MQT QMAP：高效量子电路映射

（特邀综述论文）

Robert Wille 自动化设计教席 慕尼黑工业大学 慕尼黑，德国 哈根贝格软件能力中心有限公司 哈根贝格，奥地利
robert.wille@tum.de

Lukas Burgholzer 集成电路研究所 林茨约翰开普勒大学 林茨，奥地利
lukas.burgholzer@jku.at

arXiv:2301.11935v1 [quant-ph] 27 Jan 2023

摘要

量子计算是一项新兴技术，有望彻底改变密码学、机器学习、优化和量子模拟等领域。然而，在实际机器上实现量子算法的一项重大挑战，是确保量子电路中的量子门（即相应的操作）符合目标架构的拓扑结构，使电路能够执行，同时尽可能降低由此产生的成本（例如，额外引入的量子门数量、保真度等方面）。这就是量子电路映射问题。本文概述了 QMAP——一个隶属于慕尼黑量子工具包（MQT）的开源工具，它提供高效、自动化且易于使用的方法来解决这一问题。为此，本文首先简要回顾该问题，随后从用户和开发者视角介绍如何使用 QMAP，将量子电路高效映射到量子计算架构。QMAP 以开源形式公开发布于https://github.com/cda-tum/qmap。

CCS 概念

• **硬件** →**量子计算**；**新兴语言与编译器**；物理设计（*EDA*）；硬件描述语言与编译； • **软件及其工程** → 编译器。

关键词

q子计算，量子电路映射，编译器 器

1 引言

Quantum 计算 [1] 利用量子力学原理处理数据。不同于只能使用 0 或 1 的二进制位（比特）的经典计算，Quantum 计算使用量子比特。这使其能够处于叠加态或纠缠态。由于这些量子力学特性，量子计算机能够比经典计算机快指数级地解决某些计算问题。其中最引人注目的任务之一是确定一个大型合数 [2] 的素因数。由于许多现代加密系统都基于这一问题，若能迅速解决它，将对密码破解和网络安全产生深远影响。此外，Quantum 计算有潜力改变以下领域：机器学习 [3], 优化 [4], 和量子模拟 [5]。相较于传统计算机，能够以指数级速度更快地执行特定任务的能力，也可能有助于药物发现 [6],材料科学 [7], 和人工智能 [8]。然而，量子纠错、量子算法和软件的开发，以及大规模量子计算机的物理实现，仍然是该领域面临的重大挑战。但 Quantum 计算研究正在快速推进，量子计算机的能力也在持续提升。

通常，量子算法会表示为量子电路，即由作用于量子比特的一系列量子操作（或量子门）组成的序列。算法必须先经过量子电路编译器编译，才能在真实的量子处理单元 （QPU）上执行。此类编译器通常会首先使用设备无关优化来简化原始电路描述 [9]–[14]。这些优化包括取消或重新排列量子门，以及使用高级综合例程等技术。完成这些初步优化后，便会生成 QPU 专用电路。具体做法是将所有非原生量子门转换为一组可在设备上直接执行的原生量子门 [15]–[18]。此外，有些 QPU 的量子比特之间仅具有有限的连通性，因此需要执行映射步骤，将电路中的逻辑（或虚拟）量子比特映射到设备的物理量子比特上，以确保整个电路都满足连通性约束。

本研究聚焦于量子电路映射这一环节。事实上，要确保量子电路中的所有量子门都符合指定架构的拓扑结构，同时又将由此产生的成本（例如，新增量子门的数量、保真度等方面的成本）控制在较低水平，极其困难。实践中，大多数量子电路都无法轻易映射到现有的量子架构上。因此，映射过程中会引入额外的量子门（称为 SWAP 门），以确保电路符合架构的连通性约束。可通过多种指标评估映射结果的质量，例如所得电路的（双量子比特）量子门数量、电路深度或预期保真度。为清晰起见，以下将重点关注额外量子门的数量。尽可能降低这一数值对所得电路的性能至关重要，因为它往往决定了电路能否产生预期结果，还是只会产生完全随机的输出。

本文介绍如何使用开源工具 QMAP 高效解决量子电路映射问题，该工具是

慕尼黑量子工具包 （MQT）的一部分。MQT 是一套量子计算软件工具，主要致力于为解决量子计算应用设计中的复杂问题，提供高效、自动化且易于使用的方法。为此，我们将简要概述各个方面，具体包括

(1) 对量子电路映射问题的高层次描述，以及 MQT QMAP 工具如何帮助高效解决该问题（见第 3 节）；(2) 从用户视角介绍如何利用 QMAP 将量子电路高效映射到量子计算架构（见第 4 节）；以及 (3) 从开发者视角介绍如何在 QMAP 工具中开发新方法或扩展现有方法（见第 5 节）。

2 背景

在深入探讨量子电路映射和 QMAP 之前，本节将简要回顾一般意义上的量子计算基础，尤其是当前的量子计算架构。需要指出的是，本综述论文的篇幅范围无法对广泛的量子计算领域进行全面回顾。因此，我们建议任何希望进入该领域的人参考更深入的基础知识介绍，例如 [1]。

2.1 量子计算

与使用只能为零或一的比特表示数据的经典计算不同，量子计算使用量子比特，即可以处于状态叠加态的量子比特，这意味着它们可以处于 0 和 1 状态的线性组合中。更具体地说，单个量子比特的状态 $|\psi\rangle$ 由两个复数值振幅 α_0 和 α_1 描述，使得

$$|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle \text{ with } |\alpha_0|^2 + |\alpha_1|^2 = 1.$$

量子比特测量（从量子计算机中提取信息的唯一方式）会使其状态以一定概率坍缩到基态 $|0\rangle$ 或 $|1\rangle$（对应的概率分别为 $|\alpha_0|^2$ 和 $|\alpha_1|^2$）。

与经典计算一样，计算需要使用多个量子比特。为此，一个由 n 个量子比特组成的系统，其状态由 2^n 个复数振幅 α_i 描述；每个振幅对应一个长度为 n 的可能比特串，即：

$$|\psi\rangle = \sum_{i\in\{0,1\}^n} \alpha_i|i\rangle \text{ with } \sum_{i\in\{0,1\}^n} |\alpha_i|^2 = 1.$$

同样，$|\alpha_i|^2$ 描述了测量量子比特时获得基态 $|i\rangle$ 的概率。

量子比特的状态通过量子操作（也称为量子门）来改变。量子门的例子包括经典门的对应形式，例如可翻转量子比特状态的 X 门；也包括量子特有的量子门，例如哈达玛门，它可以将量子比特从（经典）基态置于（量子）叠加态。这些量子门用于构建量子电路，即量子计算中与经典电路相对应的形式。目前，量子电路是描述量子计算的主要方式。

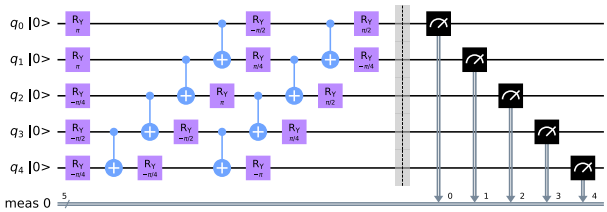


图1：五量子比特变分量子本征求解器量子电路。该量子电路由多串双量子比特 CNOT 门组成，并交错排列着单量子比特旋转门。

示例 1。图 *1* 展示了一个作用于五个量子比特的量子电路。为此，量子比特以水平线表示，量子门则以放置在线上的方框和符号表示。量子门从左向右依次施加。该量子电路由单量子比特旋转门（以紫色方框表示）和双量子比特 *CNOT* 门（以蓝色表示）组成。它展示了一种流行的近期量子算法（称为 变分量子本征求解器，简称 *VQE [19]*）在一个小型优化问题上的应用。

2.2 量子计算架构

目前，已有多种技术应用于量子计算，或正处于研发阶段。其中最值得关注的技术包括超导量子比特 [20], 囚禁离子 [21], 中性原子 [22], 和光子量子比特 [23]。这些技术各有优缺点。例如，超导量子比特易于控制和操纵，但必须在极低温度下运行。另一方面，囚禁离子可以在高得多的温度下运行，且相干时间较长，但难以扩展到大量量子比特。需要注意的是，大多数技术仍处于研究阶段，目前尚不清楚哪种技术最适用于大规模通用量子计算。本文余下部分将聚焦于基于超导量子比特的量子计算机，因为量子电路映射问题与这类计算机最为相关。

示例 2. *IBM*、*Rigetti* 和 *Google* 等主要产业参与者，以及 牛津量子计算 （*OQC*）等学术机构，目前都提供基于超导量子比特的量子计算机访问服务。以 *IBM* 为例，截至 *2022*年年底，其提供的设备规模从 5 量子比特一直到 433 量子比特，并计划根据其路线图，在 *2025*年 年前推出一套拥有超过 4000 量子比特的系统（*ibm.com/quantum/roadmap*）。

量子计算架构只提供一组特定的原生量子门，通常由若干单量子比特操作和一种双量子比特纠缠操作组成——实际上构成了一个通用门集，借助它可以实现任何量子计算。这可以与经典处理器只能支持特定指令集架构 （ISA）相比较。正如需要编译器将高级代码转换为低级指令一样，高级量子电路也需要先分解为设备的原生门集，才能在该设备上执行。

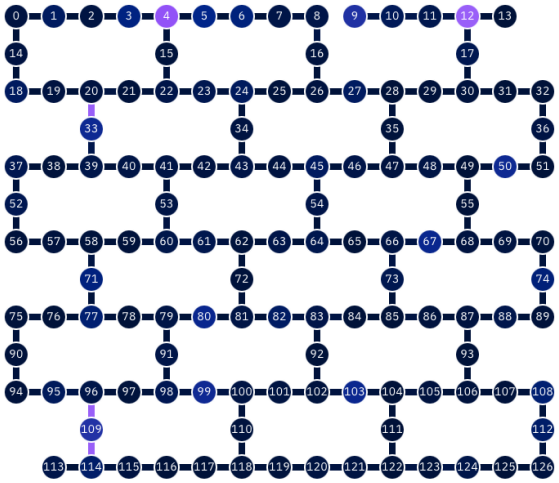


图 2：127量子比特*IBM*_华盛顿设备的耦合图

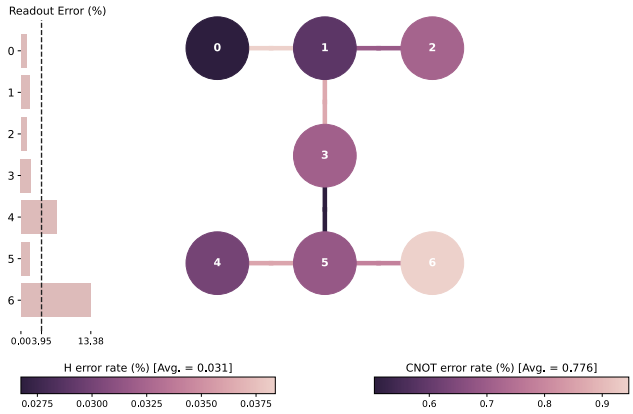


图3：带有设备校准数据的七量子比特 *ibm_oslo* 架构。圆圈表示各个量子比特。量子比特之间的连接表示可对其应用双量子比特量子门。量子比特（边）的颜色表示对该量子比特（边）执行的单量子比特（双量子比特）操作的错误率——颜色越深表示错误率越低。

示例 3. 再次以 *IBM* 提供的量子计算机为例，它们提供的原生门集由三个单量子比特门（X、$\sqrt{X}$ 和 $R_z(\theta)$）以及 *CNOT* 门组成。虽然看起来门集规模不大，但它已经构成了量子计算的通用门集，也就是说，利用其中的量子门可以实现任何可设想的量子计算。

此外，量子计算机（例如基于超导量子比特的计算机）的量子比特之间通常只有有限的连通性，也就是说，设备上并非任意量子比特之间都可以施加双量子比特门。量子比特之间可能发生相互作用的方式，通常用耦合图（或耦合图）来描述。一般来说，这是一个有向图，用于描述哪些量子比特对可以施加特定的双量子比特门。在大多数情况下，该图实际上是无向的，这意味着两个量子比特之间量子门的方向性并不重要。要在这类设备上执行量子电路，电路必须遵守设备耦合图所施加的连通性约束。

示例 4。图 2展示了127量子比特IBM_华盛顿设备的耦合图。与所有较新的 *IBM* 架构一样，它由12量子比特的模块组成，并以砖块状布局拼接在一起。请注意，整个架构的连通性十分稀疏，每个量子比特最多只有三个相邻量子比特。

3 量子电路映射与 MQT QMAP

如上一节所述，许多量子计算架构会限制可执行双量子比特操作的量子比特对，通常通过设备的耦合图来描述。要在此类架构上执行通用量子电路（其量子比特之间具有任意相互作用），就需要对电路进行映射。这包括量子比特分配，即在初始布局中将电路的（逻辑）量子比特分配给设备的（物理）量子比特；还包括路由，即向原始电路中添加 SWAP 门，使其符合目标设备的耦合图。

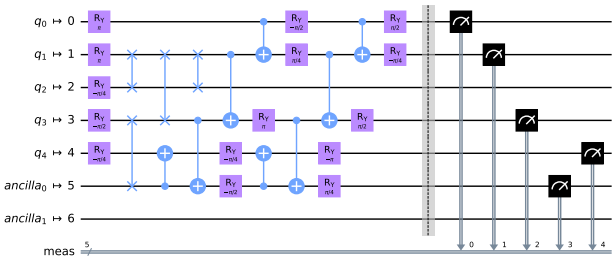


图 4：将图 1 中的量子电路映射到*ibm_oslo* 架构中，如图 3 所示。要使该量子电路可执行，需要四个 SWAP 门和一个额外的量子比特。

示例 5. 考虑示例 *1* 中的量子电路（如图 *1*所示），并假设需要将其映射到七量子比特的 ibm_oslo架构上，该架构的耦合图（以及一些校准信息）如图 *3*所示。由于该量子电路包含作用于设备耦合图中未连接的量子比特上的量子门，因此无法直接在该架构上运行。若直接插入 *SWAP* 门，动态置换逻辑量子比特到物理量子比特的映射，可能会得到如图 *4*所示的编译后电路。在映射过程中，为满足设备架构的连通性约束，共引入了四个 *SWAP* 门。此外，为实现预期功能，该量子电路还使用了一个额外的量子比特。由于每增加一个量子门（和量子比特）都会提高出错概率，因此对于如此小的量子电路而言，这种开销非常大。

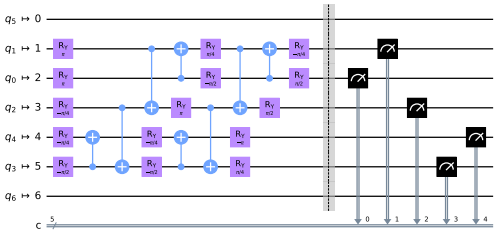


图5：将 图 1 中的量子电路映射到 *ibm_oslo*架构（如 图 3 所示），使用 QMAP 的精确映射器。无需 SWAP 门或额外的量子比特，即可使量子电路可执行。

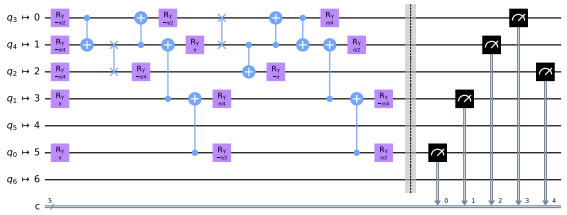


图6：将 图 1 中的量子电路映射到 *IBM*_奥斯陆架构的映射结果如 图 3 所示，该映射使用了 QMAP 的启发式映射器。映射使用两个 SWAP 门和一个额外的 CNOT 门。

在量子电路映射过程中，尽可能减少额外引入的量子门数量，是确保量子电路成功执行的关键。确定量子电路的最优映射是一个NP难问题 [24]。

撰写本文时，开源量子电路映射工具 QMAP 是慕尼黑量子工具包 (MQT) 的一部分，它提供了两种专门用于解决该问题的技术：

(1) 精确映射方法（基于 [25], [26]），可保证获得（量子门数最优的）解。(2) 启发式映射方法（基于 [27]–[29]），能够以可扩展的方式确定高效的映射解。

QMAP 中实现的精确映射器使用最少数量的 SWAP 门映射量子电路。为此，它将映射任务编码为最大可满足性问题，随后使用 SMT 求解器 Z3 [30] 求解。由于映射任务是 NP 难问题，在大多数场景下，该方法的可扩展性仅限于大约 8 个量子比特和约 1000 个量子门。

示例6。 应用于图 *1*所示的量子电路时，*QMAP* 的精确映射器会生成图 *5*所示的量子电路。在这种情况下，无需使用任何 *SWAP* 量子门，也不需要额外的量子比特即可完成量子电路映射。因此，生成的量子电路的预期性能将显著提升。获得这一性能的代价是映射运行时间约为数秒。

QMAP 中实现的启发式映射器使用 *A**-搜索 [31]高效遍历映射问题庞大的搜索空间。它以牺牲最优性为代价来缩短运行时间。因此，它能够可靠地为多达数百个量子比特、包含数十万个量子门的量子电路确定合适的映射。

示例7. 将其用于 图 *1*所示的量子电路时，*QMAP* 的启发式映射器会生成 图*6*所示的量子电路。虽然该解已不再是最优解，但仍只需两个 *SWAP* 门和额外的一个 *CNOT* 门，而朴素映射则需要四个 *SWAP* 门。此外，即使对于如此小的示例，启发式映射器也比精确映射器快几个数量级，耗时不到一毫秒。

4 用户视角：使用 QMAP

任何人都可以使用 QMAP，快速有效地将量子电路映射到量子计算系统。本节简要介绍 MQT QMAP 工具的安装过程和初步使用方法，并从用户的角度展示 QMAP。

QMAP 主要以 C++ 库的形式开发，但为了让该工具尽可能易于使用并具有良好的兼容性，它还提供了一个易于使用的 Python 接口。安装该工具只需

```
$ python3 -m venv .venv
$ source venv/bin/activate
(venv) $ pip install -U pip setuptools wheel
(venv) $ pip install mqt.qmap
```

在大多数实际情况下（使用 64 位 Linux、macOS（包括 Apple 芯片）和 Windows 时），无需编译，只需下载并安装适用于相应平台的预编译 wheel 包即可。

可以通过多种不同方式指定要映射特定量子电路的架构。最简单的方式是直接使用 IBM 的 Qiskit [32] 后端，也可以直接根据耦合图创建（任意）架构。

示例 8. 此前讨论的 ibm_oslo 设备（如图 *3*所示）可以通过以下方式实例化：

```
来自qiskit.providers.fake_提供程序 导入 FakeOslo

架构 = FakeOslo()
```

同一架构也可以不依赖 *Qiskit*，通过以下方式提供：

```
从mqt 导入 qmap

架构 = qmap.Architecture(7,{
  (0, 1), (1, 0),
  (1, 2), (2, 1),
  (1, 3), (3, 1),
  (2, 3), (3, 2),
  (3, 5), (5, 3),
  (4, 5), (5, 4),
  (5, 6), (6, 5)}
)
```

这样便可以定义任意所需的量子电路，并将其映射到架构上。再次强调，使用 Python 中的 Qiskit 是最简单的方法。也可以使用 OpenQASM [33], 等其他文件格式来定义量子电路。

示例 9。 图中所示量子电路 图 *1* 源自 *MQT Bench* 库 *[34]* 可通过以下方式安装

```
(venv) $ pip 安装 mqt.bench
```

随后，可通过以下方式将该量子电路获取为 *Qiskit* 量子电路

```
来自mqt 导入基准测试

量子电路 = 基准测试.获取_基准测试(基准测试_名称='变分量子本征求解器',
    量子电路_大小 =5, 级别='独立')
```

请注意，每次调用时获得的旋转门具体参数可能会有所不同。

定义要映射到的架构以及要进行映射的量子电路 qc 后，实际上使用 QMAP 执行映射只需一行 Python 代码

```
qc_m, res = qmap.编译(qc, 架构, 方法="精确")
```

对于精确映射器，或

```
qc_m, 资源 = qmap。编译(qc, 架构, 方法="启发式")
```

用于启发式映射器。两者都会返回映射后的量子电路（作为 IBM Qiskit QuantumCircuit 对象）以及一组结果。例如，通过这些结果可以方便地打印所需的 SWAP 操作数量和映射例程的运行时间。

```
print("Additional SWAPs: %d" % res.output.swaps)
print("Runtime:          %f" % res.time)
```

当然，前述演示仅简要介绍了 QMAP 的功能。`compile` 方法包含许多可选参数，可用于配置各个映射器。不过，QMAP 提供了全面的文档，可通过以下网址访问：https://mqtqmap.readthedocs.io 其中更详尽地介绍了所有可用选项和技术。

5 开发者视角：改进 QMAP

QMAP 是一款功能强大的工具，但仍有许多改进空间。QMAP 的许多方面都是从纯学术角度出发设计的，关注范围相当狭窄，只聚焦于非常具体的主题。本节首先简要介绍如何为 QMAP 配置开发环境，随后讨论可能的改进方向。具体而言，我们将展示

- 如何通过定义更复杂的初始布局算法以及支持更多种类的量子门来改进启发式映射器，以及
- 如何调整精确映射器的 SAT 编码及其构造，以提升映射器的整体性能。

这从开发者视角简要介绍了 QMAP。

5.1 设置开发环境

QMAP 主要以 C++ 库的形式开发，基于 MQT 量子功能表示 (QFR) 库（https:// github.com/cda-tum/qfr）。为了让该工具尽可能易于使用并兼容各种环境，它提供了易于使用的 Python 绑定。因此，为 QMAP 库开发功能主要包括扩展核心 C++ 库，然后将新添加的功能开放给 Python 使用。

首先，需要通过以下命令从 GitHub 克隆项目：

```
git clone --recursive https:// github.com/cda -tum/qmap
```

构建该项目需要支持 C++ 的编译器++17，以及最低版本为 3.19 的 CMake。此外，构建精确映射器需要安装最低版本为 4.8.15 的 SMT 求解器 Z3 [30] 3（详见 https://github.com/Z3Prover/z3 ）。与其他基于 CMake 的项目一样，构建 C++ 库分为两个步骤

```
[.../ qmap ] $ cmake -S . -B build -DBUILD_QMAP_TESTS=ON[.../qmap ] $

```

要在本地构建并安装 Python 软件包，请先按照上一节所述创建虚拟环境，然后运行

```
[.../qmap] (venv) $ pip install -e ".[开发]"
```

该库提供了许多便捷功能，例如用于测试库功能的单元测试、用于确保代码质量的代码检查和代码格式化工具，以及在 GitHub 上全面配置的 CI 工作流，以确保项目能够在所有主流操作系统上构建并运行。如需了解更多信息，请访问 mqtqmap.readthedocs.io/en/latest/DevelopmentGuide.html。

设置好开发环境后，可以轻松扩展并改进 QMAP。下面将分别介绍启发式映射器和精确映射器的示例。

5.2 改进启发式映射器

改进启发式映射器的一种方法是改进初始映射。与大多数量子电路映射器一样，启发式 QMAP 映射器将映射任务分为两步：(1) 确定初始布局，将逻辑量子比特放置到设备的物理量子比特上；(2) 根据选定的初始布局执行实际路由。截至本文撰写时，QMAP 支持三种不同类型的初始布局：

(1) 简单的恒等布局，其中逻辑量子比特 q_i 被分配给物理量子比特 Q_i，对所有从 0 到 $n-1$ 的 i 均如此。(2) 静态布局：查找量子电路第一层中共享一个量子门的量子比特，并将它们映射到架构中任意一对空闲且相连的量子比特上；其余量子比特则按索引顺序映射。(3) 动态布局：在映射过程中遇到量子比特时，再按需贪心地为其分配位置。

正如相关文献中的评估所证实的那样 [27]–[29]，这些初始布局技术与 A^* 搜索相结合，已经能够实现相当高效的映射结果。同时，众所周知，初始映射会显著影响可实现的路由性能；例如参见 [35]。因此，更复杂的初始映射可能会带来更好的整体结果——仍有进一步改进的空间。如今，许多用于映射量子电路的工具（例如 [36]）会将量子电路的量子门交互图与目标架构的耦合图进行匹配，以确定初始映射。还可以结合设备校准信息，以考虑量子比特质量和量子门质量的差异。实施此类策略可能是进一步提升 QMAP 效率的一种直接方法。

另一个方向可能涉及所支持的门集。由于历史原因，QMAP 假设量子电路中的双量子比特门只有 CNOT门。在只有 IBM 提供对其设备的公开访问权限、且 Qiskit 是唯一主要的开源量子软件开发工具包的时期，这一假设是合理的。然而，随着时间推移，出现了许多其他（且各不相同的）硬件提供商和量子计算机——每种设备都有自己的原生门集。例如，Rigetti 量子计算机采用 CZ门，而不是 CNOT 门。因此，使 QMAP 适配更新类型的设备和/或支持更多样化的门集，是改进 QMAP 的另一种直接方法。

5.3 改进精确映射器

精确映射器同样存在改进空间。如前所述，精确映射器将映射任务编码为最大可满足性问题，使其能够以最少的 SWAP 门输出结果。然而，结果的质量是有代价的，因为映射问题的搜索空间按 $N! * k$ 的规模增长，其中 N 是架构中的量子比特数量，k 是量子电路中的（双量子比特）量子门数量。因此，该技术目前仅适用于量子比特少于十个的架构，以及包含几百到一千个量子门的量子电路。

人们已采取多种措施来缩减量子电路映射问题中的搜索空间。[26] 中已经证明，在映射中每一层之前，只需考虑足够数量的排列，使任意两个量子比特能够彼此相邻放置即可。这将排列数量从 $N!$ 大幅减少。尽管如此，[26] 中描述的概念仍以相当直接的方式实现。在这方面还可以进一步改进，例如通过开发专用的凯莱图数据结构及相应的操作算法，简化必须考虑的排列的计算过程。

此外，人们也尝试限制映射过程中需要考虑的量子比特数量。若不加以限制，例如，将一个 3 量子比特量子电路映射到一个拥有 100 个量子比特的架构上将是不可能的。[25]，提出了最激进的方法：它考虑所有包含与待映射量子电路相同数量量子比特的可行子架构，而不是整个架构。尽管这种方法最大限度地减少了需要考虑的量子比特数量（量子电路不可能映射到量子比特数量少于自身的架构上），但也存在一些缺点。首先，现有实现相对简单，因为它没有考虑子架构同构，即多个子架构在结构上可能完全相同的情况。相反，它只是通过蛮力法遍历所有可能的组合。此外，[37] 已证明，由此得到的量子电路无法保证最优性，因为量子比特数量更多的子架构可能支持更好的映射。幸运的是，[37]提出了用于确定映射过程中应考虑的最优或近优子架构集合的方法。这些方法尚未直接集成到精确映射器中。

6 结论

这篇综述论文介绍了 MQT QMAP 工具，该工具使研究人员和开发人员能够将量子电路有效地映射到真实的量子计算架构上。我们简要回顾了量子电路映射问题，并展示了如何使用 QMAP 高效、自动地解决这一问题。此外，我们还概述了如何为 QMAP 进行开发，并介绍了未来改进的潜在方向。我们希望借此引起更多人对这一引人入胜且迅速发展的研究领域的兴趣。如需深入探讨相关方法和挑战，可参阅前文提及的参考文献。

致谢

我们感谢所有为 QMAP 工具作出贡献的人。特别感谢 Stefan Hillmich、Tom Peham、Sarah Schneider 和 Alwin Zulehner 对本文所述工作作出的具体贡献。

本研究获得了欧洲研究理事会（ERC）通过欧盟地平线 2020研究与创新计划（资助协议编号：101001318）提供的资金支持，是慕尼黑量子谷的一部分；该项目由巴伐利亚州政府通过 Hightech Agenda BayernPlus 提供资金支持。此外，本研究还通过 QuaST 项目获得 BMWK 的支持，该项目依据德国联邦议院的决议开展。

参考文献

[1] Michael A. Nielsen 和 Isaac L. Chuang，《量子计算与量子信息》。剑桥大学出版社，2010年。[2] Peter W. Shor，“量子计算机上用于素因数分解和离散对数的多项式时间算法”，《SIAM计算杂志》，1997年。[3] Jacob Biamonte 等，“量子机器学习”，《自然》，第549卷第7671期，第195–202页，2017年。arXiv：1611.09347。[4] Stuart Harwood 等，“在量子计算机上制定和求解路由问题”，《IEEE量子工程汇刊》，第2卷，第1–17页，2021年。doi：10.1109/TQE.2021.3049230。[5] Katherine L. Brown、William J. Munro 和 Vivien M. Kendon，“使用量子计算机进行量子模拟”，《熵》，第12卷第11期，第2268–2307页，2010年。doi：10.3390/e12112268。[6] Y. Cao、J. Romero 和 A. Aspuru-Guzik，“量子计算在药物发现中的潜力”，《IBM研发杂志》，第62卷第6期，6:1–6:20，2018年。doi：10.1147/JRD.2018.2888987。[7] Benchen Huang、Marco Govoni 和 Giulia Galli，“在量子计算机上模拟自旋缺陷的电子结构”，《PRX Quantum》，第3卷第1期，第010339页，2022年。doi：10.1103/PRXQuantum.3.010339。[8] Vedran Dunjko 和 Hans J Briegel，“量子领域中的机器学习与人工智能：近期进展综述”，《物理学报告》，第81卷，074 001，2018年7月，10 1088 1361 6633 406 no. p. 。doi：. / - /aab 。

[9] Wakaki Hattori 和 Shigeru Yamashita，“通过改变二维最近邻架构的门顺序进行量子电路优化”，载于《可逆计算国际会议》，第11106卷，2018年，第228–243页。doi：10.1007/978-3-319-99498-7 16。[10] Zahra Sasanian 和 D. Michael Miller，“可逆与量子电路优化：一种函数式方法”，载于《可逆计算国际会议》，David Hutchison 等编，第7581卷，2013年，第112–124页。doi：10.1007/978-3-642-36315-3_9。[11] Toshinari Itoko、Rudy Raymond、Takashi Imamichi 和 Atsushi Matsuo，“ 利用门变换与对易优化量子电路映射 ”，《Integration》，第70卷，第43–50页，2020年。doi：10.1016/j.vlsi.2019.10.004。[12] D. Maslov、G.W. Dueck、D.M. Miller 和 C. Negrevergne，“量子电路简化与层压缩”，IEEE集成电路与系统计算机辅助设计汇刊，第27卷第3期，第436–444页，2008年。doi：10.1109/TCAD.2007.911334。[13] Sarah Schneider、Lukas Burgholzer 和 Robert Wille，“用于最优Clifford电路综合的SAT 编码”，载于亚洲及南太平洋设计自动化会议，2023年。[14] Ed Younis 和 Costin Iancu，“通过参数化电路实例化实现量子电路优化与转译”，载于qce，IEEE，2022年，第465–475页。doi：10.1109/QCE53715.2022。00068。[15] G. Vidal 和 C. M. Dawson，“使用三个受控非门实现双量子比特变换的通用量子电路”，《物理评论A》，第69卷第1期，第010 301页，2004年。doi：10.1103/PhysRevA.69.010301。[16] Dmitri Maslov，“使用相对相位托福利门的优势及其在多重控制托福利门优化中的应用”，《物理评论A》，第93卷第2期，第022 311页，2016年。doi：10.1103/PhysRevA.93.022311。[17] D. Michael Miller、Robert Wille 和 Zahra Sasanian，“多重控制托福利门的基本量子门实现”，载于多值逻辑国际研讨会，2011年。doi：10.1109/ISMVL.2011.54。[18] Marc G. Davis 等，“迈向最优的拓扑感知量子电路综合”，载于qce，IEEE，2020年，第223–234页。doi：10.1109/QCE49297.2020.00036。[19] M. Cerezo 等，《变分量子算法》，2020年。arXiv：2012.09265。[20] P. Krantz 等，“超导量子比特工程师指南”，《应用物理评论》，第6卷第2期，第021 318页，2019年。doi：10.1063/1.5089550。[21] H Haffner、C Roos 和 R Blatt，“使用囚禁离子进行量子计算”，《物理学报告》，第469卷，第203、155–10 1016、2008 09 003号，第4页。doi：. /j.physrep. . . 。[22] Loïc Henriet 等，“使用中性原子进行量子计算”，《Quantum》，第4卷，第327页，2020年。doi：10.22331/q-2020-09-21- 327。[23] Jeremy L. O’Brien、Akira Furusawa 和 Jelena Vučković，“光子量子技术”，《自然·光子学》，第3卷第12期，第687–695页，2009年。doi：10.1038/nphoton.2009.229。[24] A. Botea、A. Kishimoto 和 Radu Marinescu，“关于量子电路编译的复杂性”，载于国际组合搜索研讨会，2018年。

[25] Robert Wille、Lukas Burgholzer 和 Alwin Zulehner，《使用最少数量的 SWAP 和 H 操作将量子电路映射到 IBM QX 架构》，载于设计自动化会议，2019年。[26] Lukas Burgholzer、Sarah Schneider 和 Robert Wille，《限制最优量子电路映射的搜索空间》，载于亚洲及南太平洋设计自动化会议，2022年。[27] Alwin Zulehner、Alexandru Paler 和 Robert Wille，《将量子电路映射到 IBM QX 架构的高效方法》，《IEEE集成电路与系统计算机辅助设计汇刊》，2019年。[28] Stefan Hillmich、Alwin Zulehner 和 Robert Wille，《在量子电路映射中利用量子隐形传态》，载于欧洲设计、自动化与测试会议，ACM，2021年，第 792–797 页。doi：10.1145/3394885.3431604。[29] Alwin Zulehner 和 Robert Wille，《将 SU(4)量子电路编译到 IBM QX 架构》，载于亚洲及南太平洋设计自动化会议，2019年，第 185–190 页。doi：10.1145/3287624.3287704。[30] Leonardo de Moura 和 Nikolaj Bjørner，《Z3：一种高效的 SMT 求解器》，载于《工具、算法与系统构造分析》，Springer，2008年，第 337–340 页。[31] Peter Hart、Nils Nilsson 和 Bertram Raphael，《最小代价路径启发式确定的形式化基础》，《IEEE系统科学与控制论汇刊》，第 4 卷，107，1968 2，100 10 1109 1968 300136，第 – 页。doi：. / TSSC. . .[32] Matthew Treinish 等，Qiskit/qiskit，Zenodo，2022年。doi：10. 5281/ZENODO.2573505。[33] Andrew W. Cross 等，《OpenQASM 3：一种更广泛、更深入的量子汇编语言》，2021年。arXiv：2104.14722。[34] Nils Quetschlich、Lukas Burgholzer 和 Robert Wille，《MQT Bench：量子计算基准测试软件与设计自动化工具》，2022年。arXiv：2204.13719。[35] Alexandru Paler，《初始量子比特布局对 NISQ 电路编译的影响》，载于《量子技术优化问题》，第 11413 卷，2019年，第 207–217 页。doi：10.1007/978-3-030-14082-3 18。[36] Seyon Sivarajah 等，《T|ket〉：一种适用于 NISQ 设备的可重定向编译器》，《量子科学与技术》，第 6 卷，014003，2021 1，10 1088 2058 9565 8 92，第 p. 页。doi：. / - /ab e .[37] Tom Peham、Lukas Burgholzer 和 Robert Wille，《量子电路映射的最优子架构》，2022年。arXiv：2210.09321 [quant-ph]。