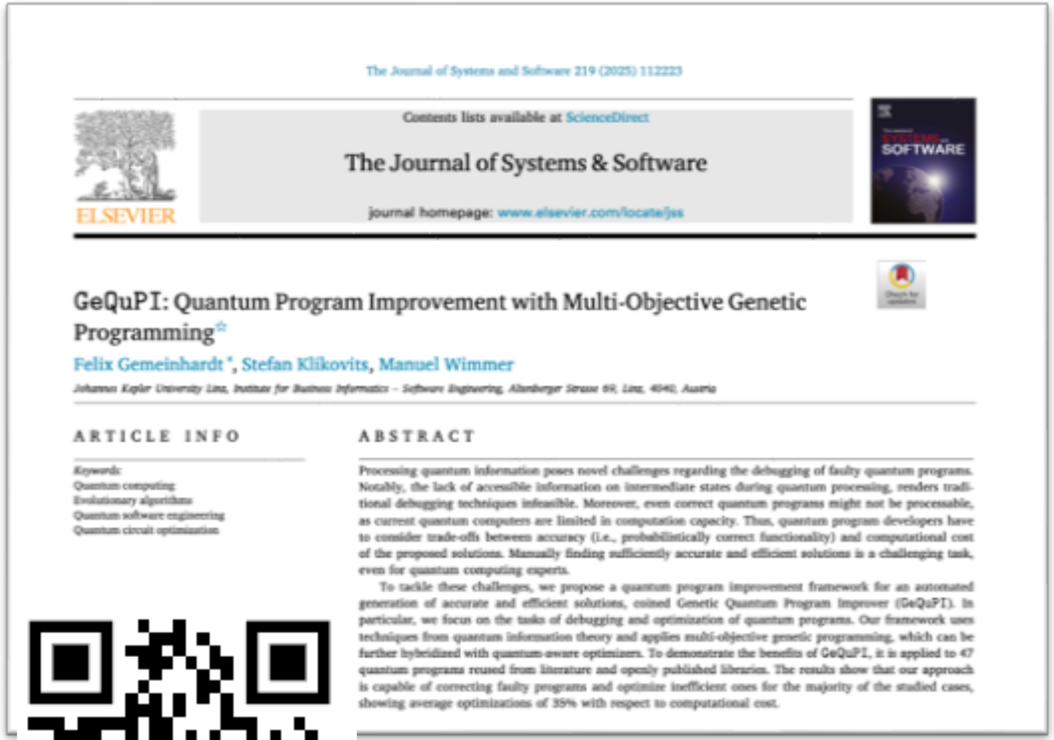
JOHANNES KEPLER
UNIVERSITY LINZ

GeQuPI:
Quantum Program Improvement with Multi-Objective Genetic Programming

F. Gemeinhardt, S. Klikovits, M. Wimmer

Institute of Business Informatics – Software Engineering (BISE)

JOHANNES KEPLER UNIVERSITY LINZ



The Journal of Systems and Software 219 (2025) 112223

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

GeQuPI: Quantum Program Improvement with Multi-Objective Genetic Programming☆

Felix Gemeinhardt *, Stefan Klikovits, Manuel Wimmer

Johannes Kepler University Linz, Institute for Business Informatics – Software Engineering, Altenberger Strasse 69, Linz, 4040, Austria

ARTICLE INFO

Keywords:
Quantum computing
Evolutionary algorithms
Quantum software engineering
Quantum circuit optimization

ABSTRACT

Processing quantum information poses novel challenges regarding the debugging of faulty quantum programs. Notably, the lack of accessible information on intermediate states during quantum processing, renders traditional debugging techniques infeasible. Moreover, even correct quantum programs might not be processable, as current quantum computers are limited in computation capacity. Thus, quantum program developers have to consider trade-offs between accuracy (i.e., probabilistically correct functionality) and computational cost of the proposed solutions. Manually finding sufficiently accurate and efficient solutions is a challenging task, even for quantum computing experts.

To tackle these challenges, we propose a quantum program improvement framework for an automated generation of accurate and efficient solutions, coined Genetic Quantum Program Improver (GeQuPI). In particular, we focus on the tasks of debugging and optimization of quantum programs. Our framework uses techniques from quantum information theory and applies multi-objective genetic programming, which can be further hybridized with quantum-aware optimizers. To demonstrate the benefits of GeQuPI, it is applied to 47 quantum programs reused from literature and openly published libraries. The results show that our approach is capable of correcting faulty programs and optimize inefficient ones for the majority of the studied cases, showing average optimizations of 35% with respect to computational cost.

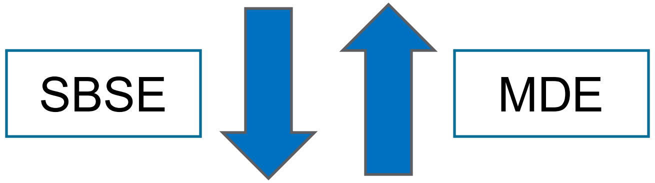
Quantum Software Team at BISE

Felix Gemeinhardt

Stefan Klikovits

Manuel Wimmer (Head)

Christoph Stein

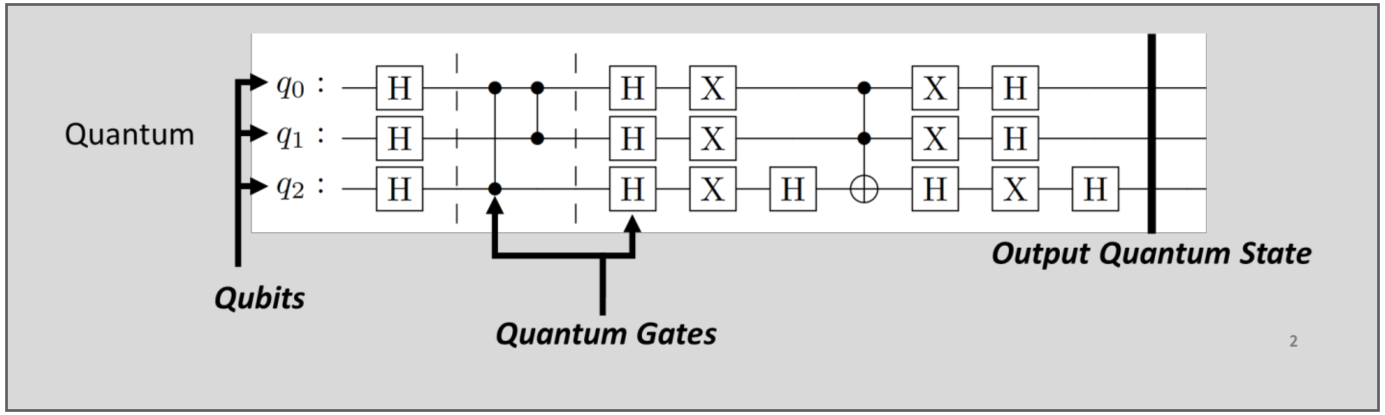


Research Focus: Automated Quantum Software Engineering

- Direction I: Model-Driven Quantum Software Engineering
- **Direction II: Search-Based Quantum Software Engineering**

Context & Problem Setting

Quantum Circuits



Challenges

– Vast design space
– Precision vs. computational cost
– Errors, Probabilistic Nature & NISQ

Search-Based Software Engineering

Extensively used for classical software systems for about two decades [Harman et al. 2012]

Goal: Find a software system / program that will optimize a given fitness function

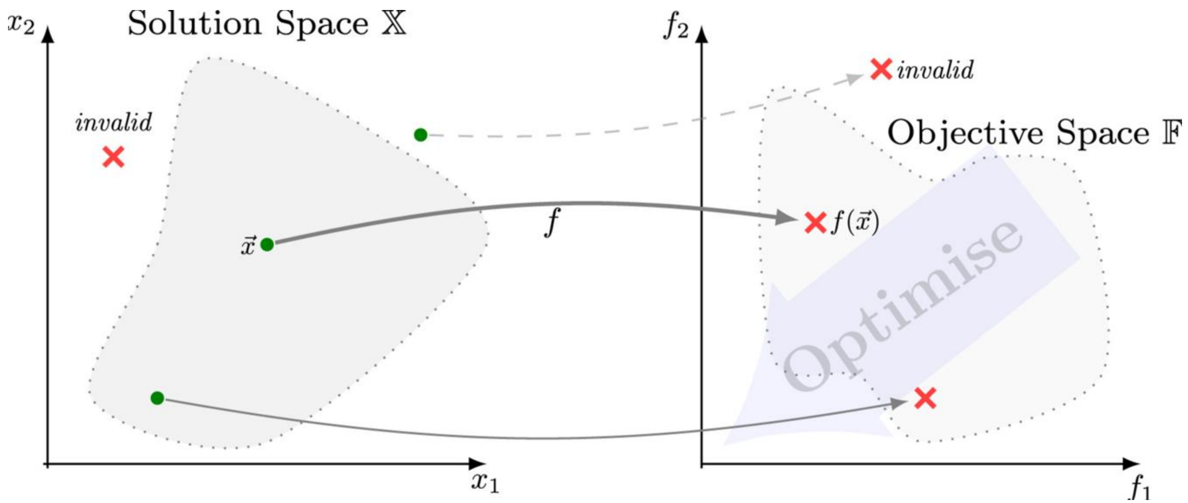
Advantages:

- Automated exploration by
- Meta-heuristic searchers

  Genetic Programming (GP), …

Challenges: Find a good

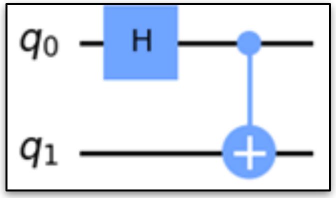
1. Program encoding
2. Fitness function



[Klikovits et al. 2023]

Typical GP Setup for Quantum Circuits

Encoding: gate vector



≅

```
[
  H(target=0),
  CNOT(target=1, control=0)
]
```

Fitness

<u>behaviour</u>: accuracy, and
<u>structure</u>: # gates, depth,
non-local gates, # parameters

Operators

mutation: add / delete / move / alter / swap gate
change qubits, …
crossover: one-point, two-point, …
selection: tournament, …

Algorithms

use of classical meta-heuristic algorithms:
GA, NSGA-II, NSGA-III, …

Is this all we need?

Using GP for QSE (1): Synthesis

GP4QSE

Circuit Synthesis

Operator Synthesis

[Gemeinhardt et al. 2023]

Hybrid Multi-Objective Genetic Programming for Parameterized Quantum Operator Discovery

Felix Gemeinhardt
Johannes Kepler University
Institute for Business Informatics - Software Engineering, CDL-MINT
Linz, Austria
felix.gemeinhardt@jku.at

Stefan Klikovits
Johannes Kepler University
Institute for Business Informatics - Software Engineering, CDL-MINT
Linz, Austria
stefan.klikovits@jku.at

Manuel Wimmer
Johannes Kepler University
Institute for Business Informatics - Software Engineering, CDL-MINT
Linz, Austria
manuel.wimmer@jku.at

ABSTRACT

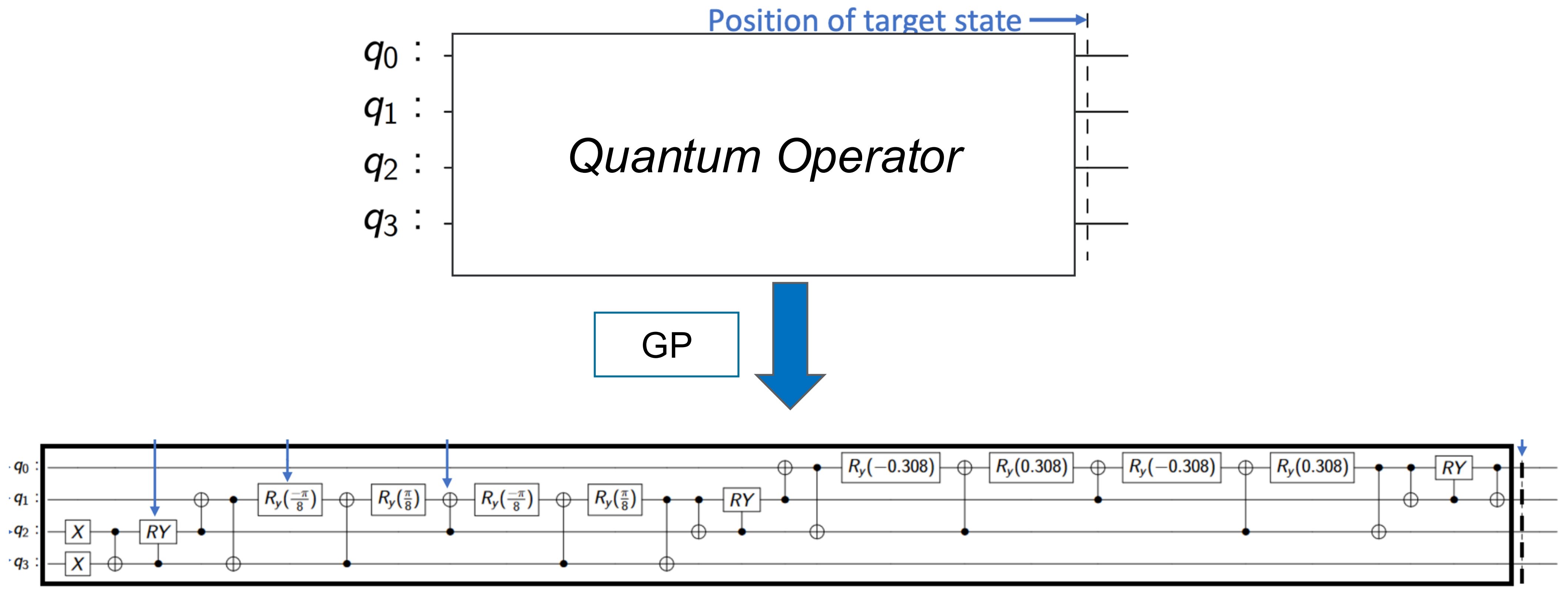
The processing of quantum information is defined by quantum circuits. For applications on current quantum devices, these are usually parameterized, i.e., they contain operations with variable parameters. The design of such quantum circuits and aggregated higher-level quantum operators is a challenging task which requires significant knowledge in quantum information theory, provided a polynomial-sized solution can be found analytically at all. Moreover, finding an accurate solution with low computational cost represents a significant trade-off, particularly for the current generation of quantum computers. To tackle these challenges, we propose a multi-objective genetic programming approach—hybridized with a numerical parameter optimizer—to automate the synthesis of parameterized quantum operators. To demonstrate the benefits of the proposed approach, it is applied to a quantum circuit of a hybrid quantum-classical algorithm, and then compared to an analytical solution as well as a non-hybrid version. The results show that

1 INTRODUCTION

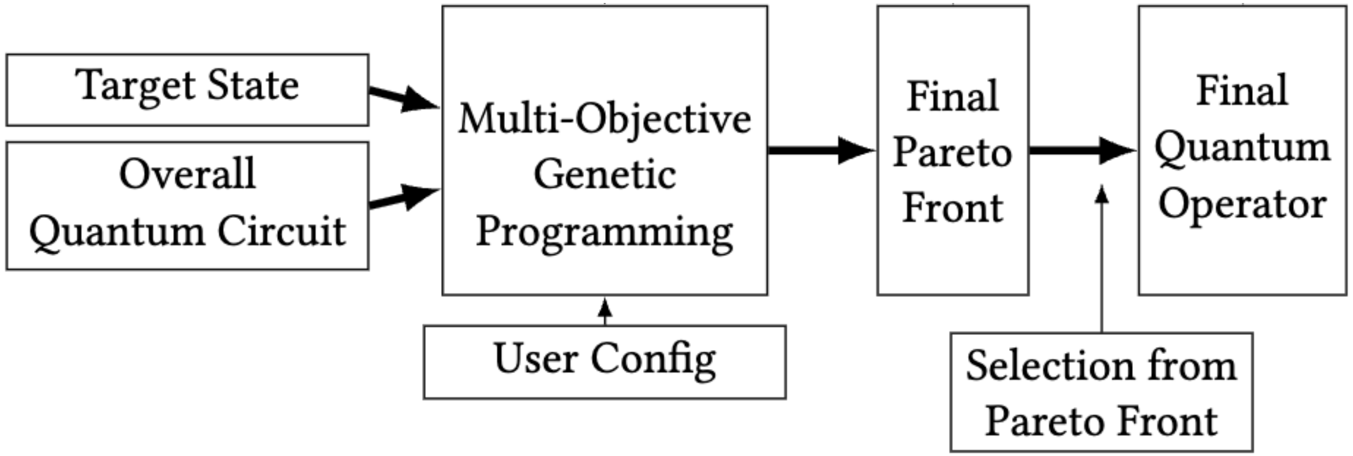
Quantum Computing. The current era of *Quantum Computing* (QC) is referred to as the *Noisy Intermediate-Scale Quantum* (NISQ) era, where the limitations of quantum hardware are mitigated by significant means of classical computation [16]. Analogously to logic gates for classical computation, in QC, quantum information is processed with operations called quantum gates. The most commonly used realistic model of QC is the so-called quantum circuit model [14]. Quantum gates can be parameterized, where the use of parameterized quantum circuits is common in the NISQ-era. This is because classical optimization of the parameters, which constitutes an NP-hard problem, allows to cope with the noise present in current quantum hardware [3]. For this reason, numerical parameter optimizers constitute a central element of NISQ-era quantum algorithms [3–5]. There is ongoing research on quantum-aware optimizers, which are particularly capable of coping with specific requirements of parameterized quantum circuits [3–5, 12].

JOHANNES KEPLER UNIVERSITY LINZ

Scenario 1 – Operator Synthesis



Our Approach: Search Scheme using GP

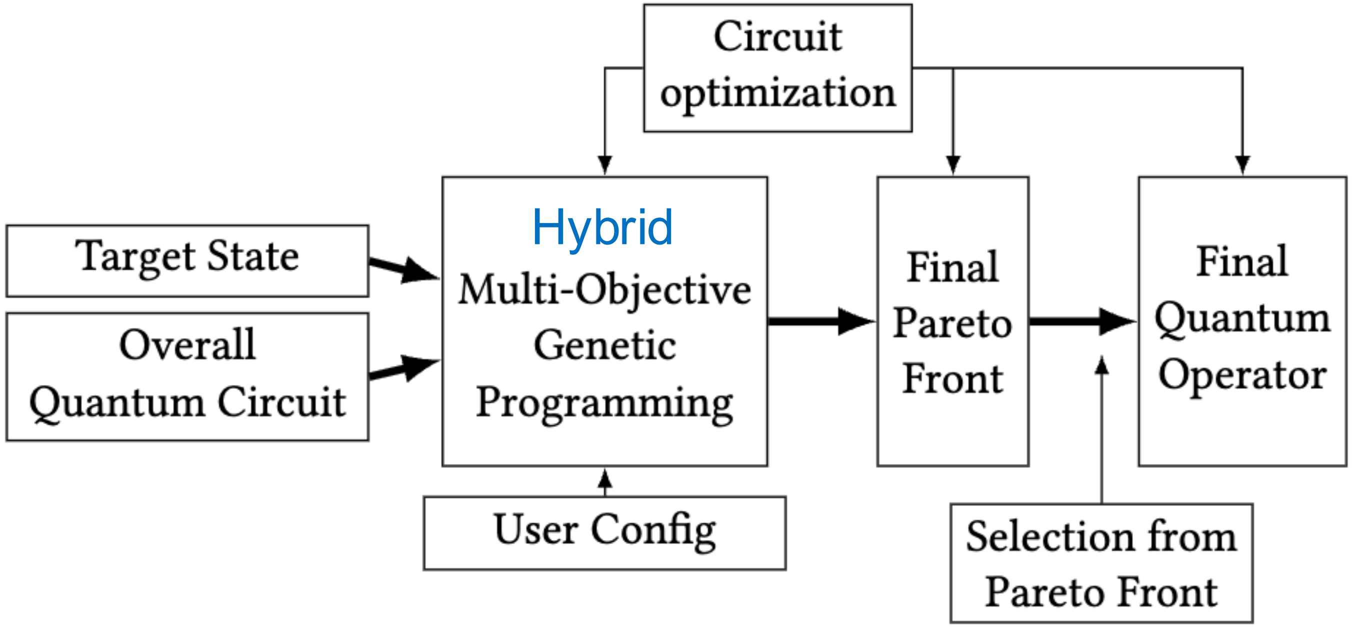


Our Approach: Search Scheme using GP

Problem: Some gates (e.g., RX_θ,RY_θ, RZ_θ) require parameters $\theta \in (0, 2\pi)$

Solution: Use Hybrid Search Scheme

– Apply parameter optimizer (Nelder-Mead) inside GP

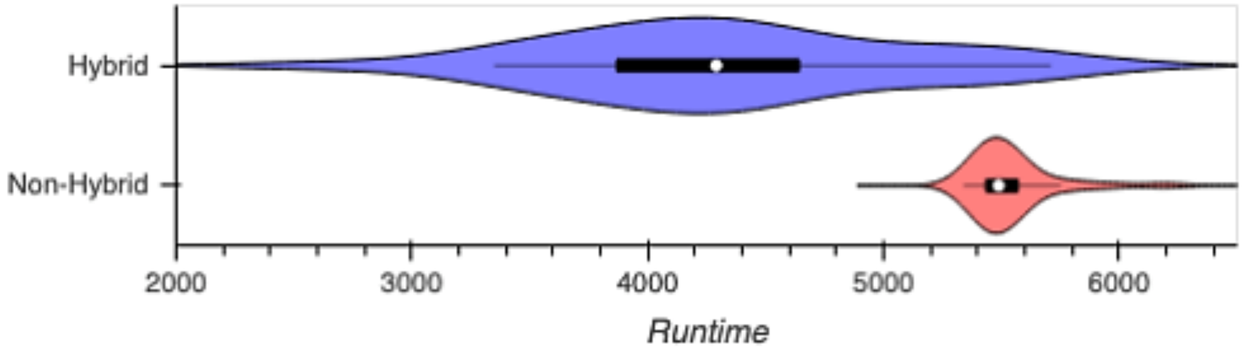


Experimental Evaluation

RQ1: Hybrid vs Non-Hybrid - **diversity**?
RQ2: Hybrid vs Non-Hybrid - **accuracy** (i.e., overlap)?
RQ3: Hybrid vs Non-Hybrid vs. Analytical Solution?

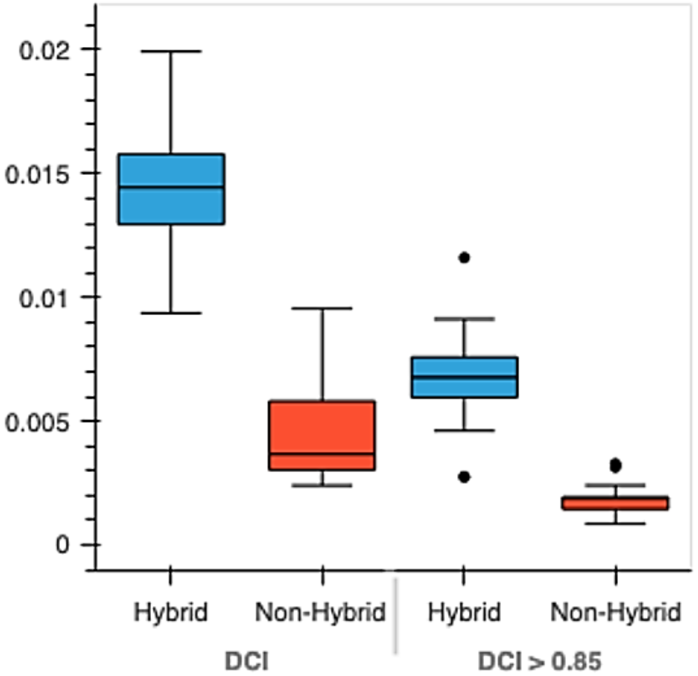
Selected Case: **GM-QAOA** [Bärtschi and Eidenbenz 2020]

Meta-heuristic search algorithm: NSGA-III
Implementation: Deap, Qiskit

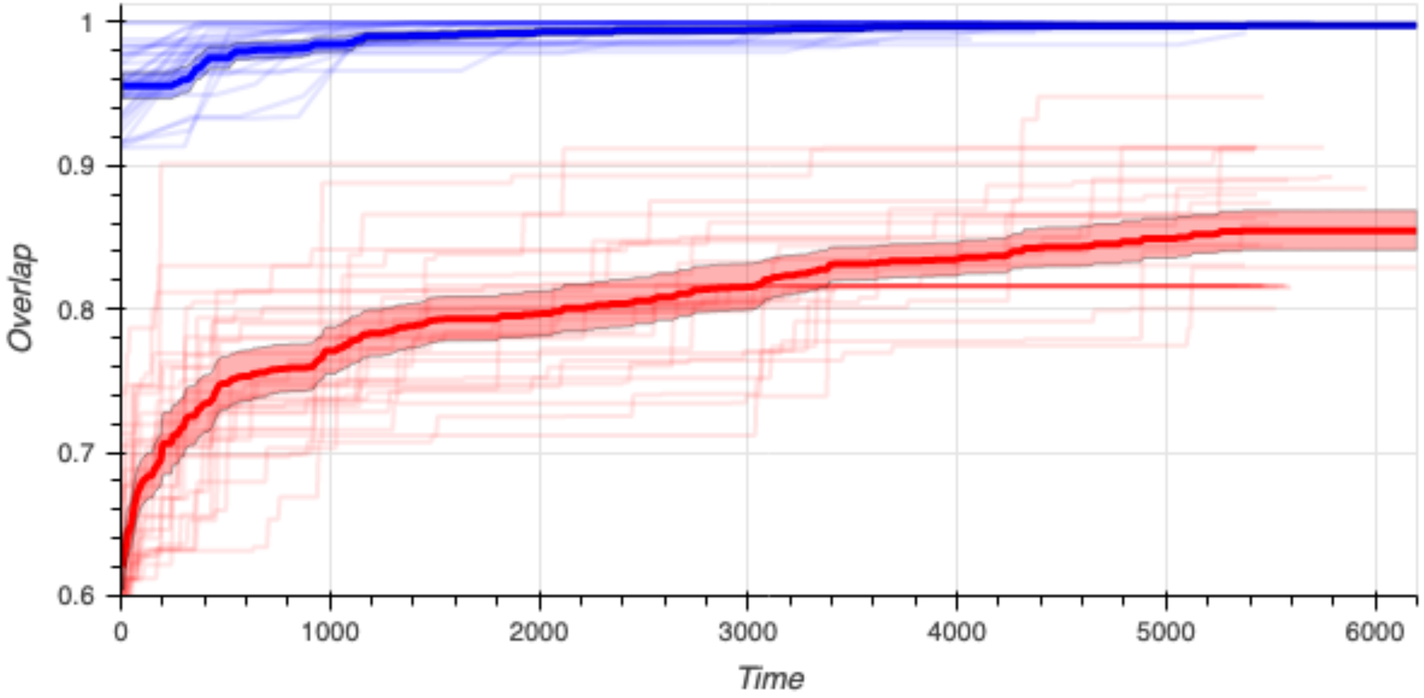


Results

Hybrid
Non-Hybrid



RQ1: better diversity



RQ2: Hybrid has higher overlap
+ faster convergence

Using GP for QSE (2): Improvement

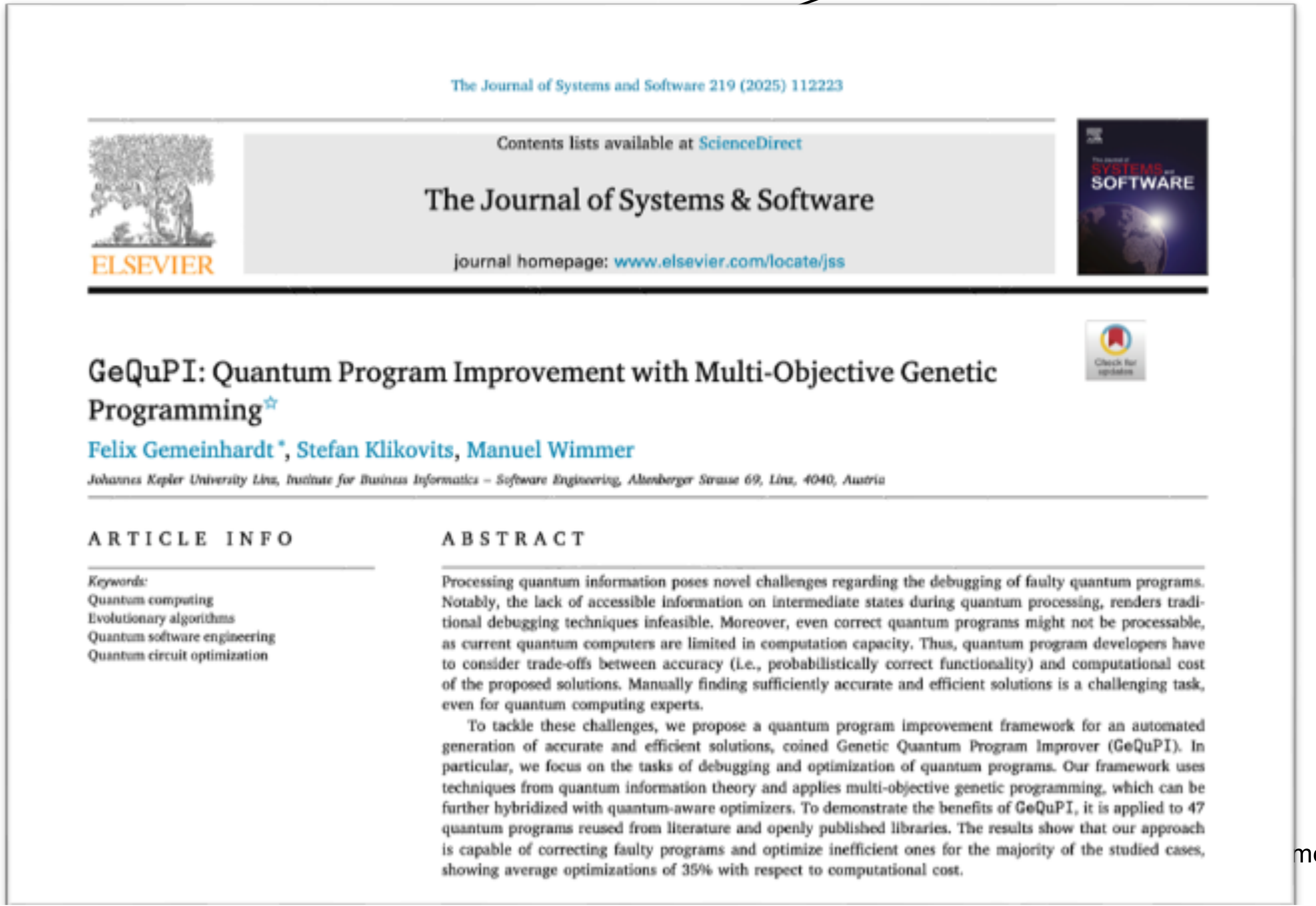
GP4QSE

Circuit Improvement

Optimization

Debugging

[Gemeinhardt et al. 2025]


The Journal of Systems and Software 219 (2025) 112223

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

GeQuPI: Quantum Program Improvement with Multi-Objective Genetic Programming

Felix Gemeinhardt *, Stefan Klikovits, Manuel Wimmer

Johannes Kepler University Linz, Institute for Business Informatics – Software Engineering, Altenberger Strasse 69, Linz, 4040, Austria

ARTICLE INFO

Keywords:
Quantum computing
Evolutionary algorithms
Quantum software engineering
Quantum circuit optimization

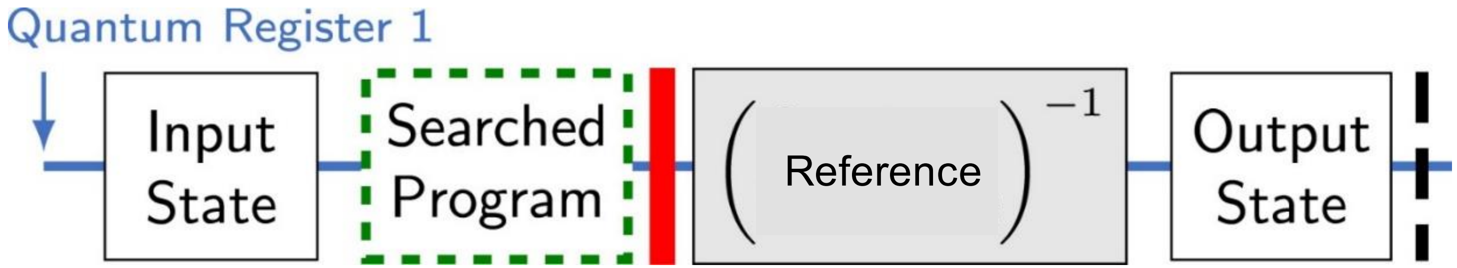
ABSTRACT

Processing quantum information poses novel challenges regarding the debugging of faulty quantum programs. Notably, the lack of accessible information on intermediate states during quantum processing, renders traditional debugging techniques infeasible. Moreover, even correct quantum programs might not be processable, as current quantum computers are limited in computation capacity. Thus, quantum program developers have to consider trade-offs between accuracy (i.e., probabilistically correct functionality) and computational cost of the proposed solutions. Manually finding sufficiently accurate and efficient solutions is a challenging task, even for quantum computing experts.

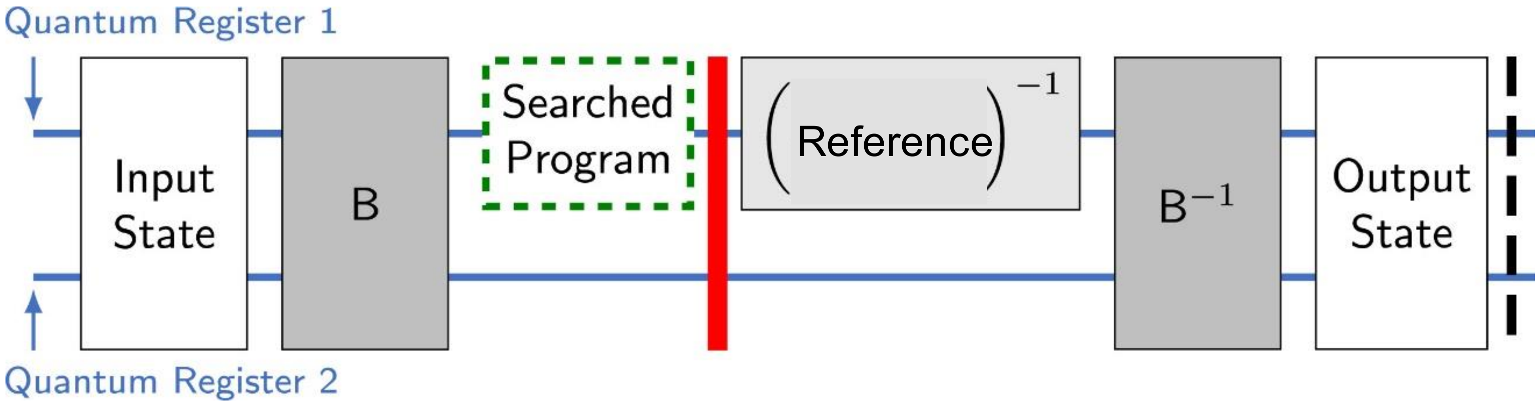
To tackle these challenges, we propose a quantum program improvement framework for an automated generation of accurate and efficient solutions, coined Genetic Quantum Program Improver (GeQuPI). In particular, we focus on the tasks of debugging and optimization of quantum programs. Our framework uses techniques from quantum information theory and applies multi-objective genetic programming, which can be further hybridized with quantum-aware optimizers. To demonstrate the benefits of GeQuPI, it is applied to 47 quantum programs reused from literature and openly published libraries. The results show that our approach is capable of correcting faulty programs and optimize inefficient ones for the majority of the studied cases, showing average optimizations of 35% with respect to computational cost.

Preliminary: Functionally-equivalent Quantum Programs

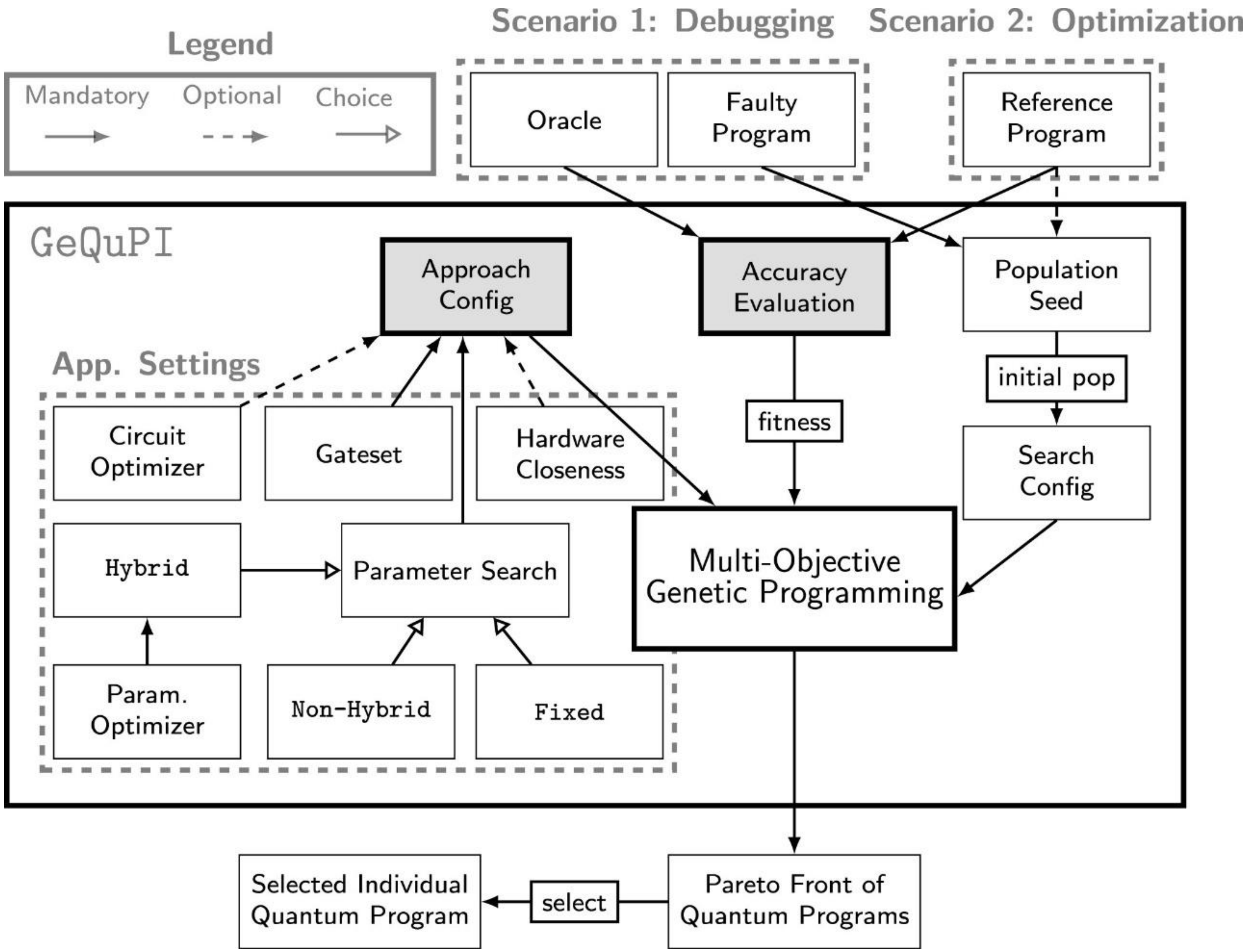
Reverse Reference and compare if Input == Output for <u>specific input states</u> [Burgholzer et al. 2020]



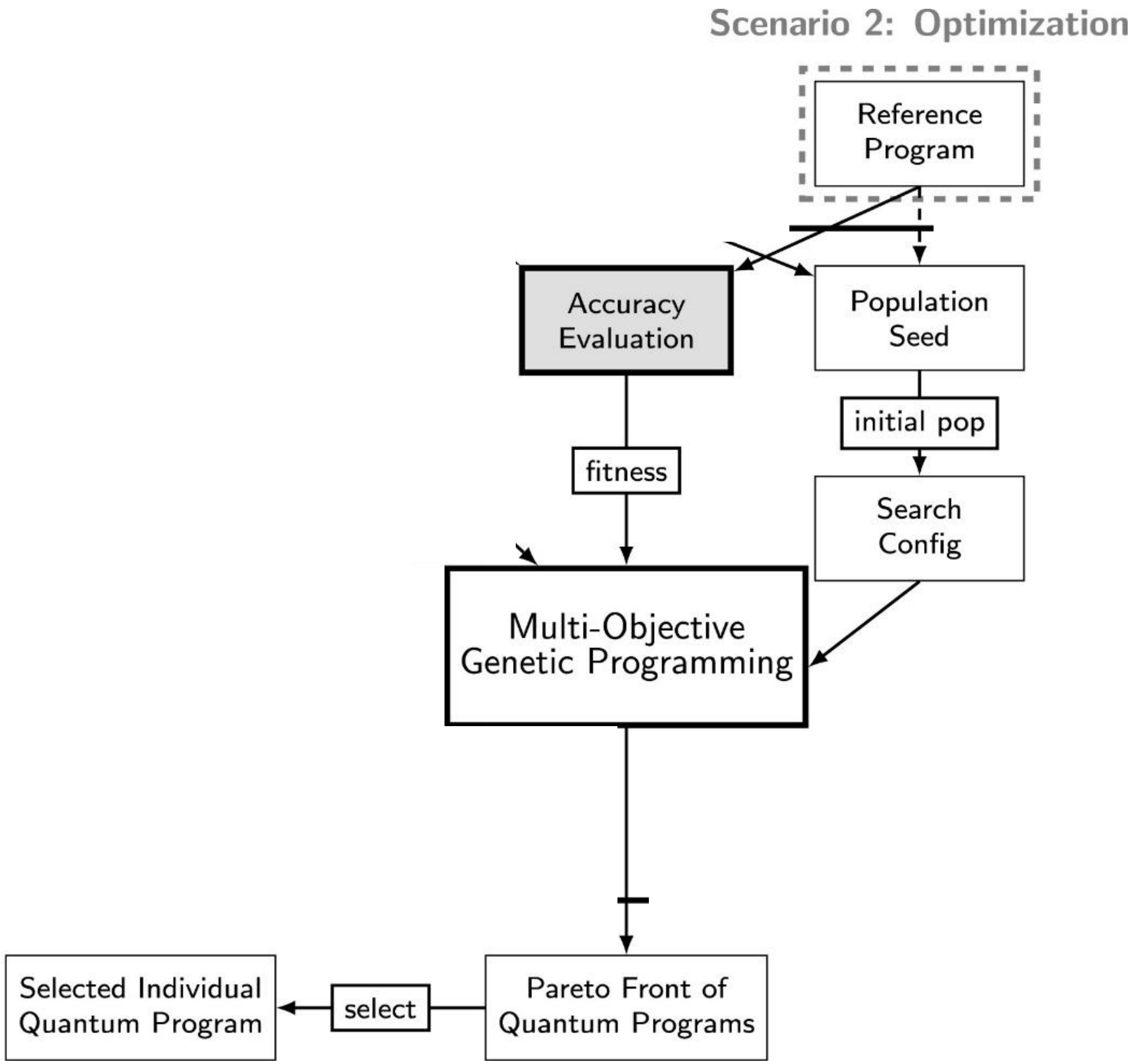
Extending to <u>arbitrary input states</u> using Bell-states [Mohseni et al. 2008]



Search Setup Overview

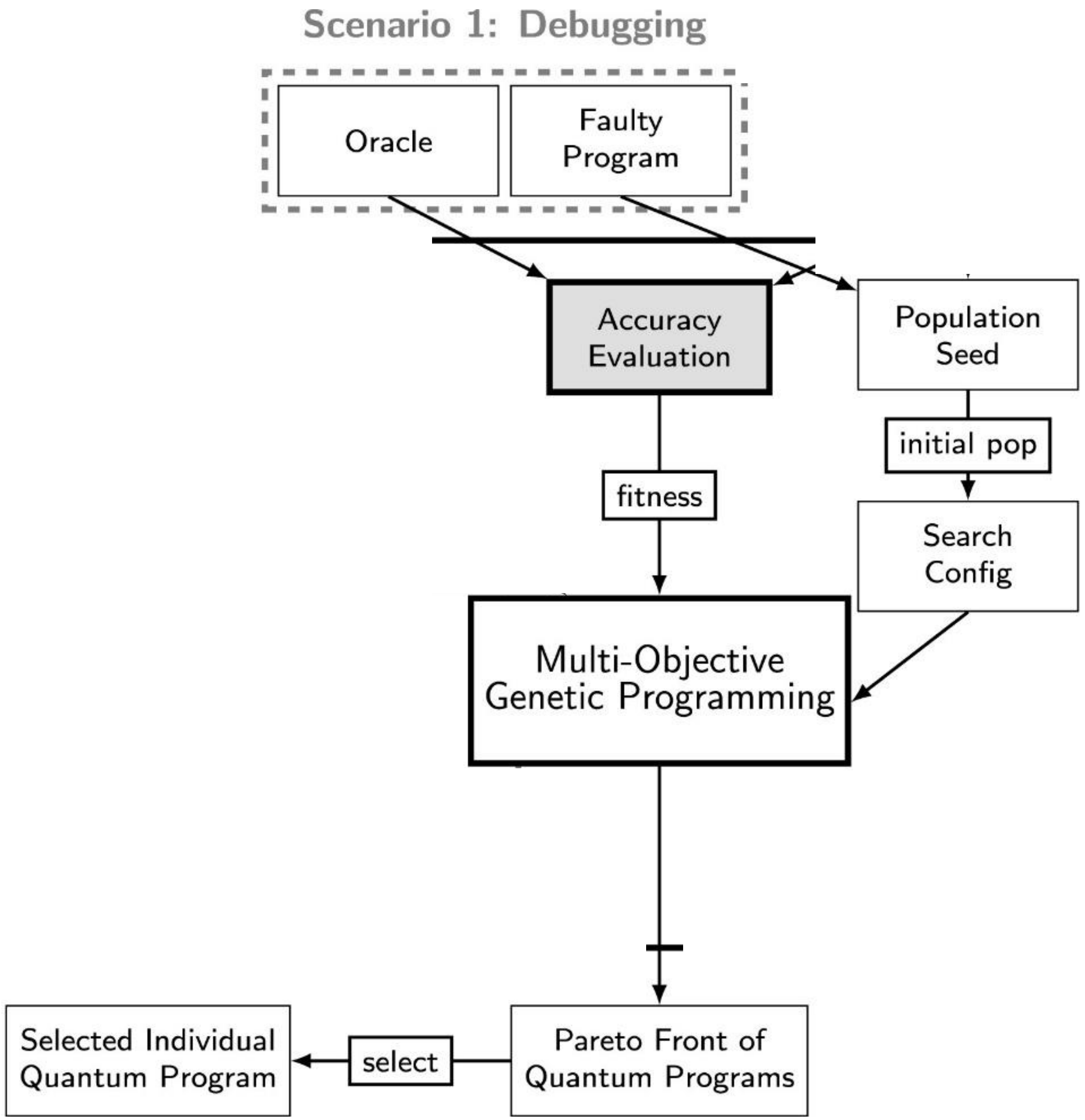


Search Setup Overview



JOHANNES KEPLER UNIVERSITY LINZ

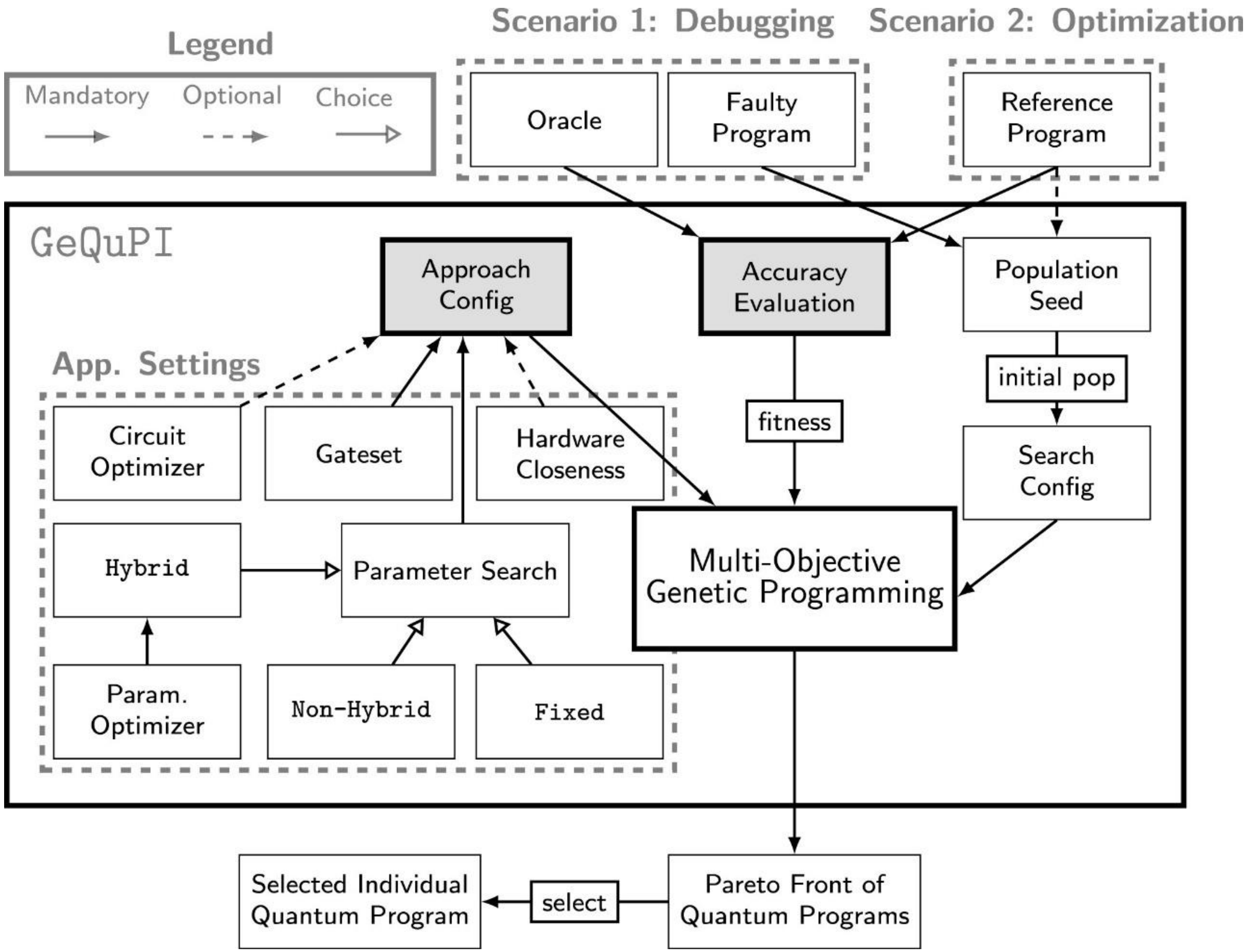
Search Setup Overview



JOHANNES KEPLER UNIVERSITY LINZ

Gemeinhardt, Klikovits, Wimmer 2025

Search Setup Overview



Evaluation

- RQ1: Debugging capabilities?
- RQ2: Optimization capabilities?

- 47 quantum programs (from literature and open source projects)
  - 9 for debugging
  - 38 for optimization
- Setup
  - Genetic Algorithm: NSGA-III
  - Hybrid: 150 gen @ 40 pop
  - Non-Hyb / Fixed: 1600 gen @ 100 pop

RQ1: Debugging capabilities?

- Hybrid can optimize
  - for specific input states
  - with Non-Hybrid & Fixed sometimes only equally good solutions are found

- Hybrid can repair arbitrary input-state programs
  - Non-Hybrid & Fixed have problems

Debugging capabilities: number of runs per category and use case. (`Hybrid` / `Non-Hybrid` / `Fixed`).

(a) RQ1.1 (Perfect Accuracy)

Input state	Problem	Optimized	Pareto Equal	Worse	Faulty
Specific	QG_8 (2 qubits)	30/30/30	0/0/0	0/0/0	0/0/0
	QSO_6 (2 qubits)	26/0/0	4/30/30	0/0/0	0/0/0
	QSO_5 (3 qubits)	30/30/30	0/0/0	0/0/0	0/0/0
	QSE_15 (4 qubits)	30/30/30	0/0/0	0/0/0	0/0/0
	QSE_3 (5 qubits)	30/30/30	0/0/0	0/0/0	0/0/0
Arbitrary	QSE2_2 (2 qubits)	30/0/0	0/0/30	0/16/0	0/14/0
	QSE2_3 (3 qubits)	0/0/0	17/0/0	8/0/8	5/30/22
	QSE2_4 (4 qubits)	0/0/0	0/0/0	7/0/0	23/30/30
	QSE2_5 (5 qubits)	0/0/0	0/0/0	6/0/0	24/30/30

RQ2 – Optimization capabilities?

- All approaches can optimize
- Hybrid performs (almost) consistently better
- Hybrid improves by 35%

- Compared to “standard approach” (Qiskit built-in optimizer):
  - optimize in significantly more cases
  - and higher on average

Optimization capabilities (Hybrid / Non-Hybrid / Fixed).

(a) RQ2.1 (Perfect Accuracy)

	Optimized	Pareto Equal	Worse	Faulty
Total	541/135/105	143/84/64	50/10/14	406/911/957
Specific	274/63/45	89/84/64	23/10/14	124/353/387
Arbitrary	267/72/60	54/0/0	27/0/0	282/558/570
2 qubits	135/101/90	40/30/30	19/0/0	16/79/90
3 qubits	248/17/11	37/20/2	9/7/14	66/316/333
4 qubits	120/6/2	55/32/30	14/3/0	141/289/298
5 qubits	38/11/2	11/2/2	8/0/0	183/227/236

RQ3: Hybrid vs. Non-Hybrid vs. Fixed?

- **Unclear** results
- Hybrid is **more diverse** (DCI, HV)
- IGD+ indicates that Hybrid is **less performant**

	PI Comparison	DCI	HV	IGD^+
All	Hybrid vs. Non-Hybrid	✓✓	✓	✗
	Hybrid vs. Fixed	✓✓✓	✓	≡
	Non-Hybrid vs. Fixed	✓✓	✓	≡
Repair	Hybrid vs. Non-Hybrid	✓✓✓	✓	≡
	Hybrid vs. Fixed	✓✓	✓	✓✓
	Non-Hybrid vs. Fixed	≡	≡	≡
Optimize	Hybrid vs. Non-Hybrid	✓✓	≡	✗
	Hybrid vs. Fixed	✓✓✓	✓✓	✗
	Non-Hybrid vs. Fixed	✓✓✓	✓	≡
Specific	Hybrid vs. Non-Hybrid	≡	≡	≡
	Hybrid vs. Fixed	✓✓✓	✓	≡
	Non-Hybrid vs. Fixed	✓✓✓	✓	≡
Arbitrary	Hybrid vs. Non-Hybrid	✓✓✓	✓	✗✗
	Hybrid vs. Fixed	✓✓✓	✓✓	≡
	Non-Hybrid vs. Fixed	✓	≡	≡

RQ4: Search configurations?

(non-exhaustive assessment)

seeding the initial population

→ **improvements** for **optimization** and **debugging**

initial population seeding

→ **most robust** configuration wrt. scaling qubits

indication of **saturation effects** wrt. population size and number of generations

RQ5 – Hardware-specificity?

Higher levels of **hardware specificity improve** the results

No hardware considerations performs worst

Using **transpiled programs** in the search process significantly improves the **share of optimized programs**

However, higher levels of **hardware-specific** considerations **increase** the **execution time**.

transpilation of programs **for the fitness values** only resembles a **viable balance** between solution quality and execution time

Conclusion & Future Work

SBSE in general is applicable for QSE

- *What is the best instantiation for QSE?*

Hybrid search seems beneficial

- *What is the best way to integrate different searchers?*

Can be applied on different abstraction levels

- *How much do we have to know about the execution?*

Seeding makes results more robust for improvement

- *How much diversity do we lose?*

Ongoing Work

Caching possibilities in search processes

- Reuse similarities within individuals to improve simulation speed

Encoding intuition in search

- E.g. „my circuit requires entanglement“

Alternative optimization approaches

- Reinforcement learning, …

Alternative encodings

- Change-based encodings with MOMoT

Thank you!

Comments? Questions? Feedback?

Looking forward to discussions and collaborations!

Tooling/data available at:

https://github.com/jku-win-se/Genetic-Programming-for-Quantum-Operator-Discovery

https://github.com/jku-win-se/QImprove