Supplementary text for Nested Qubit Routing

1 Runtime tables

In this section, we present the runtime tables for different benchmarks shown in the main text. The results for GNRPA are also included in this section. The small circuit benchmark runtime table is attached as a csv file along with the code.

1.1 Random circuit benchmark

The runtimes are presented in Table 1. Note that these results are averaged over 10 runs for any given number of gates.

Gates	NesQ	Cirq	Qroute	GNRPA	NesQ+
30	1.741882	0.095985	57.910772	80.375707	1.875454
40	2.321188	0.137800	91.984908	116.2286	2.524887
50	3.224417	0.185684	114.231547	158.796500	3.862081
60	4.343773	0.329320	182.626015	233.352610	5.248788
70	4.880676	0.414190	203.357469	233.359562	5.783488
80	5.690395	0.515367	248.183977	255.631882	6.534142
90	8.582040	0.704814	242.724301	312.194811	7.850141
100	7.485916	0.923527	263.054020	377.962436	8.297419
110	8.045412	1.279350	399.084266	400.819585	9.705672
120	9.823262	1.537304	363.455770	450.026435	9.970981
130	10.797684	2.012254	393.310650	503.998836	11.788189
140	12.106278	2.219034	443.379280	581.633442	10.778473
150	12.924456	3.202757	528.393336	650.770277	13.553065

Table 1: Average Runtimes (of 10 runs) for random circuit benchmark (in seconds)

1.2 Large circuit benchmark

The runtimes for different large scale circuits are shown in Table 2.

Circuit Name	Gates	NesQ	Cirq	Qroute	NesQ+
rd84_142	154	0.202823	0.037918	12.207869	0.270372
adr4_197	1498	11.086658	54.814014	99.807621	14.060009
radd_250	1405	9.683338	46.239436	86.420729	13.461079
z4_268	1343	7.301565	35.613734	131.339545	9.568750
sym6_145	1701	10.237229	58.775746	1944.404549	11.602539
misex1_241	2100	18.023073	142.878779	132.625253	23.869278
rd73_252	2319	19.430781	198.861971	175.083901	30.877758
cycle10_2_110	2648	27.617641	321.146342	195.689001	38.172365
square_root_7	3089	35.293369	356.253962	214.469363	45.133008
sqn_258	4459	66.416509	1491.360893	330.1394166	85.826042
rd84_253	5960	120.109507	3986.860282	489.3008025	154.411345

Table 2: Runtimes for large realistic circuit benchmark in minutes.

Circuit Name	Device	NesQ+	Qroute
radd_250	qx5	11.30	133.39
	qx20	12.05	89.18
	acorn	13.30	245.71
z4_268	qx5	8.61	1238.17
	qx20	9.64	119.50
	acorn	10.96	1652.84
rd73_252	qx5	25.34	1396.74
	qx20	25.80	165.37
	acorn	29.18	2009.93
cycle10_2_110	qx5	34.80	356.34
	qx20	36.07	188.15
	acorn	43.84	460.99
sqn_258	qx5	91.05	1833.69
	qx20	87.89	315.64
	acorn	113.56	1015.75

Table 3: Time taken (in minutes) for NesQ+ and Qroute to route selected large circuits over across different devices topologies.

1.3 Device benchmark

The time taken by NesQ+ and Qroute to route the chosen circuits on different devices is given in Table 3.

2 Generalised Nested Rollout Policy Adaptation (GNRPA)

Nested Rollout Policy Adaptation (NRPA) (Rosin, 2011) combines nested search, memorizing the best sequence of moves found at each level, and the online learning of a playout policy using this sequence. NRPA has world records in Morpion Solitaire and crossword puzzles and has also been applied to many other combinatorial problems such as the Traveling Salesman Problem with Time Windows (Edelkamp et al., 2013), 3D Packing with Object Orientation (Edelkamp et al., 2014), the physical traveling salesman problem (Edelkamp and Greulich, 2014), the Multiple Sequence Alignment problem (Edelkamp and Tang, 2015), Logistics, Graph Coloring, Vehicle Routing Problems, Network Traffic Engineering, Virtual Network Embedding (Elkael et al., 2022) or the Snake in the Box.

Generalized Nested Rollout Policy Adaptation (GNRPA) generalizes the way the probability is calculated using a bias. The bias is a heuristic that performs non uniform playouts and using it usually gives much better results than uniform playouts. The use of a bias has been theoretically demonstrated more general than the initialization of the weights. The GNRPA paper also provides a theoretical derivation of the learning of the policy, using a cross entropy loss associated to a softmax. GNRPA has been applied to some difficult problems such as Inverse RNA Folding and Vehicle Routing Problems (Sentuc et al., 2022) with better results than NRPA.

While surveying different MCTS-based algorithms for solving the qubit routing problem, we experimented with Generalised Nested Rollout Policy Adaptation and observed results at-par with most of the routers (see Section 2.2), except when circuit size increases (i.e. large circuit experiment).

2.1 Algorithm

The idea behind GNRPA is to learn a rollout policy by adapting the weights on each action. These weights of choosing an action are modelled by first, hashing a move m using a *code* function which gives each action a unique address and then mapping this to the weights using *policy* function. The GNRPA algorithm (see Algorithm 3) performs fixed number of calls (N iterations) to the lower level search and adapt function. At its base level, the algorithm performs playout (see Algorithm 1) with a probability equal to the softmax function applied to the weights plus the bias of the possible moves. The idea of finding a new sequence of moves from lower level searches and updating it based on score (reward) holds as in NMCS. In each iteration, the policy is also adapted (via Algorithm 2) by pushing the weights according to moves in the best sequence found yet.

Algorithm 1 The playout algorithm

playout (*policy*)
  $state \leftarrow root$
  while true **do**
    if terminal($state$) **then**
      return (score ($state$), $sequence$($state$))
    end if
    $z \leftarrow 0$
    for $m \in$ possible moves for $state$ **do**
      $o[m] \leftarrow e^{policy[code(m)]+\beta_m}$
      $z \leftarrow z + o[m]$
    end for
    choose a $move$ with probability $\frac{o[move]}{z}$
    play ($state$, $move$)
  end while
=0

Algorithm 2 The adapt algorithm

adapt (*policy*, *sequence*)
  $polp \leftarrow policy$
  $state \leftarrow root$
  for $b \in sequence$ **do**
    $z \leftarrow 0$
    for $m \in$ possible moves for $state$ **do**
      $o[m] \leftarrow e^{policy[code(m)]+\beta_m}$
      $z \leftarrow z + o[m]$
    end for
    for $m \in$ possible moves for $state$ **do**
      $p_m \leftarrow \frac{o[m]}{z}$
      $polp[code(m)] \leftarrow polp[code(m)] - \alpha(p_m - \delta_{bm})$
    end for
    play ($state$, b)
  end for
  $policy \leftarrow polp$
=0

Algorithm 3 The GNRPA algorithm

GNRPA (*level*, *policy*)
  if level == 0 **then**
    return playout (*policy*)
  else
    $bestScore \leftarrow -\infty$
    for N iterations **do**
      (score,new) $\leftarrow$ GNRPA($level - 1$, $policy$)
      if score $\geq$ bestScore **then**
        bestScore $\leftarrow$ score
        seq $\leftarrow$ new
      end if
      $policy \leftarrow$ adapt ($policy$, seq)
    end for
    return (bestScore, seq)
  end if
=0

2.2 Results

We ran level 1 GNRPA router for 250 iterations on random and small quantum ciruit benchmarks. The reason we leave out it for large scale quantum circuits is that we found it to take atleast 25 hours for circuits with gate count of 1498 (adr4_197) or more.

2.2.1 Random circuit benchmark

The runtimes are presented in Table 1. GNRPA maintains a slightly higher runtime than Qroute for all circuits (25.29% on average) but successfully retains a lower circuit depth than Qroute (by 20.23%), Cirq (by 31.86%), Qiskit basic (by 35.69%), Qiskit sabre (by 34.38%) and Pytket (by 7.42%). NesQ was able to beat GNRPA by 24.6% lower output circuit depth, reinforcing the superiority of our algorithm.

2.2.2 Small realistic circuits benchmark

GNRPA exhibited a cumulative circuit depth of 3324 in this benchmark, next to Qiskit stochastic (3016) and lower than the worst performing router, Cirq (3616). GNRPA took 234 minutes to route the dataset which was almost twice compared to Qroute (121 minutes).

References

Stefan Edelkamp and Christoph Greulich. Solving physical traveling salesman problems with policy adaptation. In *Computational Intelligence and Games (CIG), 2014 IEEE Conference on*, pages 1–8. IEEE, 2014.

Stefan Edelkamp and Zhihao Tang. Monte-Carlo tree search for the multiple sequence alignment problem. In *Proceedings of the Eighth Annual Symposium on Combinatorial Search, SOCS 2015*, pages 9–17. AAAI Press, 2015.

Stefan Edelkamp, Max Gath, Tristan Cazenave, and Fabien Teytaud. Algorithm and knowledge engineering for the tsptw problem. In *Computational Intelligence in Scheduling (SCIS), 2013 IEEE Symposium on*, pages 44–51. IEEE, 2013.

Stefan Edelkamp, Max Gath, and Moritz Rohde. Monte-Carlo tree search for 3d packing with object orientation. In *KI 2014: Advances in Artificial Intelligence*, pages 285–296. Springer International Publishing, 2014.

Maxime Elkael, Massinissa Ait Aba, Andrea Araldo, Hind Castel-Taleb, and Badii Jouaber. Monkey business: Reinforcement learning meets neighborhood search for virtual network embedding. *Computer Networks*, 216: 109204, 2022.

Christopher D. Rosin. Nested rollout policy adaptation for Monte Carlo Tree Search. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pages 649–654, 2011.

Julien Sentuc, Tristan Cazenave, and Jean-Yves Lucas. Generalized nested rollout policy adaptation with dynamic bias for vehicle routing. In *AI for Transportation at AAAI*, 2022.