Error: Logits contain NaNs

Potential Reasons:

1. Exploding Gradients: During the computation of logits (which involve softmax), very large values might have been generated, leading to NaNs.

Potential Solutions:

(1) Gradient clipping:

Definition: Clip the error derivative to a threshold during backward propagation through the network

Experiments: Tried to add gradient clipping with values of 1 and 0.5 in the RL4COTrainer.

Result: Did not solve the problem.

(2) Weight Initialization:

Definition: Using a proper weight initialization strategy, such as Xavier or He initialization, can help prevent gradients from becoming too large at the start of training.

(3) Batch Normalization:

2. Numerical Instability in Attention Mechanism: The attention mechanism itself might be causing numerical issues, especially if the softmax temperature is too low.

Potential Solutions:

(1) Hyper-tunning the parameters (e.g. softmax temperature)

3. AttentionModelPolicy class, a subclass of AutoregressivePolicy class, has evolved a lot in the newest version.

<https://github.com/ai4co/rl4co/blob/main/rl4co/models/zoo/am/policy.py>

This could solve the problem because it provides more parameters for tuning. These parameters are fixed in the current rl4co version.

(1) feedforward\_hidden: Dimension of the hidden layer in the feedforward network

(2) encoder\_network: Network to use for the encoder

(3) linear\_bias\_decoder: Whether to use a bias in the linear layer of the decoder

(4) out\_bias\_pointer\_attn: Whether to use a bias in the pointer attention

(5) check\_nan: Whether to check for nan values during decoding

4. Learning Rate is not optimal:

Experiment Models Log:

1. Model #1: Training on Butterfly floorplan with max\_Epoch = 3, Batch\_size = 3

1. Results: Training not stable, results not good, high loss for all the floorplans

2. Model #2: Training on Butterfly floorplan with no max\_Epoch limitation, Batch\_size = 3

1. Training stops at Epoch 36 (Report “Logits contain NaNs” error after Epoch 36)

2. Results: Results not good, high loss for all the floorplans

3. Model #3: Training on Butterfly floorplan with no max\_Epoch limitation, Batch\_size = 4

1. Training stops at Epoch 5

2. Results: Training not stable, results not good, high loss for all the floorplans

4. Model #4: Training on Butterfly floorplan with no max\_Epoch limitation, Batch\_size = 4, gradient\_clipping \_val = 0.5, “Greedy” decode for training, validation, and testing

1. Training becomes normal and continuous, I manually stop it at 200

2. Results: Visually improvements! But sometimes it was still stuck in the loop (especially in complicated floorplans)

4. Model #5: Training on Butterfly floorplan with POMO, max\_Epoch = 100, Batch\_size = 4, gradient\_clipping \_val = 0.5, “multistart\_greedy” decode for training