et al. (2019a) that requires an additional rollout of the model.

Conclusion

- Requires less data for same performance
- Sampling without replacement increases performance
- Especially well suited for structured prediction settings

References

- Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2019a.
- Wouter Kool, Herke van Hoof, and Max Welling. Stochastic beams and where to find them: The gumbel-top-k trick for sampling sequences without replacement. In *International Conference on Machine Learning*, 2019b.