Smart Pass Manager:
A Noise-Aware Transpiler Pass for Qiskit

Ruben E. Leuzinger

June 15, 2025

Abstract

Smart Pass Manager (`smart_transpile`) is a noise-aware transpiler extension for Qiskit that selects a connected subset of physical qubits on an IBM Quantum device by minimising a composite cost function of two-qubit, single-qubit, read-out and time-to-decoherence penalties. A few exploratory single-circuit tests already indicate a modest fidelity gain for circuits with more than eight qubits and non-trivial depth, but a full benchmark campaign is still pending. Source code and notebooks are available at github.com/RubenLeuz/smart_pass_manager.

1 Introduction

Noisy Intermediate-Scale Quantum (NISQ) devices [1] exhibit heterogeneous error rates across qubits and couplers. Standard Qiskit mapping workflows optimise depth and SWAP count but ignore these variations. Smart Pass Manager leverages live calibration data during transpilation to reduce aggregate infidelity, following ideas in noise-adaptive mapping [2].

2 Design and Implementation

2.1 Calibration Data

From `backend.properties()` we obtain for every physical qubit or qubit pair:

- two-qubit gate error $\epsilon_2(i \to j)$ and gate duration $t_2(i \to j)$,
- single-qubit gate error $\epsilon_1(q)$,
- read-out error $r(q)$,
- relaxation time $T_1(q)$.

Mean values such as $\overline{\epsilon_2}$ and $\overline{T}_1$ are cached for automatic hyper-parameter scaling.

2.2 Composite Edge Weight

A directed coupling graph $G(V, E)$ is formed and each edge $(i \to j)$ receives the cost

$$w(i \to j) = \epsilon_2(i \to j) + \alpha\,[\epsilon_1(i) + \epsilon_1(j)] + \beta\,[r(i) + r(j)] + \delta\,\frac{t_2(i \to j)}{\overline{T}_1} + \gamma + \mathrm{Penalty}_{\mathrm{rev}}(j \to i), \quad (1)$$

where $\alpha, \beta, \gamma, \delta$ are (optionally auto-scaled) hyper-parameters and $\mathrm{Penalty}_{\mathrm{rev}}$ discourages non-native gate directions.

Hyper-parameter interpretation

- α: scales single-qubit errors, default $\alpha = \overline{\epsilon_2}/\overline{\epsilon_1}$.
- β: read-out weight inherited from α via $\beta = \alpha/2$.
- γ: compactness bias, default $\gamma = \overline{\epsilon_2}/k$, where k is the number of qubits in the circuit.
- δ: decoherence penalty, fixed default $\delta = 0.01$.

Users may override any of these values when calling `smart_transpile`; setting a parameter to `None` activates the auto-scaling rule shown above.

2.3 Patch Selection Heuristic

For k logical qubits we need a connected k-node patch of minimal total weight. The implementation uses *multi-start greedy growth*:

1. Pick up to `num_starts` random seed edges.
2. Grow each patch by repeatedly adding the lightest neighbouring node.
3. Return the patch with the lowest aggregate weight.

2.4 Layout Integration

The selected patch becomes an `InitialLayout`. A single-iteration `SabreLayout` refines the mapping, then the standard `transpile` pipeline completes routing and optimisation. Patch details and hyper-parameters are stored in circuit metadata (`smartlayout`) for reproducibility.

3 Preliminary Tests and Future Benchmarking

A handful of random single-circuit experiments—executed on IBM *Fake* back-ends—already suggest that `smart_transpile` yields slightly higher success probabilities once the problem size exceeds eight qubits and the circuit depth becomes non-trivial.

Rigorous benchmarking, however, remains outstanding. Simulating a noisy 20-qubit circuit with depth beyond 20 can already occupy classical resources for *multiple hours*; scaling this to a comprehensive grid of qubit counts and depths is therefore expensive but essential for a definitive performance assessment. Securing compute time and completing that study is the main priority going forward.

4 Conclusion and Outlook

Smart Pass Manager adds noise awareness to Qiskit's transpilation with minimal user intervention. Future work could include:

- full benchmark grid over qubit count, depth and real hardware,
- optional local-refinement phase for patch improvement,
- modelling crosstalk and non-Markovian noise,
- extension to pulse-level optimisation.

Acknowledgments

I thank IBM [3, 4] for providing freely accessible, high-quality learning material on Qiskit and quantum computing. Code-review and writing feedback were supplied by ChatGPT (OpenAI o3 and o4-mini-high); all AI-generated suggestions were carefully reviewed and verified by me. This work builds on the foundational ideas of Preskill [1] and Murali *et al.* [2].

References

[1] J. Preskill, ‘Quantum Computing in the NISQ Era and Beyond,” *Quantum*, 2018.

[2] P. Murali *et al.*, ‘Noise-Adaptive Compiler Mappings for NISQ Computers,” in *Proc. ASPLOS*, 2019.

[3] IBM Quantum, ‘IBM Quantum Learning Portal,” 2025. `https://learning.quantum.ibm.com`

[4] IBM Quantum, ‘IBM Quantum Platform Documentation: Qiskit Runtime and Circuit Guides,” 2025. `https://quantum.cloud.ibm.com/docs/de/guides`