وزارت علوم، تحقیقات و فناوری
دانشگاه تحصیلات تکمیلی علوم پایه
گاوازنگ، زنجان

محک شبیه‌سازهای مدار کوانتومی

پایان‌نامه‌ی کارشناسی ارشد

احمد محمودیان درویشانی

اساتید راهنما: **دکتر علی ابن نصیر**

دکتر مهدی وثیقی

استاد مشاور: **دکتر منصور داوودی منفرد**


بسم الله الرحمن الرحیم

تقدیم به خانواده،

به‌خصوص برادر عزیزم

که لحظه به لحظه، یار و همراه من بود.

چکیده

عدم دسترسی به رایانه‌های کوانتومی منجر به توسعهٔ شبیه‌سازهای آن‌ها شده است. نوپایی این توسعه و تفاوت در مفاهیم بنیادی محاسبات کوانتومی، پیچیدگی زیادی در این فرایند به وجود آورده است. محک شبیه‌سازهای مدار کوانتومی برای شناخت بهتر این پیچیدگی‌ها، هدف پژوهش ما بوده است. این محک با اجرای الگوریتم‌های شناخته‌شدهٔ کوانتومی نظیر Shor و Grover در دو سامانهٔ معروف Qiskit و Cirq و اندازه‌گیری زمان اجرا و میزان مصرف حافظهٔ آن‌ها، به‌طوری‌که این اندازه‌گیری‌ها منصفانه باشد، صورت‌گرفته است. شبیه‌سازهایی که در سامانهٔ Qiskit وجود دارند، به خصوص شبیه‌سازهایی که از روش نمودارهای تصمیم برای شبیه‌ساز استفاده کرده‌اند، زمان اجرای کمتری را به ثبت رسانده‌اند. از نگاه کاربری که می‌خواهد شبیه‌سازی‌ها را با رایانهٔ شخصی انجام دهد، نتایج یافت‌شده حاکی از آن است که بستگی به نوع الگوریتم باید شبیه‌سازهای متفاوت برای اجرای آن‌ها انتخاب شود تا نتایج مطلوب‌تری حاصل شود. به‌علاوه، با رصد این محک‌ها می‌توان برای بهبود ناتوانی‌های شبیه‌سازها گام برداشت.

This part should be checked.

واژه‌های کلیدی: *محک، شبیه‌سازی، رایانش کوانتومی*

فهرست مطالب

فهرست تصاویر

پیش‌گفتار

این پایان‌نامه به چهار قسمت اصلی تقسیم شده است. در قسمت اول مقدمه‌ای از مسئلهٔ پژوهش و توضیح و بسط آن، ارائه می‌شود. در قسمت دوم، تلاشی برای آشنایی خواننده با ادبیات این حوزه صورت خواهد گرفت. قسمت سوم به بیان مطالعات پیشین و پس از آن روش‌ها و نحوهٔ محک شبیه‌سازها می‌پردازد. قسمت چهارم، توضیحات تکمیلی برای نحوهٔ پیاده‌سازی الگوریتم‌های کوانتومی انتخاب‌شده برای محک را در بر می‌گیرد و مجموعهٔ کاملی از نتایج حاصل‌شده را به تصویر می‌کشد تا به نتیجه‌گیری‌های انجام‌شده صورت ببخشد. امید است این پژوهش برای تمامی خوانندگان و پژوهشگران سودمند واقع شود.

فصل اول

مقدمه

در این فصل، تحت یک مقدمه، به بیان مسئله و ضرورت وجود محک شبیه‌سازها و پس از آن اشاره‌ای به شکاف‌های پژوهشی، روش پیشنهادی برای برطرف‌کردن شکاف و در نهایت به دستاوردها و نتایج می‌پردازیم.

بیان مسئله

محک شبیه‌سازهای مدار کوانتومی به دلیل تحول فناوری رایانش کوانتومی، ضروری است. نیاز اصلی به محک ازاین‌جهت ناشی می‌شود که لازم است عملکرد دستگاه‌ها و شبیه‌سازهای مختلف رایانش کوانتومی به طور سامان‌مند ارزیابی و مقایسه شود. با ادامهٔ توسعهٔ سخت‌افزار و الگوریتم‌های کوانتومی، محک‌های کارآمدتر، یک چارچوب ثابت برای ارزیابی بهبودها، شناسایی نقاط ضعف و هدایت تحقیقات و توسعه را فراهم می‌کنند. باتوجه‌به رویکردهای متنوع برای اصلاح و کاهش خطا در رایانش کوانتومی، محک‌ها می‌توانند اثربخشی این رویکردها را تحت شرایط مختلف برجسته کنند. این امر به پژوهشگران و توسعه‌دهندگان این امکان را می‌دهد که روش‌های خود را اصلاح کرده و دقت

و قابلیت اطمینان کلی محاسبات کوانتومی را بهبود بخشند [۲].

شکاف‌های پژوهشی

برخلاف پیشرفت‌های قابل‌توجه در رایانش کوانتومی، چندین شکاف در این حوزه وجود دارد که نیاز به توجه دارند. یکی از شکاف‌های برجسته، کمبود محک‌های جامع و عملکردگرا برای شبیه‌سازهای کوانتومی است. باوجود محک‌های متعدد [۴–۸]، اغلب، گسترهٔ کاربردهای عملی پیش‌بینی‌شده برای رایانش کوانتومی را پوشش نمی‌دهند. کاربردهای پیش‌بینی‌شده شامل جست‌وجو در پایگاه‌داده‌های مرتب‌نشده، فاکتورکردن اعداد بزرگ به عوامل اول و مسائل بهینه‌سازی با تعداد متغیرهای قابل‌توجه هستند که اهمیت وجود رایانه‌های کوانتومی را در مسائل واقعی نشان می‌دهند. این شکاف نشان می‌دهد که نیاز به مجموعه‌ای متنوع‌تر از محک‌ها وجود دارد که منعکس‌کنندهٔ موارد استفاده واقعی باشند و بتوانند معیار دقیق‌تری از قابلیت‌های یک سامانه کوانتومی را ارائه دهند. محک‌های فعلی نمی‌توانند انواع شبیه‌سازها را در یک چارچوب بررسی کنند. چارچوب به این معنا که محک‌ها از یک اصول برای پیاده‌سازی و اعمال روی شبیه‌سازها پیروی کنند. دلیل این عدم توانایی این است که شبیه‌سازها از رویکردها و روش‌های متفاوت برای شبیه‌سازی استفاده می‌کنند. بعضی شبیه‌سازها فقط برای شبیه‌سازی یک الگوریتم طراحی شده‌اند در حالی که بعضی دیگر برای شبیه‌ساز هر الگوریتمی طراحی شده‌اند. برای مثال شبیه‌ساز [۹] به طور خاص برای شبیه‌سازی الگوریتم Shor طراحی شده است که محک آن با شبیه‌سازی‌هایی که توانایی شبیه‌سازی هر الگوریتمی را دارند متفاوت خواهد بود. از دیگر تفاوت‌هایی که می‌توان به آن اشاره کرد، محک شبیه‌سازها با استفاده از ابررایانه‌ها و یا مراکز محاسبات پردازش سریع است که نتایج بی‌فایده‌ای را برای کاربران قابل‌توجهی که شبیه‌سازها را بر روی رایانه‌های شخصی اجرا می‌کنند، تولید کرده‌اند [۱]. این نتایج فقط برای کاربرانی مفید است که به ابررایانه‌ها و مراکز پردازش سریع دسترسی دارند.

همان‌طور که اشاره شد، محک‌های فعلی نتوانستند یک چارچوب برای محک همهٔ انواع شبیه‌سازی را

طراحی کنند. این ناسازگاری، توانایی مقایسهٔ عادلانهٔ سامانه‌های مختلف را مختل می‌کند. پرداختن به این مسئله، شامل توسعهٔ محک‌هایی است که بتوانند به طور مشابه از لحاظ پیاده‌سازی و الگوریتمی (به کارگیری گیت‌ها) در پلتفرم‌های مختلف اعمال شوند و در نتیجه، ارزیابی‌های دقیق‌تر و منصفانه‌تری از فناوری‌های رایانش کوانتومی را به ارمغان بیاورند [۲]. در پژوهش ما سعی شده است که باتوجه به شرایط و امکانات موجود تا حد توان به این هدف پرداخته شود؛ اما به دلیل پیچیدگی زیاد مسئله، هنوز مشکلاتی از آن برطرف نشده است. هدف طراحی محک‌هایی است که با کم‌ترین تغییرات بتوان آن را بر روی همه شبیه‌سازها اعمال کرد. به همین دلیل، نیاز است روشی برای تبدیل تمامی الگوریتم‌ها به یک ساختار واحد که قابل‌پذیرش برای همهٔ شبیه‌سازها باشد، ارائه شود. با وجود تمامی پیچیدگی‌های این مسئله، برای پیداکردن چنین روشی در پژوهش ما گام برداشته شده است؛ این روش به طراحی یک واحد مترجم برای ترجمهٔ الگوریتم‌ها از یک‌زبان مبدأ به زبان مقصد که همان شبیه‌سازهای مختلف باشند، می‌پردازد. این در حالی است که تعداد زیاد شبیه‌سازها و عدم پیروی آن‌ها از یک رویکرد یکسان، طراحی این مترجم را پیچیده و مشکل می‌سازد.

روش پیشنهادی

در پژوهشما برای بهبود ضعف‌های موجود، مبناهای محک شبیه‌ساز به‌طورکلی متفاوت و جامع در نظر گرفته شده است. کارهای پیشین، اغلب به محک رایانه‌های واقعی کوانتومی پرداخته‌اند و توجه خاص به شبیه‌سازها کم‌تر دیده می‌شود. در اندک پژوهش‌هایی که به شبیه‌سازها نیز توجه کرده‌اند، معیارهای انتخاب‌شده برای محک، معیارهای مطلوبی در نظر گرفته نشده است. مطلوب از این بابت است که معیارها عموماً برای محک سخت‌افزار و عملکرد گیت‌ها به‌صورت مجزا انتخاب شده‌اند. معیارهای مطلوب معیارهایی هستند که برای محک شبیه‌سازها در زمان اجرای الگوریتم‌ها و بررسی عملکرد آن‌ها به‌طوری‌که همهٔ جنبه‌های مربوطه از تبدیل محاسبات کوانتومی به محاسبات کلاسیک تا ذخیرهٔ اطلاعات بدست آمده، در آن‌ها بررسی شود. علاوه‌بر معیار، با تعریف روش‌های بهتر محک، می‌توان

به معیارهای مهم نظیر زمان اجرا و میزان مصرف حافظه، توجه ویژه‌ای داشت. در روش‌های محک شبیه‌سازها نیز کم‌تر به الگوریتم‌هایی که مسائل واقعی را حل می‌کنند توجه شده است درصورتی‌که مبنای محک در پژوهش ما انتخاب الگوریتم‌هایی بوده است که از آن در حل مسائل واقعی استفاده می‌شود. علاوه بر این‌ها، فرض پژوهش‌های پیشین بر آن بوده است که کاربران شبیه‌سازها تماماً به رایانه‌های پردازش سریع و یا ابررایانه‌ها دسترسی فراوان دارند و محک‌های خود را در چنین ساختارهایی انجام داده‌اند درصورتی‌که در پژوهش ما کاربرانی در نظر گرفته شده‌اند که شبیه‌سازی‌ها را در رایانهٔ شخصی معمولی انجام می‌دهند و به ابررایانه‌ها و یا رایانه‌های کوانتومی واقعی دسترسی ندارند. در کنار تمامی موارد اشاره شده، در طول پژوهش، ابزاری[1] توسعه داده شده است که پژوهش‌های آینده در این زمینه را به سمت استانداردسازی هدایت می‌کند و فرآیند محک را آسان‌تر و سریع‌تر به پیش می‌برد.

دستاوردها

در پژوهش ما ابزار Quantum Simulator Benchmark (QSB) برای محک شبیه‌سازهای کوانتومی، طراحی شده است. بعد از انتخاب محیط آزمایش، یعنی همان رایانهٔ مورداستفاده برای شبیه‌سازی، کافی است QSB نصب و اجرا شود تا باتوجه‌به رایانه‌ای که در حال اجرای آن است، نمودارها و جدول‌های مربوط به زمان اجرا و میزان حافظهٔ مصرفی، تولید شوند. این ابزار قابلیت این را دارد که الگوریتم‌های جدید پیاده‌سازی شده در هر شبیه‌ساز را در خود جای دهد و با معیارهای مربوط، توانایی شبیه‌ساز در اجرای آن‌ها را محک بزند. با استفاده از این ابزار، تعدادی آزمایش بر روی تعدادی از شبیه‌سازهای مشهور انجام شده که نشان‌دهندهٔ رفتار متفاوت شبیه‌سازها در زمان اجرای محک‌های مختلف است. برای مثال شبیه‌ساز Cirq، در اجرای الگوریتم Grover، بسیار سریع‌تر از Simon عمل می‌کند. همچنین، به طور شگفت‌آوری، با بررسی این آزمایش‌ها و مقایسهٔ آن‌ها با خروجی‌های گزارش‌شده‌ای که بر روی رایانه‌های پردازش سریع اجرا شده است [۱]، می‌توان متوجه شد که افزایش

[1] https://github.com/ahmadrv/QSB

قدرت سخت‌افزاری بهبود نسبی چندانی در شبیه‌سازی الگوریتم‌ها با افزایش تعداد کیوبیت‌ها ندارد به این معنا که اگر چه قدرت محاسباتی چه در افزایش حجم حافظه و تعداد هسته‌های پردازشی چندین برابر شده اما مدت‌زمان اجرا به همان نسبت کاهش نیافته است. در فصل آخر، به طور کامل، به توصیف رفتارهای شبیه‌سازها در مواجه با اجرای الگوریتم‌های مختلف می‌پردازیم. در نتیجه، دستاوردهای پژوهش ما به شرح زیر است:

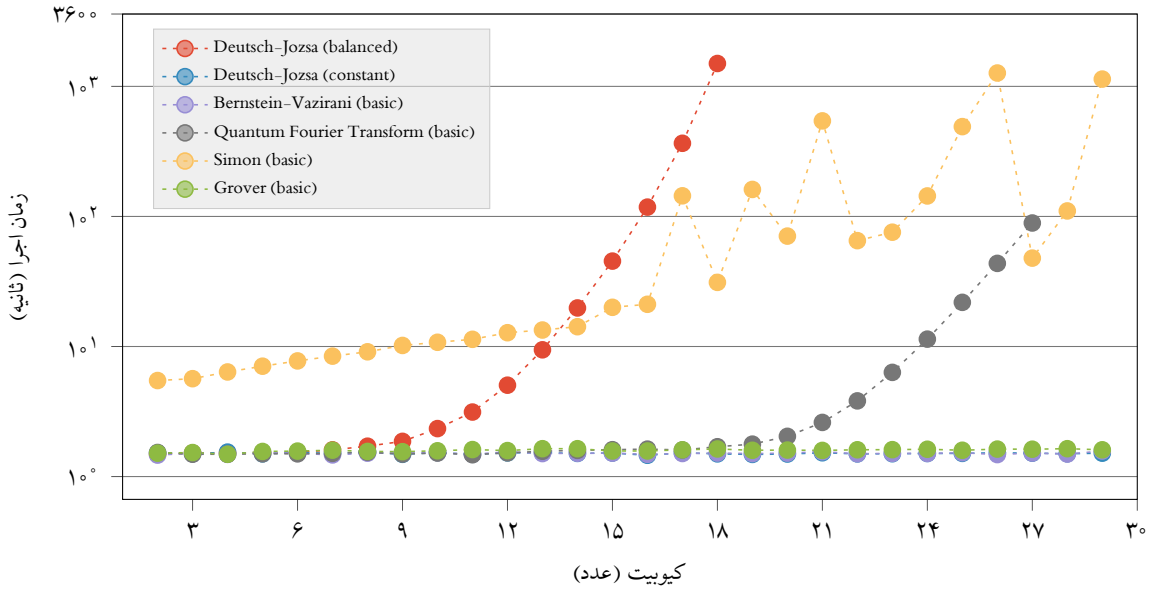
۱- محک دقیق‌تر و تخصصی‌تر شبیه‌سازهای پراستفاده در مقایسه با دیگر محک‌ها

۲- تولید یک ابزار برای کارآمد کردن و آسان‌سازی فرایند محک بر روی رایانه‌های شخصی

نتایج ابتدایی

همان‌طور که اشاره شد، با استفاده از ابزار توسعه داده شده، آزمایش‌های متفاوتی بر روی شبیه‌سازهای مختلف انجام شده است به‌طوری‌که برای مثال زمان اجرای الگوریتم‌های مختلف بر روی شبیه‌ساز Cirq باتوجه‌به افزایش تعداد کیوبیت‌ها، در شکل ۱.۱ قابل مقایسه است.

در این نتایج، رفتارهای متفاوتی از شبیه‌سازها مختلف دیده می‌شود به‌گونه‌ای که می‌توان آن‌ها را نسبت به ادعایی که در تئوری داشته‌اند مقایسه کرد. این مقایسه در فصل آخر مورد بررسی قرار گرفته شده است.



شکل ۱.۱: نمودار زمان اجرا بر حسب تعداد کیوبیت در الگوریتم‌های مختلف در شبیه‌ساز Cirq

فصل دوم

مروری بر کارهای پیشین

این بخش مفاهیم اساسی و زمینهٔ موردنیاز برای درک اصول پایه‌ای محاسبات کلاسیک و کوانتومی را پوشش می‌دهد و سعی در معرفی تاریخچه، اجزای بنیادی و تفاوت‌های کلیدی بین این دو مدل به خواننده را دارد. این توضیح باهدف جامع بودن، جزئیات تا حدی کافی را ارائه می‌دهد تا اطمینان حاصل شود که برای خوانندگانی که پیش‌زمینه‌ای در این زمینه ندارند، شفافیت داشته باشد. تمامی توضیحات این فصل برگرفته از [۱۰] می‌باشد مگر اینکه در مباحثی خاص از دیگر ارجاعات استفاده شده باشد.

اطلاعات و محاسبات کلاسیک

تکامل تاریخی محاسبات

مفهوم محاسبات با محاسبات دستی توسط انسان‌ها متولد شد. تمدن‌های اولیه مانند مصری‌ها و بابلی‌ها از ابزارهایی مانند چرتکه برای انجام عملیات ریاضی استفاده می‌کردند. مفهوم محاسبات

خودکار در قرن هفدهم با مخترعانی مانند Blaise Pascal، سازندهٔ ماشین‌حساب مکانیکی، تحول یافت. در قرن نوزدهم Charles Babbage، موتور تحلیلی را طراحی کرد، یک رایانهٔ مکانیکی همه‌منظوره که هرگز تکمیل نشد. Ada Lovelace، که اغلب به‌عنوان اولین برنامه‌نویس رایانه شناخته می‌شود، الگوریتم‌هایی برای این ماشین نوشت که پتانسیل آن را فراتر از محاسبه ساده، نشان می‌داد. قرن بیستم نشانه‌ای از گذار به رایانه‌های الکترونیکی بود. مدل نظری Alan Turing، ماشین تورینگ، اساس علم رایانه مدرن را بنا نهاد. در طول جنگ جهانی دوم، ماشین‌هایی مانند Colossus و ENIAC برای شکستن رمزها و محاسبات بالستیکی ساخته شدند. این رایانه‌های اولیه از لامپ‌های خلأ استفاده می‌کردند که بعدها در دهه ۱۹۵۰ با ترانزیستورها جایگزین شدند و به توسعه ماشین‌های کوچک‌تر و قابل‌اعتمادتر منجر شد. اختراع مدار مجتمع در دهه ۱۹۶۰ توسط Jack Kilby و Robert Noyce، محاسبات را با امکان ساخت ریزپردازنده‌های فشرده و قدرتمند متحول کرد. این جهش فناوری، راه را برای رایانه‌های شخصی در دهه ۱۹۸۰ باز کرد و قدرت محاسبات را به خانه‌ها و کسب‌وکارهای کوچک آورد. امروزه، رایانه‌های کلاسیک جزء جدایی‌ناپذیر زندگی مدرن هستند، از ارتباطات و سرگرمی تا تحقیقات علمی و خودکارسازی صنعتی. حضور گسترده آن‌ها و رابط‌های کاربری آسان، امکان انجام کارهای پیچیده را بدون نیاز به درک عمیق از فناوری زیربنایی برای افراد فراهم می‌کند.

مفاهیم بنیادی در محاسبات کلاسیک

رایانه‌های کلاسیک بر اساس منطق دودویی عمل می‌کنند، جایی که اطلاعات با استفاده از بیت‌ها نشان داده می‌شود. یک بیت، یک رقم دودویی است که می‌تواند یکی از دو مقدار ۰ یا ۱ را داشته باشد. این بیت‌ها واحدهای اصلی اطلاعات در یک سامانهٔ دیجیتال هستند. در سامانه‌های دیجیتال مفهوم دیگری به نام گیت‌های منطقی مطرح است. گیت‌ها برای اعمال تغییرات بر روی بیت‌ها مورداستفاده قرار می‌گیرند. در ادامه به معرفی ابتدایی تعدادی از این گیت‌ها می‌پردازیم.

گیت AND: تنها زمانی که هر دو ورودی ۱ باشند، خروجی ۱ می‌دهد. این گیت عملیات پیوند منطقی را انجام می‌دهد.

گیت OR: اگر حداقل یکی از ورودی‌ها ۱ باشد، خروجی ۱ می‌دهد. این گیت عملیات جداکنندهٔ منطقی را انجام می‌دهد.

گیت NOT: مقدار ورودی را معکوس می‌کند، به طوری که اگر ورودی ۰ باشد، خروجی ۱ و بالعکس.

این گیت‌ها می‌توانند با هم ترکیب شوند تا مدارهای پیچیده‌تری ایجاد کنند که قابلیت انجام عملیات‌های ریاضی و منطقی را دارند. به عنوان مثال، یک مدار نیم‌جمع‌کننده با ترکیب یک گیت AND و یک گیت XOR دو عدد دودویی را جمع می‌کند و حاصل جمع و بیت نقلی تولید می‌کند.

برگشت‌پذیری و جامعیت‌پذیری محاسبات

بیشتر گیت‌های منطقی کلاسیک غیرقابل‌برگشت هستند، به این معنا که خروجی به طور یکتا ورودی را تعیین نمی‌کند. به‌عنوان‌مثال، دانستن این که خروجی یک گیت AND برابر با ۰ است، اطلاعات دقیقی درباره ورودی‌ها نمی‌دهد. بااین‌حال، برخی عملیات مانند گیت Toffoli (که به‌عنوان گیت NOT کنترل‌شده نیز شناخته می‌شود) برگشت‌پذیر هستند. گیت Toffoli حالت بیت خروجی را درصورتی‌که بیت‌های کنترل در حالت مشخصی باشند، معکوس می‌کند.

برگشت‌پذیری یک مفهوم کلیدی در محاسبات کوانتومی است، جایی که همهٔ عملیات باید به دلیل قوانین مکانیک کوانتومی برگشت‌پذیر باشند. این خاصیت تضمین می‌کند که اطلاعات در طول محاسبه از بین نمی‌رود و یکپارچگی حالت‌های کوانتومی حفظ می‌شود.

مجموعهٔ گیت‌های جامع

در محاسبات کلاسیک، یک مجموعهٔ گیت جامع مجموعه‌ای از گیت‌هاست که می‌توانند برای انجام هر عملیات منطقی ترکیب شوند. گیت، NAND به‌عنوان‌مثال، یک مجموعهٔ جامع است؛ زیرا هر گیت دیگری (مانند AND ، NOT و غیره) را می‌توان با استفاده از آن ساخت. این مفهوم در محاسبات کوانتومی نیز به کار می‌رود، جایی که یک مجموعهٔ گیت‌های کوانتومی جامع و عمومی مانند $\{\mathbf{CNOT}, \mathbf{H}, \mathbf{T}\}$ می‌تواند برای انجام هر عملیات کوانتومی استفاده شود.

تصحیح خطا و پیچیدگی محاسباتی

تصحیح خطای کلاسیک

خطاها در محاسبات کلاسیک ممکن است از منابع مختلفی مانند نویز الکتریکی یا خرابی‌های سخت‌افزاری ناشی شوند. فن‌های تصحیح خطا برای اطمینان از انتقال و پردازش صحیح داده‌ها، ضروری هستند. یکی از روش‌های ساده، کد تکرار است که در آن هر بیت چندین بار تکرار می‌شود (مثلاً ۰ به ۰۰۰ تبدیل می‌شود) تا بتوان خطاها را با استفاده از رأی‌گیری اکثریت، شناسایی و اصلاح کرد.

تصحیح خطای کوانتومی

حالت‌های کوانتومی به دلیل برهم‌کنش‌های محیطی و سایر پدیده‌های کوانتومی بیش‌تر مستعد خطا هستند. رمزهای تصحیح خطای کوانتومی مانند کد Shor اطلاعات کوانتومی را با رمزگذاری یک بیت کوانتومی در چند بیت فیزیکی محافظت می‌کنند و به‌این‌ترتیب می‌توانند خطاها را شناسایی و اصلاح کنند بدون اینکه حالت کوانتومی را مستقیماً اندازه‌گیری کنند.

پیچیدگی محاسباتی

نظریهٔ پیچیدگی محاسباتی، مسائل را بر اساس منابع موردنیاز برای حل آن‌ها، مانند زمان و حافظه، طبقه‌بندی می‌کند. در محاسبات کلاسیک، مسائل قابل‌حل در زمان چندجمله‌ای (P)، ساده در نظر گرفته می‌شوند، درحالی‌که مسائل قابل‌حل توسط رایانه‌های کوانتومی در زمان چندجمله‌ای (BQP) با تفاوتی که با مسائل سادهٔ کلاسیک دارد، می‌تواند مزیت بالقوه‌ای در محاسبات کوانتومی را نشان دهد. این تفاوت‌ها دلالت‌های نظری و عملی محاسبات کوانتومی در حل مسائل را برجسته می‌کند. در ادامه، در طول قسمت‌های مختلف به این تفاوت‌ها اشاره خواهد شد.

نقش محاسبات کلاسیک در محاسبات کوانتومی

درک محاسبات کلاسیک برای درک مفاهیم محاسبات کوانتومی ضروری است. الگوریتم‌های کوانتومی اغلب شامل اجزای کلاسیک هستند و سامانه‌های کنترل کلاسیک، مدیریت عملیات کوانتومی را بر عهده دارند. به‌عنوان‌مثال، اجرای تصحیح خطای کوانتومی به پردازش کلاسیک نیاز دارد تا خطاها را در حالت‌های کوانتومی شناسایی و اصلاح کند. علاوه بر این، بسیاری از الگوریتم‌های کوانتومی مانند الگوریتم‌های Shor و Grover همتایان کلاسیک دارند که بینش‌هایی درباره همتایان کوانتومی آن‌ها ارائه می‌دهد.

اطلاعات و محاسبات کوانتومی

معرفی اطلاعات کوانتومی

محاسبات کوانتومی الگوی جدیدی را معرفی می‌کند که در آن اطلاعات با استفاده از بیت‌های کوانتومی یا کیوبیت‌ها نمایش داده می‌شود. برخلاف بیت‌های کلاسیک که تنها می‌توانند در یکی از دو حالت (۰ یا ۱) باشند، کیوبیت‌ها می‌توانند به طور هم‌زمان در ترکیبی از هر دو حالت وجود داشته باشند. این

خاصیت از اصول مکانیک کوانتومی، به‌ویژه، برهم‌نهی خطی حالت‌های کوانتومی، ناشی می‌شود.

یک کیوبیت می‌تواند به‌صورت $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ نمایش داده شود که در آن α و β اعداد مختلط هستند و $|\alpha|^2 + |\beta|^2 = 1$. این نمایش، احتمالات اندازه‌گیری کیوبیت در حالت $|0\rangle$ یا $|1\rangle$ را توصیف می‌کند. ضریب‌های α و β به‌عنوان دامنه‌های احتمال شناخته می‌شوند و مربع اندازه آن‌ها احتمال مربوطه را مشخص می‌کند.

کیوبیت‌ها و حالت‌های کوانتومی

اصل برهم‌نهی اجازه می‌دهد یک کیوبیت اطلاعات بیشتری نسبت به یک بیت کلاسیک حمل کند. درحالی‌که یک بیت کلاسیک فقط می‌تواند در یکی از دو حالت باشد، یک کیوبیت می‌تواند هر ترکیبی از این حالت‌ها را به‌صورت هم‌زمان نشان دهد. این قابلیت منجر به محاسبات موازی می‌شود که یک خاصیت ذاتی در محاسبات کوانتومی است.

حالت‌های کوانتومی را می‌توان بر روی Bloch sphere، یک نمایش هندسی، تجسم کرد. هر نقطه روی سطح این نمایش هندسی، نشان‌دهندهٔ یک حالت خالص کوانتومی است. حالت‌های خاص مانند $|0\rangle$ و $|1\rangle$ در قطب‌های شمال و جنوب کره قرار دارند، درحالی‌که حالات دیگر روی سطح پراکنده‌اند.

اندازه‌گیری و اصل ناپیوستگی

یکی از تفاوت‌های کلیدی بین محاسبات کلاسیک و کوانتومی در نحوهٔ پردازش اطلاعات کوانتومی در طول اندازه‌گیری است. برخلاف اندازه‌گیری‌های کلاسیک که اطلاعات را بدون تأثیر بر سامانه اندازه‌گیری می‌کنند، اندازه‌گیری در سامانه‌های کوانتومی می‌تواند حالت کوانتومی را تغییر دهد یا به‌اصطلاح ناهمدوس کند.

هنگامی که یک کیوبیت اندازه‌گیری می‌شود، حالت آن به یکی از دو حالت پایه‌ای $|0\rangle$ یا $|1\rangle$ به‌اصطلاح،

فرومی‌ریزد. احتمال اندازه‌گیری هر حالت پایه‌ای با مربع اندازه دامنهٔ مربوطه تعیین می‌شود. این اصل ناپیوستگی به این معناست که نمی‌توانیم قبل از اندازه‌گیری پیش‌بینی قطعی از نتیجه اندازه‌گیری داشته باشیم، بلکه فقط احتمالات ممکن را می‌توانیم محاسبه کنیم.

درهم‌تنیدگی کوانتومی

درهم‌تنیدگی کوانتومی یکی از پدیده‌های برجسته مکانیک کوانتومی است که هنگامی رخ می‌دهد که حالت کوانتومی دو یا چند ذره به طور جدایی‌ناپذیری به هم مرتبط شود. درهم‌تنیدگی اجازه می‌دهد که اندازه‌گیری حالت یکی از ذرات بلافاصله حالت ذره دیگر را تعیین کند، حتی اگر این ذرات در فواصل دور از هم قرار داشته باشند. این پدیده «شبح‌وار بودن عمل در فاصله» را که توسط Albert Einstein به‌عنوان «کنشی شگفت‌انگیز» توصیف شده، معرفی می‌کند.

یکی از نمونه‌های معروف درهم‌تنیدگی، Bell's theorem است که در آن دو کیوبیت در حالتی هستند که هیچ توصیف محلی و قطعی نمی‌تواند حالت هر دو کیوبیت را به طور مستقل تعیین کند. درهم‌تنیدگی، پایه‌ای برای بسیاری از پروتکل‌های کوانتومی مانند رمزنگاری کوانتومی و انتقال آنی کوانتومی است.

در ادامه به مفهومی به نام «رتبهٔ Schmidt» می‌پردازیم که در توضیح میزان درهم‌تنیدگی دو زیرسیستم به ما کمک می‌کند.

تجزیه و رتبهٔ Schmidt

رتبهٔ Schmidt مفهومی در نظریه اطلاعات کوانتومی است که به‌ویژه در زمینهٔ حالت‌های کوانتومی دوگانه اهمیت دارد. این رتبه به تعداد جملات ناصفر در تجزیهٔ Schmidt یک حالت خالص اشاره دارد که دو زیرسیستم (که اغلب به‌صورت A و B نشان داده می‌شوند) را توصیف می‌کند.

هر حالت خالص $|\psi\rangle$ که در فضای $H_A \otimes H_B$ (محصول تانسوری فضای سیستم‌های A و B) تعریف

شده باشد، می‌تواند به‌صورت تجزیهٔ Schmidt بیان شود:

$$|\psi\rangle = \sum_{i=۱}^{r} c_i |a_i\rangle |b_i\rangle$$

در حالی که

- c_i ضرایب غیرمنفی هستند.
- $|a_i\rangle$ بردارهای پایهٔ متعامد برای زیرسیستم A هستند.
- $|b_i\rangle$ بردارهای پایهٔ متعامد برای زیرسیستم B هستند.
- r، که به عنوان رتبهٔ Schmidt شناخته می‌شود، برابر است با تعداد ضرایب ناصفر ($c_i > ۰$)

با دانستن رتبهٔ Schmidt دیدگاهی درباره سطح درهم‌تنیدگی بین دو زیرسیستم بدست می‌آید. اگر $r = ۱$ باشد، حالت قابل جداسازی است؛ یعنی هیچ درهم‌تنیدگی بین آن‌ها وجود ندارد ولی اگر $r > ۱$ شود، حالت نوعی درهم‌تنیدگی را نشان می‌دهد که نشان‌گر جدایی‌ناپذیری سیستم خواهد بود؛ رتبه‌های بالاتر نشان‌دهندهٔ روابط پیچیده‌تری بین زیرفضاها هستند.

به طور خلاصه، درک رتبهٔ Schmidt به فیزیک‌دانان و پژوهش‌گران در زمینهٔ سیستم‌های کوانتومی بینشی بهتر درباره ساختار این سیستم‌ها از نظر خاصیت جدایی‌پذیری در مقابل ویژگی‌های درهم‌تنیدگی می‌دهد.

تصحیح خطای کوانتومی

یکی از چالش‌های اصلی در محاسبات کوانتومی مقابله با خطاهایی است که در اثر ناهمدوسی و نویز ایجاد می‌شوند. تصحیح خطای کوانتومی برای حفظ یکپارچگی اطلاعات کوانتومی ضروری است. برخلاف بیت‌های کلاسیک که می‌توانند به طور مستقل کنترل شوند، کیوبیت‌ها حساس به نویز هستند

و ممکن است خطاهای پیچیده‌ای مانند تغییر بیت و یا تغییر فاز را تجربه کنند.

کد Shor

یکی از اولین کدهای تصحیح خطای کوانتومی است که یک کیوبیت منطقی را به نُه کیوبیت فیزیکی رمزگذاری می‌کند. این کد می‌تواند خطاهای تک‌کیوبیتی دلخواه را با شناسایی و تصحیح هر دو نوع خطای تغییر بیت و تغییر فاز، اصلاح کند. کد با پخش اطلاعات کوانتومی در میان چندین کیوبیت کار می‌کند که اجازه می‌دهد خطاها بدون اندازه‌گیری مستقیم حالت، شناسایی و اصلاح شوند.

اندازه‌گیری Syndrome

در تصحیح خطای کوانتومی، اندازه‌گیری Syndrome برای شناسایی وجود و نوع خطاها انجام می‌شود. این اندازه‌گیری حالت کوانتومی را به یک زیرفضای مربوط به Syndrome خطا نگاشت می‌کند، که امکان شناسایی و تصحیح خطاها را فراهم می‌کند. در اندازه‌گیری‌های Syndrome یکپارچگی حالت کوانتومی حفظ می‌شود چرا که اطلاعات کوانتومی کدگذاری شده را مختل نمی‌کنند.

تحلیل تطبیقی: محاسبات کلاسیک در مقابل کوانتومی

محاسبات کوانتومی چندین مزیت نسبت به محاسبات کلاسیک دارد، به‌ویژه در حوزهٔ برخی از مسائل محاسباتی. درحالی‌که رایانه‌های کلاسیک به عملیات قطعی بر روی بیت‌ها متکی هستند، رایانه‌های کوانتومی از ماهیت احتمالی مکانیک کوانتومی بهره می‌برند و از کیوبیت‌ها برای تبدیل و فناوری اطلاعات استفاده می‌کنند.

بهبود پیچیدگی در مدل کوانتومی

الگوریتم‌های کوانتومی می‌توانند بهبود پیچیدگی نمایی در حل برخی مسائل ارائه دهند. به‌عنوان‌مثال، الگوریتم Shor می‌تواند اعداد بزرگ را به طور نمایی، سریع‌تر از بهترین الگوریتم‌های کلاسیک فاکتور کند که می‌تواند تهدیدی برای سامانه‌های رمزنگاری کلاسیک باشد. از سوی دیگر، الگوریتم Grover برای مسائل جستجوی بی‌ساختار، بهبود مربعی در پیچیدگی محاسباتی راه‌حل مسئله، ارائه می‌دهد.

چالش‌های محاسبات کوانتومی

باوجود پتانسیل‌ها، محاسبات کوانتومی با چالش‌های بزرگی مواجه است. حفظ همدوسی کیوبیت‌ها به دلیل تعاملات محیطی که منجر به ناهمدوسی می‌شوند، دشوار است. علاوه بر این، پیاده‌سازی گیت‌های کوانتومی بادقت بالا نیاز به کنترل دقیق بر سامانه‌های کوانتومی دارد. رایانه‌های کوانتومی فعلی نیز محدود به تعداد کیوبیت‌ها و دقت عملیات کوانتومی هستند.

در نهایت، محاسبات کلاسیک و کوانتومی نمایانگر دو رویکرد اساسی به محاسبات هستند. درحالی‌که محاسبات کلاسیک برای مسائل قطعی و به‌خوبی درک شده مناسب است، محاسبات کوانتومی پتانسیل انقلابی در حل مسائل در حوزه‌هایی را دارد که رویکردهای کلاسیک ناکارآمد یا غیرعملی هستند. بااین‌حال، توسعهٔ رایانه‌های کوانتومی عملی، نیازمند غلبه بر چالش‌های فنی قابل‌توجهی از جمله تصحیح خطا و تحمل خطا است.

شبیه‌سازهای مدار کوانتومی

در این قسمت، یک مرور جامع از مدل‌های مختلف شبیه‌سازی و شبیه‌سازهای محاسبات کوانتومی انجام شده است که نقاط قوت، ضعف‌ها و محدودیت‌های آن‌ها را به‌تفصیل بیان می‌کند. مقایسه بین مدل‌های مختلف، ملاحظات مربوط به‌دقت، مصرف منابع، مقیاس‌پذیری و انعطاف‌پذیری را برجسته

می‌کند و بینش‌های ارزشمندی برای انتخاب ابزارهای شبیه‌سازی مناسب برای وظایف خاص محاسبات کوانتومی ارائه می‌دهد. تمامی قسمت‌های این بخش برگرفته از پژوهش [۱۱] است مگر آن که به طور خاص به موارد دیگر ارجاع داده شده باشد.

شبیه‌سازهای مبتنی بر Schrödinger

این شبیه‌سازها به دلیل نمایش مستقیم حالت‌ها و تغییرات کوانتومی از دقت بالایی برخوردارند. به‌علاوه قادر به مدیریت مدارهای کوانتومی متوسط (حدود ۳۰ تا ۴۰ کیوبیت) هستند و قابلیت اجرا بر روی HPC[1] را دارند. بااین‌همه، ضعف‌هایی در این دسته دیده می‌شود. نیاز به منابع محاسباتی قابل‌توجه، از جمله حافظه و قدرت پردازش، به دلیل رشد نمایی فضای حالت با تعداد کیوبیت‌ها و در نتیجه عدم مقیاس‌پذیری برای سامانه‌های بسیار بزرگ، نمونه‌ای از آن‌ها هستند.

شبیه‌ساز کوانتومی Intel (IQS): شبیه‌ساز کوانتومی Intel (IQS) از پلتفرم‌های محاسبات سریع (HPC) و ابری برای شبیه‌سازی حالت‌های کوانتومی توزیع شده با استفاده از گروه‌هایی از پردازنده‌های مرکزی (CPU) و پردازنده‌های گرافیکی (GPU) بهره می‌برد. IQS بردارهای دامنه‌های پیچیده را تقسیم کرده و هر قسمت را به فرایندهای مختلف تخصیص می‌دهد. این شبیه‌ساز قادر است سامانه‌هایی تا ۴۰ کیوبیت را شبیه‌سازی کند. تاکنون الگوریتم‌هایی مانند بهینه‌سازی تقریبی کوانتومی (QAOA) برای Max-Cut در گراف‌های ۳ـ منظم، تخمین فاز و تبدیل فوریه کوانتومی (QFT) با استفاده از آن شبیه‌سازی شده‌اند.

QuEST: از بردارها برای نمایش حالت‌های خالص و از برهم‌نهی‌ها و ماتریس‌های چگالی برای حالت‌های کوانتومی مختلط استفاده می‌کند. این شبیه‌ساز از موازی‌سازی از طریق OpenMP و MPI پشتیبانی می‌کند که آن را برای پلتفرم‌های توزیع شده و چندرشته‌ای مناسب می‌سازد. QuEST می‌تواند

[1] High-Performance Computer

مدارهای کوانتومی شبه‌تصادفی را تا ۳۸ کیوبیت شبیه‌سازی کند و یک‌زبان مبتنی بر C ارائه می‌دهد که روی پلتفرم‌های ترتیبی، چندرشته‌ای و موازی اجرا می‌شود.

Qiskit Aer: توسط IBM Quantum و Qiskit ارائه شده است که برخلاف شبیه‌سازهای مبتنی بر نمونه‌برداری، تابع موج را به‌عنوان یک بردار حالت با اندازهٔ ۲^n محاسبه و بازمی‌گرداند. این محاسبه به طور مداوم در حین اعمال گیت‌ها و دستورالعمل‌ها انجام می‌شود. این شبیه‌ساز از شبیه‌سازی‌های نویزی و ایده‌آل پشتیبانی کرده و مقیاس‌پذیری تا ۳۲ کیوبیت را پوشش می‌دهد.

Cirq: این شبیه‌ساز که توسط Google توسعه‌یافته شده است، از ماتریس‌های پراکنده[1] برای شبیه‌سازی حالت‌های خالص استفاده می‌کند. این شبیه‌ساز قادر است تا سخت‌افزار کوانتومی واقعی را شبیه‌سازی کند. مثلاً در حالتی که وضعیت اولیه به طور کامل صفر باشد و بردار حالت اولیه در دسترس نباشد. این شبیه‌ساز از اجراهای تصادفی نیز پشتیبانی می‌کند. درحالی‌که شبیه‌ساز وضعیت بردار را در پایان شبیه‌سازی ارائه می‌دهد، وضعیت اولیه می‌تواند با استفاده از یک بردار حالت کامل تعیین شود؛ اما این خود ضعف‌هایی مثل وابستگی به سخت‌افزار را در پی دارد. علاوه بر این، از شبیه‌سازی‌های خالص و نویزی پشتیبانی می‌کند، به‌طوری که شبیه‌سازی‌های نویزی از روش‌های تابع موج Monte Carlo استفاده می‌کنند که در آن عملگرهای Kraus به طور تصادفی نمونه‌برداری و به تابع موج اعمال می‌شوند.

qsim: یک شبیه‌ساز بردار حالت پیاده‌سازی شده با زبان C++ توسط TensorFlow Quantum است که برای استفاده بر روی رایانه‌های شخصی طراحی شده و خروجی شبیه‌سازی را به‌صورت یک بردار حالت کامل ارائه می‌دهد. این شبیه‌ساز می‌تواند تا ۳۰ کیوبیت را با ۱۶ گیگابایت حافظه شبیه‌سازی کند، اگرچه نیاز به حافظه با هر کیوبیت اضافی دوبرابر می‌شود. شبیه‌ساز با انجام ضرب ماتریس - بردار برای هر گیت کار می‌کند و پیچیدگی زمان اجرای آن $O(g \cdot ۲^n)$ است، درحالی‌که g تعداد گیت‌های

[1] Sparse Matrices

۲ ـ کیوبیتی است. بهینه‌سازی‌هایی مانند ادغام گیت‌ها، محاسبات بادقت یک رقم اعشار و موازی‌سازی با استفاده از OpenMP برای بهبود عملکرد به‌کارگرفته شده است.

شبیه‌سازهای ترکیبی Schrödinger-Feynman

این طیف از شبیه‌سازها با بهره‌گیری از ترکیب نقاط قوت هر دو روش، امکان شبیه‌سازی کارآمد مدارهای کوانتومی بزرگ‌تر را فراهم کرده‌اند. همچنین، پشتیبانی از موازی‌سازی که می‌تواند کارایی محاسباتی را به طور قابل‌توجهی افزایش دهد، از دیگر نقاط قوت است. پیچیدگی بیش‌تر در پیاده‌سازی و بهینه‌سازی به دلیل طبیعت ترکیبی روش، نیاز به منابع محاسباتی قابل‌توجه به‌ویژه برای مدارهایی با عمق زیاد از نقاط ضعف این دسته هستند.

DDSIM: از روش ترکیبی Schrödinger-Feynman برای شبیه‌سازی محاسبات کوانتومی با استفاده از نمودارهای تصمیم[1] (QuIDD) استفاده می‌کند. این روش تغییرات کوانتومی را با تجزیه بازگشتی ماتریس‌ها به ماتریس‌های دو در دو ثبت می‌کند. تجزیهٔ Schmidt به گیت‌های دو کیوبیتی اعمال می‌شود که به‌صورت ضرب‌های تنسوری نمایش داده می‌شوند. این روش ترکیبی برای مدارهایی با عمق محدود مناسب است؛ زیرا تعداد اجراهای شبیه‌سازی با افزایش تعداد گیت‌های متقابل، به‌صورت نمایی، رشد می‌کند.

Rollright: این شبیه‌ساز توسط دانشگاه Michigan و Google توسعه‌یافته شده است. در این شبیه‌ساز روش‌های Schrödinger و Feynman ترکیب شده‌اند. این شبیه‌ساز کیوبیت‌ها را بر اساس استفاده از گیت‌ها گروه‌بندی کرده و گیت‌های کنترل‌شده مشترک در گروه‌ها را تجزیه می‌کند و قبل از ادغام نتایج در محاسبه کلی، محاسبات گیت‌های Schrödinger را روی هر گروه انجام می‌دهد.

[1] Decision-Diagram

qsimh: این شبیه‌ساز برگرفته و گسترش‌یافته‌ای از qsim است که شبیه‌سازی مسیر Feynman را با تقسیم شبکهٔ الگوریتم‌های کوانتومی به دو قسمت با استفاده از تجزیهٔ Schmidt پشتیبانی می‌کند. این شبیه‌ساز از موازی‌سازی در دستگاه‌های تک و چندگانه پشتیبانی می‌کند و ادعای آن بر این است که شبیه‌سازی‌های بیش از ۵۰ کیوبیت را مدیریت کند.

شبیه‌سازهای مبتنی بر Heisenberg

از کاربردهای درخشان این دسته از شبیه‌سازها، شبیه‌سازی کارآمد مدارهای پایدارکننده و معماری‌های خاص مقاوم به خطا است. بهینه‌سازی‌های پیشرفته مانند وارونگی جدول پایداری که سرعت شبیه‌سازی را افزایش می‌دهد، از دیگر برتری‌های این دسته به شمار می‌رود. از آن‌جایی که در صورت توجه خاص به یک بعد از بهینگی، دیگر ابعاد دچار کمبود خواهند شد، این قضیه در طراحی شبیه‌سازها نیز مستثنیٰ نیست. چون این دسته از شبیه‌سازها محدود به انواع خاصی از مدارها (مثلاً مدارهای پایدارکننده) هستند برای شبیه‌سازی عمومی مدارهای کوانتومی مناسب نمی‌باشند. دیگر نقطه‌ضعف این دسته، شبیه‌سازهای در حال توسعهٔ آن است که مستندات و پشتیبانی کاربر جامع و کاملی در دسترس ندارند.

CHP: یک شبیه‌ساز تخصصی برای مدارهای پایدارکنندهٔ کوانتومی است که برای مدیریت مدارهای بزرگ با هزاران کیوبیت بهینه شده است. این شبیه‌ساز از طراحی و رفع اشکال سامانه‌های تصحیح خطای کوانتومی و شبیه‌سازی حالت‌های بسیار درهم‌تنیده پشتیبانی می‌کند. CHP از یک‌زبان اسمبلی ساده برای مشخص‌کردن مدارها استفاده می‌کند و شامل گیت اندازه‌گیری تک‌کیوبیتی است. باوجود کارایی بالا، عملکرد CHP به دلیل افزایش منابع موردنیاز به‌صورت درجه دوم با افزایش تعداد کیوبیت‌ها محدود می‌شود.

Stim: توسعهٔ این شبیه‌ساز توسط Google انجام شده است که با اعمال بهبودهایی همچون بردارسازی، نمونه‌برداری از فریم مرجع و وارونگی جدول پایداری، کارایی شبیه‌ساز CHP را ارتقا

می‌دهد. این بهبودها امکان پردازش کارآمدتر، به‌ویژه در مدیریت معماری‌های بزرگ و مقاوم به خطا در مدارهای کوانتومی را فراهم می‌کنند. Stim می‌تواند مدارهای پیچیده مانند مدارهای surface code با ۲۰۰۰۰ کیوبیت را در عرض ۱۵ ثانیه تحلیل کند.

مقایسه مدل‌های مختلف شبیه‌سازی

Schrödinger در مقابل ترکیبی Schrödinger–Feynman

شبیه‌سازهای مبتنی بر Schrödinger، در توسعه به بلوغ نسبی بیش‌تری رسیده‌اند و درعین‌حال شبیه‌سازی دقیق‌تر حالت‌های کوانتومی را فراهم می‌کنند اما مصرف غیربهینهٔ منابع از مشکلات آن‌ها است درحالی‌که شبیه‌سازهای ترکیبی Schrödinger–Feynman، همان‌طور که اشاره شد، با بهره‌گیری از موازی‌سازی و نقاط قوت دو روش ذیل، شبیه‌سازی‌های کارآمدتری را ارائه می‌دهند؛ ولی این برتری با پیچیدگی بیش‌تر در پیاده‌سازی و بهینه‌سازی همراه است.

ترکیبی Schrödinger–Feynman در مقابل Heisenberg

همان‌طور که گفته شد، شبیه‌سازهای ترکیبی Schrödinger–Feynman، شبیه‌سازهای کارآمدتر ولی با مشکل پیچیدگی در پیاده‌سازی هستند درحالی‌که شبیه‌سازهای مبتنی بر Heisenberg از دیگر شبیه‌سازها، کارآمدتر هستند؛ ولی فقط برای مدارهای خاص مثل مدارهای پایدارکننده قابل استفاده‌اند و برای شبیه‌سازی مدارهای عمومی کوانتومی مناسب نیستند.

روش‌های محک

در این بخش با بهره‌گیری از پژوهش [۱۲] به بررسی روش‌های محک می‌پردازیم.

در حوزهٔ محاسبات کلاسیک، محک، برای بررسی جنبه‌های مختلف عملکرد سامانه، مانند سرعت،

کارایی و استفاده از منابع، تکامل‌یافته است. بااین‌حال، ماهیت منحصربه‌فرد محاسبات کوانتومی چالش‌های جدیدی را به وجود می‌آورد که نیازمند تطبیق روش‌های سنتی با حوزهٔ کوانتوم است.

محک، شامل اجرای مجموعه‌ای از آزمایش‌های استاندارد روی سامانه‌های مختلف برای اندازه‌گیری و مقایسه عملکرد آن‌هاست. هدف این محک‌ها فراهم‌کردن یک روش قابل‌اعتماد و عینی برای ارزیابی قابلیت‌های سامانه‌های محاسباتی مختلف است که به کاربران و توسعه‌دهندگان امکان می‌دهد نقاط قوت و ضعف سامانه‌های مختلف را درک کنند.

به طور سنتی، محک در محاسبات کلاسیک بر روی معیارهایی مانند زمان اجرا، توان عملیاتی و کارایی متمرکز بوده است. این معیارها با استفاده از مجموعه‌ای از آزمایش‌های محک، از محک‌های ساختگی که وظایف محاسباتی خاصی را شبیه‌سازی می‌کنند تا محک‌های کاربردی واقعی که عملکرد سامانه‌ها را در شرایط استفاده معمولی منعکس می‌کنند، ارزیابی می‌شوند.

محاسبات کوانتومی تفاوت‌های اساسی با محاسبات کلاسیک دارد که نیازمند بازنگری روش‌های محک است. سامانه‌های کوانتومی با ویژگی‌هایی مانند برهم‌نهی، درهم‌تنیدگی و همدوسی کوانتومی مشخص می‌شوند که هیچ معادل مستقیمی در سامانه‌های کلاسیک ندارند. این ویژگی‌ها بر نحوهٔ پردازش اطلاعات و چگونگی اندازه‌گیری عملکرد، تأثیر بسزایی می‌گذارند.

یکی از چالش‌های اصلی در محک کوانتومی، نبود یک روش محاسباتی متحد در میان سامانه‌های کوانتومی مختلف است. برخلاف محاسبات کلاسیک که معماری‌ها نسبتاً استاندارد هستند، رایانه‌های کوانتومی می‌توانند بسته به فناوری زیربنایی (مانند یون‌های به‌دام‌افتاده، کیوبیت‌های ابررسانا، کیوبیت‌های فوتونی) به طور قابل‌توجهی متفاوت باشند. این تنوع، توسعهٔ یک محک واحد که بتواند به طور جامع در تمام سامانه‌های کوانتومی اعمال شود را دشوار می‌کند.

علاوه بر این، وضعیت نوپای فناوری محاسبات کوانتومی به این معناست که محک‌ها باید عواملی مانند نویز، نرخ خطا و تعداد محدود کیوبیت‌های موجود در سامانه‌های کنونی را در نظر بگیرند. این

عوامل می‌توانند تأثیر زیادی بر عملکرد الگوریتم‌های کوانتومی داشته باشند و باید در طراحی محک‌ها به‌دقت در نظر گرفته شوند.

چندین روش برای محک سامانه‌های کوانتومی پیشنهاد شده است که هر یک بر جنبه‌های مختلف عملکردی تمرکز دارند. در زیر به برخی از رایج‌ترین روش‌های محک در محاسبات کوانتومی اشاره شده است:

محک تصادفی: محک تصادفی یک روش بسیار پرکاربرد برای اندازه‌گیری Fidelity گیت‌های کوانتومی است. به زبان ساده، Fidelity ، میزان شباهت نتیجهٔ عملی را با نتیجهٔ دلخواه می‌سنجد. با اعمال یک دنباله از گیت‌های تصادفی کوانتومی و سپس معکوس‌های آن‌ها، نرخ خطای کلی یک سیستم کوانتومی را ارزیابی می‌کند. این تکنیک به ویژه در برابر خطاهای آماده‌سازی حالت و اندازه‌گیری مقاوم است که آن را برای ارزیابی عملکرد گیت‌های کوانتومی در محیط‌های پر نویز مفید می‌سازد.

Quantum Volume (QV): QV یک معیار جامع است که بزرگ‌ترین مدار کوانتومی که می‌توان با موفقیت بر روی یک رایانه کوانتومی اجرا کرد را اندازه‌گیری می‌کند. این معیار عواملی مانند تعداد کیوبیت‌ها، Fidelity گیت‌ها و اتصال کیوبیت‌ها را در نظر می‌گیرد و یک عدد واحد را که نشان‌دهندهٔ عملکرد کلی سامانه است ارائه می‌دهد. بااین‌حال، این معیار محدودیت‌هایی دارد؛ زیرا به طور کامل عملکرد مدارهایی با طرح‌های غیرمربع (مدارهایی با عمق زیاد و عرض کم یا بالعکس) را منعکس نمی‌کند.

مدارهای آینه‌ای: روش مدار آینه‌ای شامل ایجاد مداری است که معکوس مدار کوانتومی داده شده باشد. با مقایسه نتایج مدارهای مستقیم و معکوس، این روش می‌تواند سازگاری و دقت محاسبات کوانتومی را ارزیابی کند. مدارهای آینه‌ای به‌ویژه برای محک مدارهایی که شامل گیت‌های Clifford و $Z(\theta)$ هستند که در بسیاری از پردازنده‌های کوانتومی رایج‌اند، مفید هستند.

محک آنتروپی متقاطع: این روش برای ارزیابی برتری کوانتومی با مقایسه توزیع خروجی یک رایانهٔ کوانتومی با توزیع نظری مورد انتظار، استفاده می‌شود. با ارزیابی اینکه چقدر خروجی واقعی با توزیع مورد انتظار تطابق دارد، محک آنتروپی متقاطع معیاری از توانایی یک رایانهٔ کوانتومی برای انجام وظایفی که باور بر این است که برای سامانه‌های کلاسیک غیرقابل‌حل هستند، فراهم می‌کند.

همان‌طور که فناوری محاسبات کوانتومی به تکامل خود ادامه می‌دهد، توسعه روش‌های محک استاندارد برای پیشرفت این حوزه حیاتی خواهد بود. محک‌های آینده باید با معماری‌های جدید کوانتومی قابل‌تطبیق باشند و قادر به ارائهٔ مقایسه‌های معنادار در میان سامانه‌های مختلف باشند. علاوه بر این، همان‌طور که رایانه‌های کوانتومی قدرتمندتر می‌شوند، محک‌ها باید پیچیدگی روزافزون الگوریتم‌ها و سامانه‌های کوانتومی بزرگ‌تر را در نظر بگیرند.

ایجاد یک چارچوب محک استاندارد کوانتومی، مشابه با ارزیابی عملکرد استاندارد (SPEC) در محاسبات کلاسیک، می‌تواند پایه‌ای برای ارزیابی عملکرد سازگار و قابل‌اعتماد در محاسبات کوانتومی فراهم کند. چنین چارچوبی به هدایت توسعهٔ فناوری‌های کوانتومی کمک خواهد کرد و اطمینان حاصل خواهد کرد که ادعاهای عملکردی بر اساس محک‌های دقیق و قابل تکرار، استوار هستند.

چگونگی تولد رایانش کوانتومی و اهداف آن

ظهور رایانش کوانتومی به اوایل دهه ۱۹۸۰ بازمی‌گردد، زمانی که فیزیک‌دان، Richard Feynman، ایدهٔ استفاده از مکانیک کوانتومی برای انجام محاسبات را مطرح کرد. Feynman مشاهده کرد که رایانه‌های کلاسیک در شبیه‌سازی سامانه‌های کوانتومی با چالش‌های قابل‌توجهی مواجه هستند، زیرا پیچیدگی مرتبط با حالت‌های کوانتومی به‌صورت نمایی افزایش می‌یابد. این بینش منجر به پایه‌گذاری مفهومی تحت عنوان «رایانش کوانتومی» شد که هدف آن استفاده از پدیده‌های مکانیکی کوانتومی برای پردازش اطلاعات با کارایی بیش‌تری نسبت به رایانه‌های کلاسیک است. رایانش کوانتومی،

بهره‌برداری از اصول برهم‌نهی، درهم‌تنیدگی و تداخل کوانتومی برای حل مسائل پیچیده‌ای است که برای رایانه‌های کلاسیک غیرقابل‌حل هستند. رایانه‌های کوانتومی برای انجام وظایفی مانند فاکتورگیری اعداد بزرگ، جستجو در پایگاه‌های مه‌داده‌ها و شبیه‌سازی فرایندهای فیزیکی کوانتومی طراحی شده‌اند که با سرعت بسیار بیش‌تری نسبت به همتایان کلاسیک خود این فعالیت‌ها را انجام می‌دهند. این پتانسیل برای افزایش توان محاسباتی، باعث تحقیقات و توسعه مداوم در این زمینه شده است با این امید که پیشرفت‌های چشم‌گیری در حوزه‌های علمی و فناوری مختلف به دست آید [۱۳].

نقش شبیه‌سازهای مدار کوانتومی در پیشبرد رایانش کوانتومی

شبیه‌سازهای مدار کوانتومی نقش مهمی در پیشبرد رایانش کوانتومی دارند؛ زیرا بستری را برای طراحی، آزمایش و بهینه‌سازی الگوریتم‌های کوانتومی قبل از پیاده‌سازی آن‌ها روی سخت‌افزار واقعی کوانتومی فراهم می‌کنند. این شبیه‌سازها رفتار مدارهای کوانتومی را مدل‌سازی می‌کنند و به پژوهشگران این امکان را می‌دهند تا ویژگی‌ها و عملکرد الگوریتم‌های کوانتومی را تحت شرایط کنترل‌شده بررسی کنند. این قابلیت برای درک مزایای محاسباتی مکانیک کوانتومی و شناسایی مشکلات احتمالی در طراحی الگوریتم بسیار ضروری است [۱۳].

یکی از وظایف شبیه‌سازهای مدار کوانتومی این است که شبیه‌سازی سامانه‌های کوانتومی را که برای رایانه‌های کلاسیک بسیار پیچیده هستند، امکان‌پذیر کنند. با استفاده از شبیه‌سازهای مدار کوانتومی، پژوهشگران می‌توانند درک بهتری از دینامیک الگوریتم‌های کوانتومی پیدا کنند، راهبُردهای محاسباتی جدیدی را کشف کنند و رویکردهای خود را برای بهره‌برداری کامل از پتانسیل رایانش کوانتومی بهینه‌سازی کنند. این فرایند تکراری شبیه‌سازی و بهینه‌سازی برای توسعهٔ عملی فناوری‌های رایانش کوانتومی ضروری است و راه را برای سامانه‌های کوانتومی مقاوم‌تر و مقیاس‌پذیرتر در آینده، هموار می‌کند [۱۳].

پرسش‌های پژوهش

توضیح این پرسش‌ها برای واضح کردن رویهٔ پژوهش بوده است. تمامی پرسش‌های گفته‌شده برای محک کلی رایانه‌های کوانتومی در لایه‌های مختلف آن یعنی نرم‌افزار، میان‌افزار و سخت‌افزارها بوده است. از آن‌جایی که در پژوهش ما تمرکز بر محک شبیه‌سازهای مدار کوانتومی است، معیارها همان معیارهایی است که در فصل اول، مقدمه، بیان شده است چرا که بعضی معیارها زمانی باید مورداستفاده قرار بگیرند که تمامی قسمت‌های رایانه‌های کوانتومی اعم از نرم‌افزار، میان‌افزار و سخت‌افزار در اجرای این محک‌ها دخیل باشند.

معیارهای عملکرد کلیدی برای محک منصفانه‌ی شبیه‌سازهای مدار کوانتومی چیست؟

زمان شبیه‌سازی

این زمان، زمان کلی موردنیاز برای شبیه‌سازی یک مدار کوانتومی است. این معیار بسیار مهم است زیرا بازتابی از کارایی و سرعت شبیه‌ساز است. عواملی که بر زمان شبیه‌سازی تأثیر می‌گذارند شامل پیچیدگی مدار کوانتومی، تعداد کیوبیت‌ها و ماهیت الگوریتمی است که شبیه‌سازی می‌شود [۲،۱].

مصرف حافظه

اندازه‌گیری مقدار حافظه‌ای که برای انجام شبیه‌سازی موردنیاز است از دیگر معیارهای مهم به شمار می‌رود. این معیار به‌ویژه برای شبیه‌سازی‌های بزرگ‌مقیاس که محدودیت‌های حافظه می‌تواند به یک عامل محدودکننده تبدیل شود، اهمیت دارد. به دلیل رشد نمایی فضای حالت کوانتومی با افزایش تعداد کیوبیت‌ها، شبیه‌سازهای کوانتومی، اغلب نیاز به مدیریت مقادیر زیادی از داده‌ها دارند [۲].

قابلیت مقیاس‌پذیری

توانایی شبیه‌ساز برای مدیریت افزایش تعداد کیوبیت‌ها و عمق مدار[1]، قابلیت مقیاس‌پذیری نامیده می‌شود. عمق مدار، همان تعداد لایه‌هایی است که میزان پیچیدگی مدارهای کوانتومی را می‌سنجد. مقیاس‌پذیری برای کاربردهای عملی شبیه‌سازی کوانتومی، حیاتی است؛ زیرا تعیین می‌کند که شبیه‌ساز چگونه می‌تواند مشکلات واقعی با اندازه و پیچیدگی قابل‌توجه را مدیریت کند [۲،۱].

دقت

میزان دقتی که شبیه‌ساز می‌تواند رفتار یک سامانهٔ کوانتومی را تکرار کند نیز باید مورد محک قرار بگیرد. این، شامل چگونگی برخورد شبیه‌ساز با نویز کوانتومی، کاهش ناهمدوسی و سایر اثرات کوانتومی است که بر صحت نتایج شبیه‌سازی تأثیر بسزایی می‌گذارند [۲،۱].

انعطاف‌پذیری

از توانایی شبیه‌ساز برای پشتیبانی از الگوریتم‌های مختلف کوانتومی و انواع مختلف مدارهای کوانتومی به انعطاف‌پذیری یاد می‌شود. انعطاف‌پذیری شامل پشتیبانی از زبان‌های برنامه‌نویسی مختلف و ادغام با چارچوب‌های مختلف رایانش کوانتومی است [۱۱،۲،۱].

پیچیدگی پیاده‌سازی الگوریتم‌های کوانتومی

پیاده‌سازی الگوریتم‌ها در شبیه‌سازهای مختلف، متفاوت است، به‌گونه‌ای که هر کدام باتوجه‌به فنّاوری مورداستفاده مثل زبان برنامه‌نویسی، قابلیت اجرا بر روی سخت‌افزارهای خاص، دارای پیچیدگی‌های منحصربه‌فرد هستند. این معیار، ترکیبی از زمان یادگیری نحوهٔ استفاده از شبیه‌ساز و زمان پیاده‌سازی یک الگوریتم در آن است که باتوجه‌به این که به توانایی افراد وابسته است، یک معیار کیفی است؛ اما به دلیل تأثیر زیاد آن در روند پیاده‌سازی، لایق بررسی است.

[1] circuit depth

عملکرد شبیه‌سازهای مختلف بر روی الگوریتم‌های مختلف کوانتومی چگونه است؟

شبیه‌سازهای مختلف مدار کوانتومی، عملکرد متفاوتی بر روی الگوریتم‌های کوانتومی مختلف دارند. قسمت‌های زیر ویژگی‌های عملکردهای مشاهده شده را خلاصه می‌کنند.

شبیه‌سازهای مبتنی بر Schrödinger

این شبیه‌سازها که بر پایهٔ نمایش مستقیم بردار حالت، استوار هستند. به دلیل پیاده‌سازی ساده و کاربرد عمومی، به طور گسترده‌ای استفاده می‌شوند. بااین‌حال، آن‌ها اغلب در مقیاس‌بندی به بیش از یک تعداد خاص از کیوبیت‌ها به دلیل نیازهای حافظه‌ای نمایی با چالش‌هایی مواجه می‌شوند [۱۱].

شبیه‌سازهای مبتنی بر شبکهٔ تنسوری

این شبیه‌سازها در مدارهای کوانتومی با محدودیت درهم‌تنیدگی، خوب عمل می‌کنند. با نمایش حالت‌های کوانتومی به‌عنوان شبکه‌های تنسوری، می‌توانند برخی از دسته‌های مدارهای کوانتومی را که با روش‌های مبتنی بر Schrödinger غیرقابل‌حل هستند، به طور کارآمد شبیه‌سازی کنند. بااین‌حال، ممکن است با مدارهایی که شامل درجات بالایی از درهم‌تنیدگی هستند، دچار مشکل شوند [۱۱].

روش‌های ترکیبی

این روش‌ها، ترکیبی از فن‌های شبیه‌سازی مختلف را برای بهره‌برداری از نقاط قوت هر یک ترکیب می‌کنند. به‌عنوان‌مثال، ادغام شبکه‌های تنسوری با روش‌های مبتنی بر Schrödinger می‌تواند تعادلی بین استفاده از حافظه و کارایی زمانی ایجاد کند که آن‌ها را برای طیف وسیع‌تری از مدارهای کوانتومی مناسب می‌سازد [۱۱].

شبیه‌سازها با کاربرد خاص

شبیه‌سازهایی که برای انواع خاصی از مدارها یا الگوریتم‌های کوانتومی بهینه‌سازی شده‌اند، مانند مدارهای پایدارکننده[1] یا الگوریتم‌های کوانتومی متغیر[2]، در دامنه‌های مربوط به خود، عملکرد بهتری نسبت به شبیه‌سازهای عمومی دارند. این شبیه‌سازهای خاص می‌توانند به طور قابل‌توجهی زمان شبیه‌سازی و نیازهای منابع را برای کاربردهای هدف خود، کاهش دهند [۱۱].

شبیه‌سازها با سخت‌افزار خاص

شبیه‌سازهایی که برای بهره‌برداری از ویژگی‌های خاص سخت‌افزاری، مانند پردازنده‌های گرافیکی، طراحی شده‌اند، می‌توانند به بهبودهای قابل‌توجهی در عملکرد دست یابند. این شبیه‌سازها از قابلیت‌های پردازش موازی برای مدیریت سامانه‌های کوانتومی بزرگ‌تر با کارایی بیشتر نسبت به شبیه‌سازهای مبتنی بر CPU سنتی استفاده می‌کنند [۱۱،۱۴].

نقاط قوت و ضعف شبیه‌سازهای فعلی در چه قسمت‌هایی است؟

نقاط قوت

دقت بالا: بسیاری از شبیه‌سازهای فعلی، شبیه‌سازی‌هایی بادقت بالا ارائه می‌دهند که برای آزمایش و اعتبارسنجی الگوریتم‌های کوانتومی ضروری است. آن‌ها می‌توانند رفتار سامانه‌های کوانتومی را به طور دقیقی تکرار کنند که برای تحقیقات و توسعه در رایانش کوانتومی بسیار مهم است [۱۱].

عمومیت: شبیه‌سازهای مدرن از طیف وسیعی از الگوریتم‌ها و کاربردهای کوانتومی پشتیبانی می‌کنند، از گیت‌های کوانتومی پایه تا کدهای پیچیده تصحیح خطای کوانتومی. این عمومیت، آن‌ها را به ابزارهای ارزشمندی برای حوزه‌های مختلف تحقیقات رایانش کوانتومی تبدیل کرده است [۱۴].

[1] Stabilizer Circuits

[2] Variational Quantum Algorithms (VQA)

ویژگی‌های پیشرفته: برخی از شبیه‌سازها ویژگی‌های پیشرفته‌ای مانند مدل‌سازی نویز، شبیه‌سازی تصحیح خطا و پشتیبانی از الگوریتم‌های کوانتومی متغیر را ارائه می‌دهند که برای شبیه‌سازی‌های واقع‌گرایانه و توسعه رایانش کوانتومی مقاوم به خطا، ضروری هستند [۱].

نقاط ضعف

مشکلات مقیاس‌پذیری: بسیاری از شبیه‌سازها با مقیاس‌بندی به تعداد بیش‌تری از کیوبیت‌ها به دلیل افزایش نمایی در نیازهای حافظه و محاسبات، با چالش‌هایی مواجه هستند. این محدودیت‌ها استفاده آن‌ها را برای سامانه‌های کوانتومی بزرگ‌تر و مسائل واقعی محدود می‌کند [۱۴].

مصرف منابع زیاد: استفاده بالا از حافظه و زمان طولانی شبیه‌سازی چالش‌های رایجی هستند. شبیه‌سازی مدارهای بزرگ کوانتومی اغلب به منابع محاسباتی قابل‌توجهی نیاز دارد که برای بسیاری از کاربران بدون دسترسی به امکانات محاسباتی قوی، عملی نیست [۱۱].

محدودیت‌های تخصصی: درحالی‌که شبیه‌سازهای خاص در حوزه‌های خود عالی هستند، اغلب فاقد انعطاف‌پذیری برای مدیریت انواع مختلف مدارهای کوانتومی هستند. این نیاز به استفاده از چندین شبیه‌ساز برای پوشش جنبه‌های مختلف رایانش کوانتومی را ایجاد می‌کند که می‌تواند فرایند شبیه‌سازی را بسیار زمان‌بر و پیچیده کند [۱].

پیچیدگی: پیچیدگی راه‌اندازی و اجرای شبیه‌سازی‌ها، می‌تواند مانعی برای ورود کاربران جدید باشد. دانش دقیقی از اصول رایانش کوانتومی و ساختار خاص شبیه‌ساز اغلب برای دستیابی به عملکرد بهینه، موردنیاز است [۱].

فصل سوم

پژوهش

مطالعات پیشین

بعد از طرح پرسش‌های پژوهش و کسب دانش و آشنایی با ادبیات حوزه، بدیهی است، گام بعدی یافتن مطالعات پیشین و تحقیق دربارهٔ آن‌ها است. همان‌طور که بحث شد، حوزهٔ رایانش کوانتومی، نوظهور و نوپا است. محک شبیه‌سازها نیز به طبع آن در مراحل اولیهٔ خود قرار دارد و مطابق آنچه که انتظار می‌رفت، مطالعات انجام‌شده از نظر کمی غنی نمی‌باشد. با تمام این تفاسیر، تمامی پژوهش‌های مطالعه‌شده، به ترتیب اهمیت و داشتن شباهت با پژوهش ما، در ادامه بررسی و نقد شده‌اند.

محک نرم‌افزارهای شبیه‌سازی رایانه‌های کوانتومی [۱]

در این مقاله، انتخاب الگوریتم‌ها برای محک شبیه‌سازها بر اساس پژوهش [۱۵] بوده است که برای پژوهش ما نیز سودمند واقع شده است. تحلیل‌های ارائه‌شده در این پژوهش محدود به شبیه‌سازهای بردار حالت است. در پژوهش ما هم به همین‌گونه است؛ اما با بهره‌گیری از [۱۱]، سعی در آن بوده است که از دسته‌های مختلف شبیه‌سازی برای انجام محک انتخاب شود.

در این پژوهش [۱]، انتخاب شبیه‌سازها بر چند اساس استوار است. اول این که آیا می‌توانند قدرت محاسباتی ارائه‌شده توسط سامانه‌های HPC را بهره‌برداری کنند یا خیر. دوم این که آیا به طور فعال نگهداری و توسعه داده می‌شوند یا خیر و سوم این که دسترسی به مستندات مربوطه (مثال‌ها و آموزش‌های مورداستفاده) فراهم شده است یا خیر. درصورتی‌که انتخاب شبیه‌سازهای پژوهش ما بر اساس دسترسی به آن‌ها از لحاظ متن‌باز بودن، رویکرد شبیه‌سازی، امکان استفاده بر روی رایانه‌های شخصی و مانند این پژوهش [۱] در دسترس بودن مستندات بوده است.

در این پژوهش [۱]، تمرکز اصلی بر محک شبیه‌سازهای سازگار با HPC است. در پژوهش ما، تأکید اصلی بر محک شبیه‌سازهای مدار کوانتومی بهینه‌سازی شده برای رایانه‌های شخصی که توسط کاربران روزمره استفاده می‌شوند، است. بااین‌حال، باید توجه داشت که شبیه‌سازی مدارهایی با بیش از ۳۰ کیوبیت بر روی رایانه شخصی با ۱۶ گیگابایت حافظه، تقریباً غیرممکن است.

تفاوت‌های مجموعهٔ دستورات سطح بالاتر در میان شبیه‌سازها، چالش مهمی را به وجود می‌آورد. هر شبیه‌ساز نیازمند ترجمهٔ منطق الگوریتمی یکسان به مجموعه‌ای متمایز از دستورات و متغیرها است. این فرایند دستی توسط سطح انتزاع ریاضی شبیه‌ساز، محدود شده و زمان‌بر و مستعد خطا است. در نتیجه، خطر ترجمه نادرست دستورات از الگوریتم کوانتومی اصلی وجود دارد. این چالش بین تحقیق ما و این پژوهش [۱] مشترک است.

محک ایده‌آل مستلزم تعاریف یکسان در میان شبیه‌سازها و حتی نسخه‌های مختلف یک شبیه‌ساز است که به دلیل تغییرات زیاد که زادهٔ نوپا بودن این حوزه است، چالش‌های زیادی را در پی دارد. این چالش‌ها، در میان تمامی پژوهش‌ها مشترک هستند.

چالش دیگری که از تعارضات در وابستگی‌های شبیه‌سازها به دیگر نرم‌افزارها ناشی می‌شود، زمانی است که شبیه‌سازهای خاص نیازمند نسخه‌های مشخصی از دیگر نرم‌افزارها هستند. نصب هم‌زمان همهٔ شبیه‌سازها زمانی که چندین شبیه‌ساز یک وابستگی مشترک دارند، غیرعملی می‌شود. در این

پژوهش [۱] از Singularity و Charliecloud به‌عنوان ابزارهایی برای رفع این مشکل، استفاده شده است. بااین‌حال، باتوجه‌به نیازهای کاربردی‌تر و تفاوت پژوهش ما در این که شبیه‌سازهایی که در رایانه‌های شخصی قابل‌اجرا باشند، مدنظر است، ما از ابزارهای دیگری مانند Docker استفاده کرده‌ایم.

به دلیل محدودیت منابع سخت‌افزاری، ما تأثیر استفاده از GPU را ارزیابی نکرده‌ایم درحالی‌که در این پژوهش [۱] این تأثیر به طور گسترده بحث‌وبررسی شده است.

یکی از ویژگی‌های مهم توسعه‌یافته در این پژوهش [۱]، ترجمهٔ مجموعهٔ دستورات سطح بالای تبدیل‌شده به OpenQASM یک الگوریتم کوانتومی به مجموعه دستورات خاص شبیه‌ساز انتخاب شده است. در تحقیق ما، این ویژگی موردبحث قرار گرفته؛ اما تصمیم به تأخیر در پیاده‌سازی آن برای کارهای آینده گرفته شده است. اگر هدف ما محک کل شبیه‌ساز کوانتومی به‌عنوان یک جزء باشد، باید یک مترجم طراحی شود که قادر به تبدیل اسکریپت‌های OpenQASM به اسکریپت‌های شبیه‌ساز هدف، مانند Cirq یا Qiskit باشد، تقریباً مشابه آن‌چه که در این پژوهش [۱] وجود دارد؛ اما با این تفاوت که این ترجمه به‌جای این که توسط یک انسان مهندسی شود، توسط یک واحد نرم‌افزاری انجام شود.

در این پژوهش [۱]، معیارهایی شامل زمان و مصرف حافظه مطرح شده است. مصرف حافظه، استخراج نشده است؛ زیرا استخراج استفادهٔ واقعی از منابع حافظه‌ای در تنظیمات HPC چندهسته‌ای و چندگره‌ای غیرقابل‌انجام یا حداقل بسیار دشوار می‌شود. بااین‌وجود، ما هر دو ویژگی عملکردی ذکر شده را در نظر می‌گیریم. ویژگی‌های دیگر که می‌توان در نظر گرفت شامل سهولت استفاده، مقیاس‌پذیری و همگرایی برای الگوریتم‌های خاص است که در این پژوهش [۱] لحاظ نشده است.

سخت‌افزار معرفی شده در این پژوهش [۱]، شامل یک خوشهٔ HPC محلی با محدودیت‌هایی نظیر ۲۳ ساعت با محدودیت حافظه ۳۰۰ گیگابایت برای اجرای CPU و ۳۲۰ گیگابایت حافظه برای GPU به‌ازای ۹۰۰ گیگابایت حافظه CPU برای اجراهایی که از دو واحد GPU و CPU بهره می‌برند.

در مقابل، سخت‌افزار ما شامل یک رایانه شخصی استاندارد و به نسبت بسیار ضعیف‌تر با ۴ هسته بوده است که قادر به اجرای ۱ ساعت با محدودیت حافظهٔ ۱۶ گیگابایت برای اجرای CPU است. همچنین، سخت‌افزار ما فاقد GPU بوده است.

الگوریتم‌های محک انتخاب شده در این پژوهش [۱] شامل دینامیک مدل XYZ-Heisenberg، مدارهای کوانتومی تصادفی (RQC) و تبدیل فوریه کوانتومی (QFT) است. در تحقیق ما، الگوریتم‌هایی مانند Deutsch-Jozsa (DJ)، Bernstein-Vaziran (BV)، QFT و Simon انتخاب شده‌اند چرا که مثال‌های شبیه‌تری به مسائل دنیای واقعی هستند.

همان‌طور که در این پژوهش [۱] ذکر شده، انتظار می‌رود که پیاده‌سازی بهینه با افزایش مقیاس ورودی الگوریتم‌ها به‌گونه‌ای باشد که رفتاری متناسب از خود نشان دهد. به طرز شگفت‌انگیزی، این رفتار مورد انتظار تنها برای اعداد نسبتاً بزرگ کیوبیت مشاهده می‌شود. برای تعداد کیوبیت‌های کوچک‌تر، زمان، وابستگی بسیار ضعیف‌تری به اندازهٔ سامانه نشان می‌دهد و حتی در برخی موارد خاص می‌تواند تقریباً مستقل از اندازهٔ سامانه باشد. این پدیده ممکن است به تحلیل داخلی مدار و بهینه‌سازی که توسط برخی شبیه‌سازها انجام می‌شود، نسبت داده شود. در تحقیق ما نیز این پدیده مشاهده شده است.

در پژوهش ما و این پژوهش [۱] محدودیت‌هایی در سه دسته شناسایی شده است: طراحی، زمان و حافظه. محدودیت‌های طراحی به محدودیت‌هایی اشاره دارد که توسط تصمیمات طراحی شبیه‌سازها تحمیل می‌شود و توانایی آن را برای پردازش سامانه‌هایی فراتر از اندازه خاص محدود می‌کند. محدودیت‌های زمانی و حافظه به محدودیت‌هایی اشاره دارد که ناشی از محدودیت‌های منابع یا محدودیت‌هایی است که خود محققان برای صورت‌بخشی عملی به آزمایش‌ها اعمال کرده‌اند.

بررسی معیارهای عملکرد کاربردی برای محاسبات کوانتومی [۲]

در مقالهٔ مرور شده، یک مجموعه معیارهای متن‌باز برای برنامه‌های کاربردی کوانتومی معرفی شده است. این مجموعه قادر است رایانه‌های کوانتومی واقعی و شبیه‌سازها را با برنامه‌ها و الگوریتم‌های

کوچک که شبیه‌سازی آن‌ها به‌صورت کلاسیک به‌صورت نمایی پیچیده نیست، محک بزند. این معیارها بر اساس دو اندازه‌گیری زمان و Fidelity استوار هستند. هر رایانهٔ کوانتومی یا شبیه‌سازی که در زمان کمتر به نتیجه موردنظر برسد، بهتر عمل کرده است. به همین ترتیب، اگر خروجی هر شبیه‌ساز یا رایانه کوانتومی دارای Fidelity بیش‌تری از نتایج باشد، کار بهتری انجام داده است. Fidelity نوعی اندازه‌گیری است که نزدیکی دو توزیع احتمالی را کمی می‌کند. از آن برای اندازه‌گیری فاصلهٔ بین خروجی رایانهٔ کوانتومی یا شبیه‌ساز و خروجی ایده‌آل استفاده می‌شود.

از مزایای این پژوهش [۲]، دسترسی رایگان به پیاده‌سازی‌ها و ابزارهای محک است. مزیت دیگر، توضیح کامل مسائل و تعاریف ارائه‌شده در متن و ضمیمهٔ مقاله است که در پژوهش ما برای پیاده‌سازی محک‌ها از آن‌ها بهره‌گیری شده است.

از معایب آن توجه بیش‌تر این پژوهش [۲] به اندازه‌گیری‌های فیزیکی برای بهبود سخت‌افزار کوانتومی است و کمتر به نرم‌افزارها و شبیه‌سازها توجه شده است. این در حالی است که عیب اصلی این تحقیق، عدم تمرکز در محک تعداد بیش‌تری از شبیه‌سازها است. مشکل دیگر ذکر شده در مقاله، روش بسیار ساده و پایه‌ای برای اندازه‌گیری زمان اجرا است که می‌تواند بسیار دقیق‌تر باشد و جنبه‌های مختلف شبیه‌ساز و لایه‌های نرم‌افزاری آن را بررسی کند.

در این پژوهش [۲]، الگوریتم‌ها برای محک با این محدودیت انتخاب شده‌اند که شبیه‌سازی آن‌ها به‌صورت نمایی هزینه‌بر نباشد که این خود ممکن است باعث گمراهی در روند سنجش و محک شبیه‌سازها شود.

یک مجموعه معیار QASM سطح پایین برای ارزیابی و شبیه‌سازی NISQ [۳]

در این پژوهش [۳]، مجموعهٔ معیار طراحی شده را به طور خاص برای ارزیابی و شبیه‌سازی محاسبات کوانتومی (QC) در دورهٔ میانی کوانتومی پر از نویز (NISQ) معرفی می‌کند. این مجموعه که QASMBench نامیده می‌شود، بر اساس زبان اسمبلی OpenQASM است و شامل مجموعه‌ای

از الگوریتم‌ها و روال‌های کوانتومی از حوزه‌های مختلف مانند شیمی کوانتومی، جبر خطی و یادگیری ماشین است. هدف QASMBench ارائهٔ یک مجموعهٔ معیار استاندارد و آسان برای محک است که می‌تواند برای ارزیابی عملکرد دستگاه‌های NISQ، کامپایلرهای کوانتومی و شبیه‌سازها استفاده شود. QASMBench یک مجموعه ابزار جامع است که برای ارزیابی قابلیت‌ها و محدودیت‌های سامانه‌های کوانتومی فعلی با تمرکز بر زبان اسمبلی کوانتومی سطح پایین، طراحی شده است. این مجموعه شامل معیارهایی است که نمایانگر عملیات کوانتومی رایج هستند و بر روی چندین سامانه مانند IBM-Q، Rigetti و IonQ آزمایش می‌شوند. این مجموعه همچنین چندین معیار از جمله تراکم گیت، طول عمر نگهداری، تراکم اندازه‌گیری و واریانس درهم‌تنیدگی را معرفی می‌کند تا بینش‌های عمیق‌تری از اجرای مدارهای کوانتومی ارائه دهد.

حاصل این پژوهش [۳]، شامل مجموعه‌ای گسترده از الگوریتم‌های کوانتومی از چندین حوزه است که آن را به ابزاری همه‌کاره برای ارزیابی جنبه‌های مختلف سامانه‌های کوانتومی تبدیل می‌کند. با مبنا قراردادن معیارها بر اساس OpenQASM، این مجموعه یک رویکرد استاندارد فراهم می‌کند که می‌تواند در پلتفرم‌های مختلف کوانتومی استفاده شود.

با تمامی نقاط قوتی که این پژوهش [۳] دارد، ماهیت جامع این مجموعه و گنجاندن معیارهای دقیق ممکن است استفاده کامل از قابلیت‌های آن را برای تازه‌کاران در محاسبات کوانتومی دشوار کند. به‌علاوه، عملکرد معیارها می‌تواند به‌شدت به سخت‌افزار مورداستفاده وابسته باشد که ممکن است تعمیم‌پذیری نتایج را در سامانه‌های کوانتومی مختلف محدود کند. درحالی‌که QASMBench طیف گسترده‌ای از اندازه‌های مدار را پوشش می‌دهد، اثربخشی آن ممکن است با افزایش مقیاس و پیچیدگی سامانه‌های کوانتومی آینده محدود شود.

با این که این پژوهش [۳] به نظر جامع می‌رسد، اما رویکرد آن برای محک شبیه‌سازها با رویکرد پژوهش ما متفاوت است. در این پژوهش [۳]، آن دسته از لایه‌های نرم‌افزار که کاربر با آن‌ها در تعامل است، واحدهایی نظیر مترجم یا مفسر زبان سطح بالا به زبان سطح پایین، نادیده گرفته می‌شوند. در واقع

فقط الگوریتم‌های ترجمه شده به زبان ماشین اجرا شده و در محک‌ها زمان تبدیل الگوریتم‌ها به کدهای ماشین به طور کلی حذف شده است.

دیگر پژوهش‌ها

پژوهش‌های دیگر در این زمینه با رویکردهای متفاوت انجام شده است. در تعدادی از آن‌ها نظیر [۱۶]، سعی در آن بوده که الگوریتم‌های خاصی را برای محک انتخاب کنند چرا که جنبهٔ خاصی از شبیه‌سازی برای آن‌ها مطرح بوده است. در تعدادی دیگر نظیر [۱۷، ۱۸]، سخت‌افزارهای خاص مدنظر بوده است. رویکردهای دیگر برای ساده‌سازی مسئله و یا هدف خاصی که به دنبال آن بوده‌اند از جامعیت‌بخشی به محک‌های آزمایش‌شده، خودداری کرده‌اند که این برخلاف رویکردی است که در پژوهش ما مدنظر بوده است.

همان‌طور که مشخص است، با این که فعالیتی که در پژوهش ما انجام شده است با دیگران شباهت‌هایی دارد؛ اما به طور دقیق، محک شبیه‌سازهایی که بر روی رایانه‌های استاندارد شخصی قابل‌اجرا باشند و در عین حال این محک‌ها از الگوریتم‌هایی انتخاب‌شده باشند که به مسائل دنیای واقعی نزدیک‌تر باشند، در هیچ پژوهشی مشاهده نشده است. در واقع تمامی پژوهش‌ها، محک‌ها را روی ابررایانه‌ها و رایانه‌های پردازش سریع بررسی کرده‌اند که تفاوت محسوسی در رویکرد ما و دیگران ایجاد می‌کند.

انتخاب شبیه‌سازها

همان‌طور که در قبل اشاره شد، شبیه‌سازها مبتنی بر این که از چه روشی برای انجام شبیه‌سازی استفاده می‌کنند را می‌توان دسته‌بندی کرد. باتوجه‌به پژوهش [۱۱]، سعی شده است برای محک منصفانه‌تر از هر دستهٔ شبیه‌سازها، یک شبیه‌ساز انتخاب شود که هر دسته برای شرکت در محک حداقل یک نماینده داشته باشد. باید به این نکته توجه داشت که بعضی دسته‌ها دارای هیچ شبیه‌ساز پیاده‌سازی شده‌ای

نیستند و در بعضی دسته‌ها انجام پیاده‌سازی زمانی فراتر از زمان در اختیار پژوهش ما نیاز داشته‌اند.

باتوجه‌به توضیحات داده شده و دیگر شرایط مثل عدم دسترسی به نسخهٔ نهایی، عدم وجود راهنماها و سندهای استفاده از شبیه‌سازها که به توسعه‌دهندگان شبیه‌سازها مربوط می‌شود، شبیه‌سازهای زیر برای محک انتخاب شده‌اند:

- شبیه‌ساز Aer، زیرمجموعهٔ سامانهٔ Qiskit
- شبیه‌ساز QASM، زیرمجموعهٔ سامانهٔ Qiskit
- شبیه‌ساز QASM، زیرمجموعهٔ سامانهٔ DDSIM
- شبیه‌ساز Hybrid QASM، زیرمجموعهٔ سامانهٔ DDSIM
- شبیه‌ساز Pure، زیرمجموعهٔ سامانهٔ Cirq
- شبیه‌ساز QSimCirq، زیرمجموعهٔ سامانهٔ Cirq

همان‌طور که اشاره شد، از آن‌جایی که توسعهٔ این شبیه‌سازها هنوز به بلوغ کامل نرسیده است، استفاده از آن‌ها در پیاده‌سازی تمامی الگوریتم‌ها دشوار و گاهی اوقات غیرممکن است. برای مثال چالشی که در طول پژوهش به آن برخورد کردیم، پیاده‌سازی الگوریتم‌هایی که به ذخیرهٔ اندازه‌گیری‌ها در طول آن نیاز دارند در شبیه‌ساز Hybrid QASM (مورد چهارم) انجام نشد. دلیل آن هم عدم توانایی این شبیه‌ساز در ارائهٔ ویژگی یادشده است. اما کار ما در این‌جا پایان نیافت و با مکاتبه‌ای که با توسعه‌دهندگان این شبیه‌ساز انجام شد، قول پیاده‌سازی این ویژگی داده شد. با استناد بر مکاتبه[1]، تا زمان نگارش این نوشته، ویژگی یادشده تکمیل و پیاده‌سازی نشده است.

انتخاب الگوریتم‌های محک

در این بخش سعی شده است الگوریتم‌هایی برای محک شبیه‌سازها انتخاب شود که از شهرت بیش‌تری برخوردار هستند و در نتیجه کاربرد بیش‌تری نسبت به دیگر الگوریتم‌ها دارند. به‌علاوه، بعضی الگوریتم‌ها

[1] https://github.com/cda-tum/mqt-ddsim/issues/326

نظیر Quantum Fourier Transform که پایه‌ای برای دیگر الگوریتم‌ها هستند نیز انتخاب شده‌اند. درصورتی‌که شبیه‌سازها در اجرای این الگوریتم‌ها به‌خوبی عمل کنند می‌توان نتیجه گرفت که در حل بسیاری از مسائل دارای عملکرد خوبی هستند.

Deutsch-Jozsa

الگوریتم Deutsch-Jozsa یکی از اولین الگوریتم‌های کوانتومی است که قدرت محاسبات کوانتومی را نسبت به روش‌های کلاسیک نشان می‌دهد. این الگوریتم با استفاده از تعداد ورودی‌های بسیار کمتری نسبت به هر الگوریتم کلاسیک، تعیین می‌کند که آیا یک تابع دودویی داده‌شده مثل f، ثابت است (برای همه ورودی‌ها خروجی یکسانی تولید می‌کند) یا متعادل است (برای نیمی از ورودی‌ها خروجی ۰ و برای نیمه دیگر خروجی ۱ تولید می‌کند). در الگوریتم کلاسیک، برای تعیین این که آیا یک تابع ثابت یا متعادل است، ممکن است نیاز باشد، در بدترین حالت، به تابع $۲^{n-۱}+۱$ بار ورودی بدهیم تا خروجی آن را مشخص کنیم. اما الگوریتم Deutsch-Jozsa این مسئله را تنها با یک‌بار ورودی دادن به Oracle کوانتومی حل می‌کند که این نشان‌دهندهٔ یک کاهش پیچیدگی نمایی است.

پیاده‌سازی

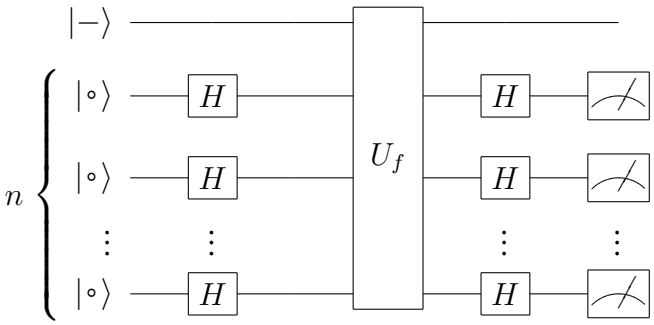
الگوریتم Deutsch-Jozsa روی یک رایانه یا شبیه‌ساز کوانتومی به‌صورت زیر پیاده‌سازی می‌شود:

- **مقداردهی اولیه:** با n کیوبیت در حالت $|۰\rangle$ و یک کیوبیت اضافی (کیوبیت جواب) در حالت $|۱\rangle$ شروع می‌کنیم. سپس کیوبیت جواب با استفاده از گیت Hadamard به حالت $|-\rangle$ تبدیل می‌کنیم.
- **برهم‌نهی:** گیت Hadamard را به اولین n کیوبیت اعمال کرده تا یک برهم‌نهی از تمام حالات ورودی ممکن ایجاد شود. حالت سامانه به یک برهم‌نهی یکنواخت از تمام ۲^n حالت ورودی ممکن تبدیل می‌شود.

- **پرسش از Oracle:** Oracle کوانتومی U_f را اعمال کرده که تابع f را درون خودش حفظ می‌کند با این تفاوت که فاز دامنهٔ مربوط به ورودی‌هایی که در آن $f(x) = ۱$ است را تغییر می‌دهد.
- **تداخل:** دوباره گیت Hadamard را به تمام n کیوبیت اعمال می‌کنیم. این مرحله باعث تداخل سازنده بین دامنه‌ها می‌شود و دامنه جواب صحیح را تقویت می‌کند.
- **اندازه‌گیری:** اولین n کیوبیت را اندازه‌گیری می‌کنیم. اگر نتیجه اندازه‌گیری $|۰\rangle^{\otimes n}$ باشد، تابع ثابت است؛ در غیر این صورت، تابع متعادل است.

مدار کوانتومی

در زیر نمودار مدار کوانتومی برای ورودی با n کیوبیت رسم شده است.



شکل ۱.۳: نمودار مدار کوانتومی الگوریتم Deutsch-Jozsa

چالش‌ها

چالشی که در پیاده‌سازی تمامی الگوریتم‌ها مشترک بوده است، پیاده‌سازی آن‌ها برای شبیه‌سازهای مختلف است. برای پیاده‌سازی هر الگوریتم در یک شبیه‌ساز، باید چارچوب‌های آن شبیه‌ساز رعایت شود. به‌عنوان‌مثال، با این که چارچوب پیاده‌سازی در سامانهٔ Qiskit با سامانهٔ Cirq متفاوت است اما

باید سعی شود که پیاده‌سازی‌ها تا جایی که ممکن است برای رعایت انصاف، مشابه هم باشد. رعایت این نکته کاری زمان‌بر و مشکل است که عمدهٔ زمان پژوهش را به خودش اختصاص داده است. در کل، به طور خاص برای این الگوریتم، چالشی وجود نداشت.

Bernstein-Vazirani

الگوریتم Bernstein-Vazirani یک الگوریتم کوانتومی است که مسئلهٔ یافتن یک‌رشتهٔ مخفی کدگذاری شده در یک تابع را به طور کارآمد حل می‌کند. با داشتن تابع $f(x)$ که حاصل‌ضرب نقطه‌ای یک‌رشته دودویی ناشناخته s و ورودی x را خروجی می‌دهد، هدف یافتن رشته s است. به‌عبارت‌دیگر می‌توان نوشت، $f(x) = s \cdot x$ که ما به دنبال یافتن s هستیم. درحالی‌که یک روش کلاسیک به n بار ورودی‌دادن به Oracle برای یک‌رشته به طول n نیاز دارد، الگوریتم Bernstein-Vazirani می‌تواند رشته s را تنها با یک‌بار ورودی‌دادن، پیدا کند که این یک کاهش پیچیدگی چندجمله‌ای را در پی دارد.

پیاده‌سازی

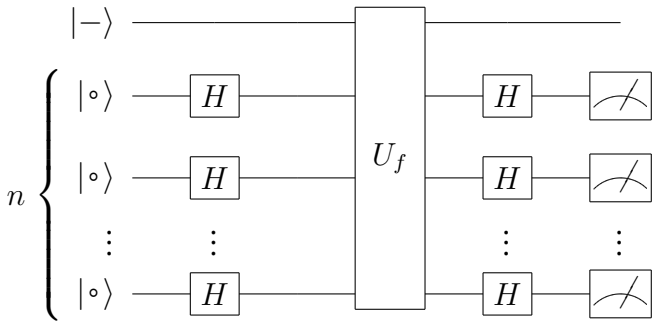
الگوریتم Bernstein-Vazirani روندی مشابه الگوریتم Deutsch-Jozsa را دنبال می‌کند:

- **مقداردهی اولیه:** با n کیوبیت در حالت $|0\rangle$ و یک کیوبیت اضافی (کیوبیت جواب) در حالت $|1\rangle$ شروع می‌کنیم. سپس کیوبیت جواب با استفاده از گیت Hadamard به حالت $|-\rangle$ تبدیل می‌کنیم.
- **برهم‌نهی:** گیت Hadamard را به اولین n کیوبیت اعمال می‌کنیم تا یک برهم‌نهی از تمام حالات ورودی ممکن ایجاد شود. این عمل هر کیوبیت را به حالتی تبدیل می‌کند که برهم‌نهی ۰ و ۱ را نشان می‌دهد.
- **پرسش از اوراکل:** Oracle U_f را اعمال می‌کنیم که تابع $f(x) = s \cdot x$ را محاسبه کرده و نتیجه را در کیوبیت جواب کدگذاری می‌کند. این مرحله عملاً حالات مربوط به مقادیر صحیح

s را نشانه‌گذاری می‌کند تا یافتن آن‌ها در زمان اندازه‌گیری آسان شود.

- **تداخل:** دوباره گیت Hadamard را به تمام n کیوبیت اعمال می‌کنیم. این باعث تداخل سازنده و مخرب بین حالات کوانتومی می‌شود و سامانه را در حالتی باقی می‌گذارد که مستقیماً رشتهٔ مخفی s را بعد از اندازه‌گیری نشان دهد.
- **اندازه‌گیری:** اولین n کیوبیت را اندازه‌گیری می‌کنیم که نتیجه، رشتهٔ مخفی s خواهد بود.

مدار کوانتومی



شکل ۲.۳: نمودار مدار کوانتومی الگوریتم Bernstein–Vazirani

چالش‌ها

چالش‌های مربوط به پیاده‌سازی این الگوریتم هم مانند Deutsch–Jozsa است و تفاوت زیادی بین آن‌ها وجود ندارد.

Simon

الگوریتم Simon یک الگوریتم کوانتومی است که مسئله‌ای خاص به نام مسئلهٔ Simon را به طور نمایی سریع‌تر از هر الگوریتم کلاسیک شناخته‌شده‌ای، حل می‌کند. این مسئله شامل یافتن یک‌رشتهٔ مخفی s است به‌طوری‌که برای یک تابع داده‌شده f، $f(x) = f(y)$ اگر و تنها اگر $x \oplus y = s$ باشد.

این الگوریتم مهم است؛ زیرا اولین الگوریتمی بود که کاهش پیچیدگی نمایی الگوریتم‌های کوانتومی نسبت به الگوریتم‌های کلاسیک را نشان داد و قدرت بالقوه محاسبات کوانتومی را برجسته کرد.

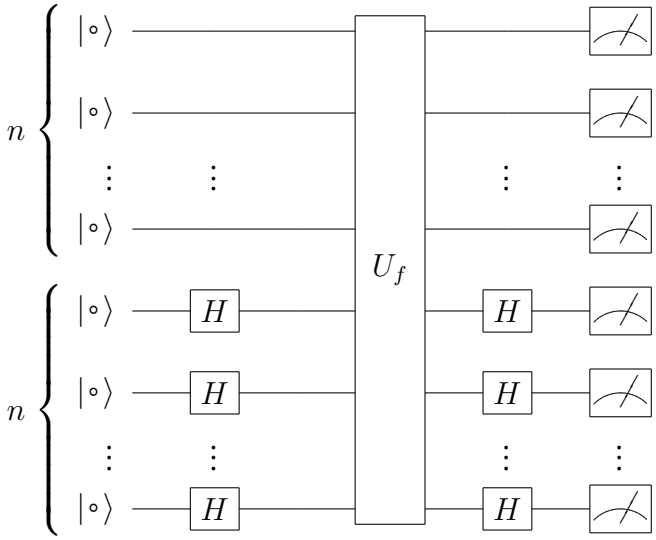
پیاده‌سازی

پیاده‌سازی الگوریتم Simon شامل مراحل زیر است:

- **مقداردهی اولیه:** در ابتدا n کیوبیت را در حالت $|0\rangle$ (نمایندهٔ Register ورودی) و به همین صورت، n کیوبیت دیگر را در حالت $|0\rangle$ (نماینده Register خروجی) مقداردهی اولیه می‌کنیم.
- **برهم‌نهی:** گیت Hadamard را به کیوبیت‌های ورودی اعمال می‌کنیم تا یک برهم‌نهی از تمام ورودی‌های ممکن ایجاد شود.
- **پرسش از Oracle:** Oracle U_f را اعمال می‌کنیم که $|x\rangle|y\rangle$ را به $|x\rangle|y \oplus f(x)\rangle$ نگاشت می‌کند. Oracle به‌گونه‌ای طراحی شده که $f(x) = f(y)$ شود، زمانی که $x \oplus y = s$ باشد.
- **تداخل:** دوباره گیت Hadamard را به کیوبیت‌های ورودی اعمال می‌کنیم. این مرحله از تداخل کوانتومی استفاده می‌کند تا اطلاعات موردنظر در مورد رشتهٔ مخفی s را استخراج کند.
- **اندازه‌گیری:** کیوبیت‌های ورودی را اندازه‌گیری می‌کنیم. نتایج اندازه‌گیری رشته‌های بیتی z را می‌دهند که $s \cdot z = 0$ (در مبنای ۲) را برقرار می‌کنند. با تکرار الگوریتم با پیچیدگی $O(n)$ بار، اطلاعات کافی برای تعیین s جمع‌آوری می‌شود.

مدار کوانتومی

در ادامه، شکل ۳.۳ نمودار مدار کوانتومی برای ورودی با n کیوبیت را نشان می‌دهد.



شکل ۳.۳: نمودار مدار کوانتومی الگوریتم Simon

چالش‌ها

چالش‌های پیاده‌سازی این الگوریتم هم تا حد زیادی مانند دیگر الگوریتم‌ها بوده است. به‌علاوه، چون ماهیت این الگوریتم تکرار اجرای آن و جمع‌آوری خروجی‌های هر اجرا است، همان‌طور که در قبل هم اشاره شد، بعضی شبیه‌سازها نظیر DDSIM قابلیت پیاده‌سازی آن به صورتی که قابل‌مقایسه با دیگر شبیه‌سازها باشد را نداشتند.

Quantum Fourier Transform (QFT)

Quantum Fourier Transform (QFT) معادل کوانتومی تبدیل فوریهٔ گسسته کلاسیک است. این تبدیل یک جزء اساسی در بسیاری از الگوریتم‌های کوانتومی، از جمله الگوریتم Shor برای تجزیه اعداد بزرگ است. QFT یک حالت کوانتومی را به حوزهٔ بَسامد آن تبدیل می‌کند، همان‌طور که تبدیل فوریهٔ کلاسیک برای یک سیگنال انجام می‌دهد. برخلاف معادل کلاسیک خود که برای مجموعه داده‌ای به‌اندازه N به $O(N \log N)$ عملیات نیاز دارد، QFT را می‌توان با استفاده از گیت‌های کوانتومی برای

پیاده‌سازی راه‌حل با پیچیدگی $O(\log^2 N)$ پیاده‌سازی کرد که یک افزایش سرعت نمایی در پیچیدگی مدار ارائه می‌دهد.

پیاده‌سازی

الگوریتم QFT روی یک رایانهٔ کوانتومی با استفاده از مراحل زیر پیاده‌سازی می‌شود:

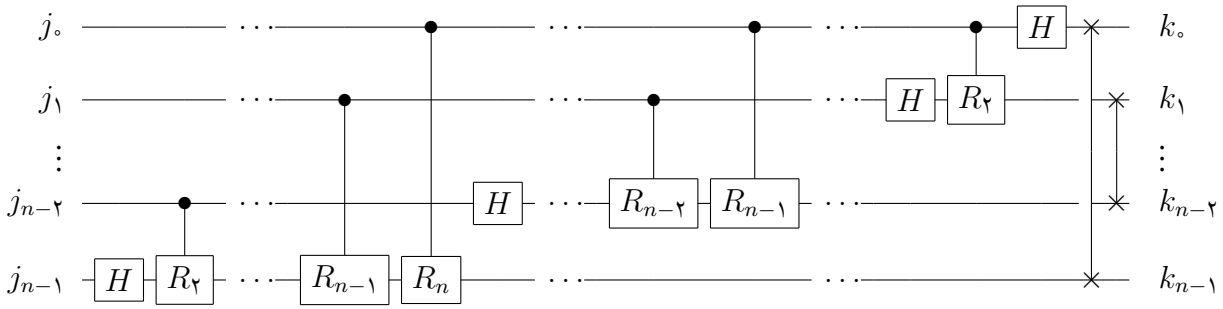
- **مقداردهی اولیه:** یک Register کوانتومی در حالت پایه $|j\rangle$ را مقداردهی اولیه می‌کنیم.
- **برهم‌نهی:** یک گیت Hadamard را به کیوبیت با بیش‌ترین ارزش (منظور سمت چپ‌ترین کیوبیت در نوشتار دودویی است) اعمال می‌کنیم تا یک برهم‌نهی از حالات ایجاد شود.
- **چرخش فاز کنترل‌شده:** یک‌رشته گیت‌های چرخش فاز کنترل‌شده R_k را از اولین کیوبیت به هر کیوبیت بعدی اعمال می‌کنیم، جایی که R_k یک چرخش با زاویه‌ای وابسته به موقعیت کیوبیت در Register است. این وابستگی را می‌توان با ورودی دیگر کنترلی گیت‌ها، اعمال کرد.
- **معکوس کردن:** برای تصحیح ترتیب کیوبیت‌ها پس از اعمال QFT، موقعیت کیوبیت‌ها را معکوس می‌کنیم.

مدار کوانتومی

شکل ۴.۳ نمودار مدار کوانتومی برای ورودی با n کیوبیت را نشان می‌دهد.

چالش‌ها

همان چالش مشابه با دیگر الگوریتم‌ها یعنی پیاده‌سازی یکسان در شبیه‌سازها مختلف است. تفاوت دیگر این الگوریتم، نبود مرحلهٔ اندازه‌گیری است که از ماهیت آن نشئت می‌گیرد در نتیجه چالش عمده، بررسی درستی عملکرد این الگوریتم است.



شکل ۴.۳: نمودار مدار کوانتومی الگوریتم QFT

Grover

الگوریتم Grover یک الگوریتم جستجوی کوانتومی است که کاهش پیچیدگی درجهٔ دوم نسبت به الگوریتم‌های جستجوی کلاسیک را فراهم می‌کند. این الگوریتم برای جستجو در یک پایگاه‌داده نامرتب یا حل مسئلهٔ یافتن یک ورودی برای تابعی که خروجی خاصی تولید می‌کند، استفاده می‌شود. درحالی‌که الگوریتم‌های کلاسیک به $O(N)$ بار ورودی‌دادن برای یافتن عنصر موردنظر در یک فهرست از N شئ نیاز دارند، الگوریتم Grover می‌تواند این کار را با $O(\sqrt{N})$ انجام دهد که نشان‌دهندهٔ توانایی محاسبات کوانتومی برای حل برخی مسائل به طور کارآمدتر نسبت به محاسبات کلاسیک است.

پیاده‌سازی

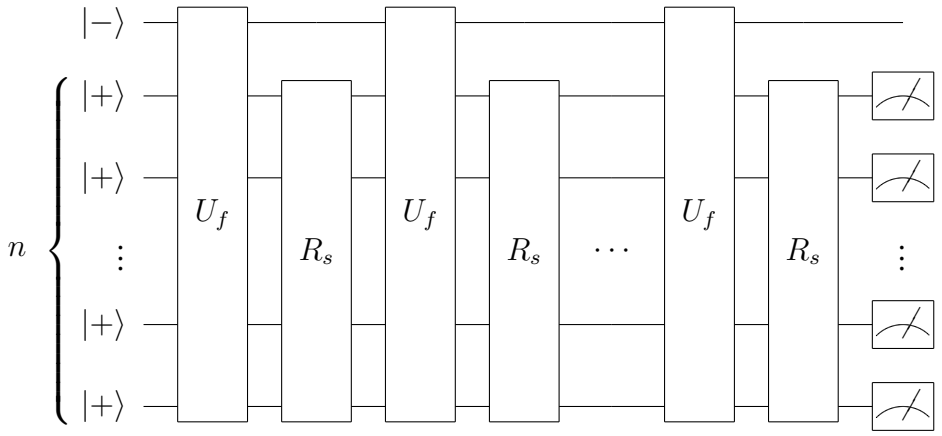
الگوریتم Grover شامل مراحل کلیدی زیر است:

- **مقداردهی اولیه:** الگوریتم را با مقداردهی اولیه n کیوبیت در حالت $|0\rangle$ شروع می‌کنیم.
- **برهم‌نهی:** سپس گیت‌های Hadamard را اعمال می‌کنیم تا همهٔ کیوبیت‌ها در یک برهم‌نهی از تمام حالات ممکن قرار بگیرند.
- **پرسش از Oracle:** Oracle کوانتومی را اعمال می‌کنیم تا راه‌حل صحیح را با معرفی یک تغییر فاز π در دامنه حالت هدف علامت‌گذاری کند.

- **تقویت دامنه:** الگوریتم با اعمال یک‌رشته بازتاب[1]ها، احتمال دامنهٔ حالت صحیح را تقویت می‌کند. بازتاب اول روی میانگین اندازهٔ تمام حالات است و سپس بازتاب دیگر روی حالت اولیه اعمال می‌شود.
- **تکرار:** مراحل ۲ و ۳ به تعداد $O(\sqrt{N})$ بار تکرار می‌شوند تا احتمال اندازه‌گیری حالت صحیح به حداکثر برسد.
- **اندازه‌گیری:** در نهایت، کیوبیت‌ها اندازه‌گیری می‌شوند و نتیجه با احتمال بالاتر پاسخ ما خواهد بود.

مدار کوانتومی

نمودار مدار کوانتومی الگوریتم Grover در شکل ۴.۳ رسم شده است.



شکل ۵.۳: نمودار مدار کوانتومی الگوریتم Grover

[1] Reflection

چالش‌ها

علاوه بر چالش‌های مطرح شده مثل دیگر الگوریتم‌ها، پیاده‌سازی این الگوریتم مانند الگوریتم ،Simon نیاز به ذخیرهٔ خروجی هر بار اجرای الگوریتم را دارد که با بعضی شبیه‌سازهایی که توسعهٔ آن‌ها تکمیل نشده، به چالش برخورد کردیم.

Shor

الگوریتم Shor یک الگوریتم کوانتومی است که برای تجزیهٔ اعداد صحیح، به‌ویژه تجزیهٔ یک عدد صحیح بزرگ مانند N به عوامل اول آن، طراحی شده است. این یک عملیاتی است که برای رایانه‌های کلاسیک محاسباتی دشوار است، به‌ویژه با بزرگ‌شدن N که آن را به پایه‌ای برای سامانه‌های رمزنگاری رایج مانند RSA تبدیل کرده است. الگوریتم Shor که توسط Peter Shor در سال ۱۹۹۴ معرفی شد، می‌تواند N را به طور کارآمد با استفاده از محاسبات کوانتومی تجزیه کند، رمزنگاری RSA را شکسته و توانایی رایانه‌های کوانتومی برای حل این دسته از مسائل را به طور نمایی سریع‌تر از رایانه‌های کلاسیک نشان دهد.

پیاده‌سازی

الگوریتم Shor را می‌توان به مراحل زیر تقسیم کرد:

- **انتخاب یک عدد تصادفی:** یک عدد صحیح تصادفی a را انتخاب می‌کنیم به‌طوری‌که نامعادلهٔ $1 < a < N$ برقرار باشد. بزرگ‌ترین مقسوم‌علیه مشترک (ب.م.م) a و N را با استفاده از الگوریتم اقلیدسی محاسبه می‌کنیم. اگر ب.م.م بیشتر از ۱ باشد، پس این یک عامل غیربدیهی N است و الگوریتم متوقف می‌شود.
- **یافتن دوره تناوب:** اگر بزرگ‌ترین مقسوم‌علیه مشترک برابر با ۱ باشد، مرحلهٔ بعدی یافتن دورهٔ تناوب r تابع $f(x) = a^x \mod N$ است. دورهٔ r کوچک‌ترین عدد صحیح است که معادلهٔ

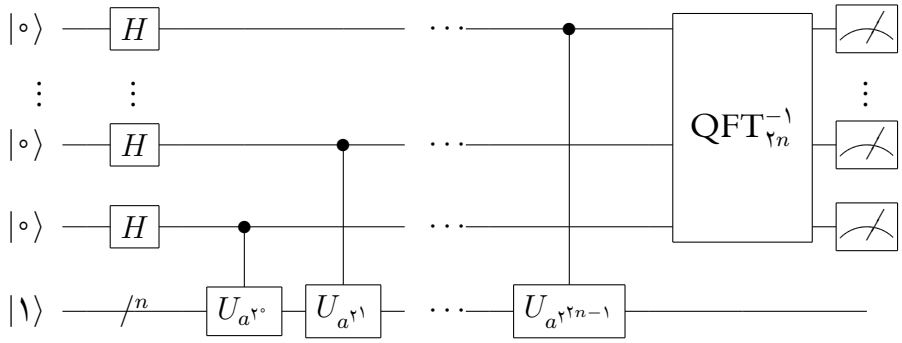
$a^r \equiv ۱ \mod N$ در رابطه با آن برقرار است. یافتن این دوره تناوب برای رایانه‌های کلاسیک پیچیده است، اما می‌توان آن را به طور کارآمد با استفاده از یک رایانهٔ کوانتومی و تبدیل فوریه کوانتومی انجام داد.

- **تعیین عوامل:** هنگامی که دورهٔ r پیدا شد، اگر r زوج بود، مقدار $a^{r/۲} \mod N$ را محاسبه می‌کنیم. اگر $a^{r/۲} \not\equiv -۱ \mod N$ بود، عوامل N توسط بزرگ‌ترین مقسوم‌علیه‌های مشترک $\gcd(a^{r/۲}-۱, N)$ و $\gcd(a^{r/۲}+۱, N)$ مشخص می‌شوند. اگر این مقادیر، عوامل غیربدیهی تولید نکنند، این فرایند با یک a دیگر تکرار می‌شود.

قسمت کوانتومی الگوریتم Shor شامل یافتن دوره r با استفاده از هم‌زمانی کوانتومی و تبدیل فوریه کوانتومی است که اجازه می‌دهد دوره r به طور کارآمد تعیین شود.

مدار کوانتومی

مدار کوانتومی الگوریتم Shor در شکل ۶.۳ رسم شده است.



شکل ۶.۳: نمودار مدار کوانتومی الگوریتم Shor

چالش‌ها

علاوه بر چالش‌های ذکر شده، پیاده‌سازی این الگوریتم، بسیار زمان‌بر و مشکل بود. جدا از فهم خود الگوریتم، منطبق کردن خواسته‌های آن و پیاده‌سازی آن در شبیه‌سازهای مختلف، زمان زیادی از این پژوهش را به خود اختصاص داد.

نرم‌افزار محک پیاده‌سازی‌شده

در روند محک، عملیات‌هایی وجود دارند که برای همهٔ شبیه‌سازها تکراری هستند. برای مثال، رسم نمودار، تغییرات جزئی در نحوهٔ ورودی‌های اصلی، نحوه‌ی اندازه‌گیری معیارهای محک و غیره. به همین منظور در اواسط طول دورهٔ پژوهش، تصمیم بر آن شد که یک ابزار نرم‌افزاری برای تسهیل در این روند توسعه داده شود که به روند پژوهش کمک کند. این ابزار در مخزن GitHub[1] به‌صورت عمومی قابل‌دسترس است. با این که این ابزار تمامی پیاده‌سازی‌ها و روند پژوهش را شامل می‌شود؛ اما تلاش بر آن است که در آینده به یک محصول نرم‌افزاری پیشرفته برای محک شبیه‌سازهای کوانتومی تبدیل شود به‌گونه‌ای که استفاده از آن مستقل از نرم‌افزار شبیه‌ساز و سخت‌افزار مربوط به آن باشد.

سخت‌افزار محک

همان‌طور که در قسمت پژوهش‌های پیشین هم گفته شد، یکی از وجه تمایزهای پژوهش ما با مشابه‌ترین پژوهش مربوطه، این است که فرض بر آن است، کاربر شبیه‌سازها از سخت‌افزار بسیار پیچیده و بزرگ استفاده نمی‌کند، بلکه از رایانهٔ شخصی معمول بهره می‌برد. به همین منظور رایانهٔ استفاده شده در پژوهش ما برای انجام آزمایش‌ها به شرح زیر است:

- **OS:** Linux, Ubuntu 22.04 LTS, 5.15.0-119-generic Kernel Version

[1] https://github.com/ahmadrv/QSB

- **CPU:** Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz with 8 Cores
- **RAM:** DDR3 with 16 Gigabyte Storage

فصل چهارم

نتیجه‌گیری

نتایج آزمایش‌های انجام‌شده با استفاده از ابزار توسعه داده شده در نمودارها و جدول‌های متفاوت گردآوری شده‌اند. دسته‌بندی این نتایج در حالت‌های متفاوت و با دیدگاه‌های مختلف در بخش‌های این فصل تدوین‌شده تا خواننده بتواند بر اساس آن‌ها قضاوت منصفانه‌تری را داشته باشد.

روند آزمایش‌ها

همان‌طور که در قبل هم اشاره شد، انجام آزمایش‌ها روندی الگوپذیر بوده است. برای بهبود و تسریع در این روند، از ابزار اشاره شده در طول آزمایش‌ها بهره‌برداری شده است.

آزمایش‌ها به این صورت بوده که برای هر پلتفرم، پیاده‌سازی الگوریتم‌های محک، به صورتی که این پیاده‌سازی‌ها از یک ساختار الگوریتمی و نرم‌افزاری یکسان پیروی کنند، انجام شده است. این که ساختار در پیاده‌سازی‌های مختلف حفظ شود به این معنا است که مثلاً اگر در پیاده‌سازی یک الگوریتم در یک پلتفرم نیاز است از حلقه استفاده شود، در دیگری هم به همین صورت باشد به‌طوری‌که در هر دو پلتفرم پیاده‌سازی‌ها از پیچیدگی محاسباتی یکسانی برخوردار باشند چرا که هدف محک منصفانهٔ آن‌ها

است. برای این امر، وقت زیادی از پژوهش صرف شد چرا که پیاده‌سازی‌های موجود در هر پلتفرم با دیگری متفاوت است و عملاً برای محکی که جانب انصاف را رعایت کند، نمی‌تواند مورداستفاده قرار بگیرد. در نتیجه در پیاده‌سازی‌های موجود باید تجدید نظر انجام می‌شد.

همان‌طور که اشاره شد، هدف پژوهش ما کاربرانی بوده‌اند که از رایانه‌های شخصی استفاده می‌کنند. به همین منظور، رایانهٔ استفاده شده در طول این آزمایش‌ها، رایانهٔ شخصی موجود در آزمایشگاه دانشکده با پردازندهٔ Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz با ۸ هستهٔ پردازشی، حافظهٔ ۱۶ گیگابایتی، بدون پردازندهٔ گرافیکی و سیستم‌عامل لینوکس، توزیع اوبونتو با نسخهٔ هستهٔ 5.15.0-119-generic بوده است.

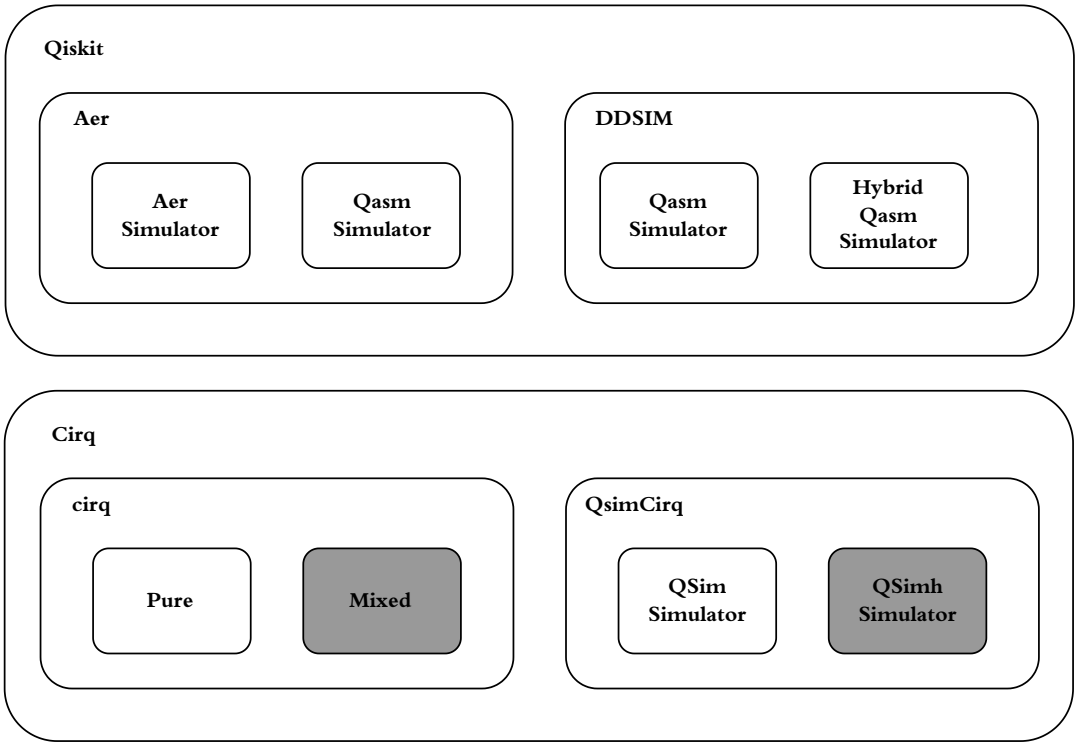
برای آزمایش هر شبیه‌ساز توسط ابزار توسعه داده شده، زمان اجرا و میزان حافظهٔ مصرفی با افزایش تعداد کیوبیت اندازه‌گیری و در پایگاه‌داده ذخیره شده است. از آن‌جایی که شبیه‌سازها در دو پلتفرم Qiskit و Cirq قرار دارند، در ادامه، آن‌ها را به تفکیک پلتفرم بررسی می‌کنیم. شکل ۱.۴ بیانگر نحوهٔ ارتباط پلتفرم‌ها، بک‌اند[1]ها و شبیه‌سازهاست.

Qiskit

این پلتفرم توسط گروه تحقیقاتی IBM [۱۹] معرفی شده است که شامل چندین شبیه‌ساز با رویکردهای متفاوت شامل آن‌چه که در بخش شبیه‌سازهای مدار کوانتومی گفته شده است. این پلتفرم علاوه بر شبیه‌ساز قابلیت‌هایی نظیر بهینه‌سازی و ترجمهٔ الگوریتم‌ها به زبان اسمبلی کوانتومی را نیز دارا است. علاوه‌برآن قابلیت اتصال به رایانه‌های واقعی کوانتومی ابری را نیز فراهم کرده است.

هر پلتفرم برای جداسازی شبیه‌سازها باتوجه‌به رویکرد آن‌ها از بک‌اندهای متفاوت که همان لایه‌ای نرم‌افزاری باهدف چارچوب‌بندی به هر رویکرد شبیه‌سازی هستند، تشکیل شده است. در ادامه، شبیه‌سازها را به تفکیک بک‌اند، بررسی خواهیم کرد.

[1] Back-End



شکل ۱.۴: نمودار روابط پلتفرم‌ها و اجزای آن‌ها (قسمت‌های خاکستری در پژوهش ما مورد بحث و بررسی نبوده‌اند.)

Aer

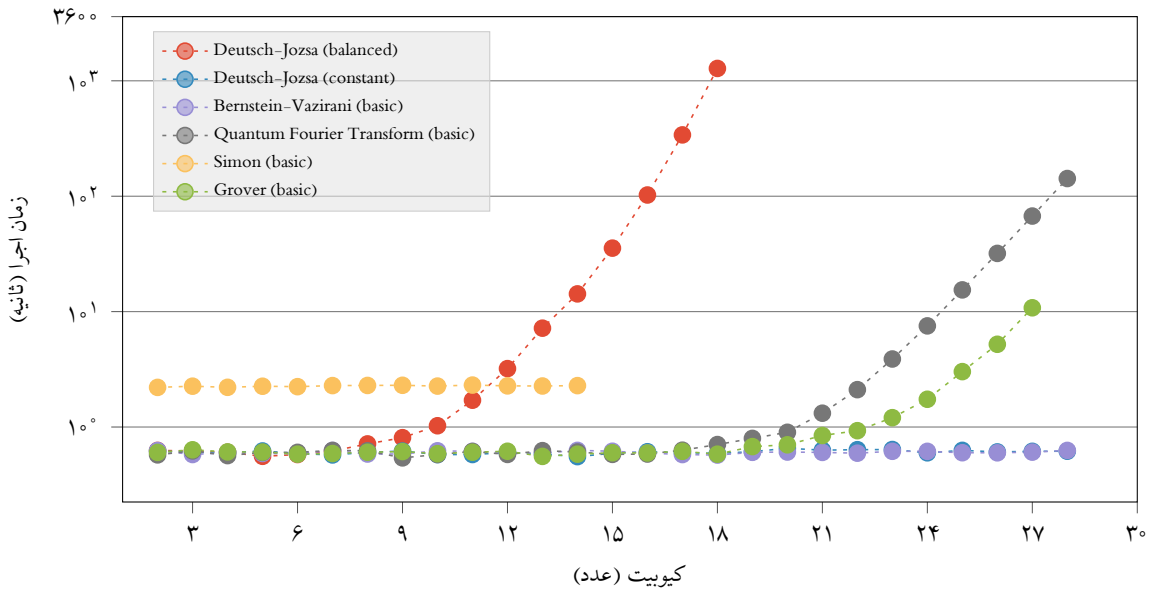
این بک‌اند شامل چهار شبیه‌ساز AerSimulator، QasmSimulator، StatevectorSimulator و UnitarySimulator است [۱۹]. در پژوهش ما فقط به بررسی دو مورد اول پرداخته شده است.

AerSimulator: این شبیه‌ساز از چندین روش شبیه‌سازی و گزینه‌های قابل‌تنظیم برای هر روش شبیه‌سازی پشتیبانی می‌کند [۲۰]. در محک این شبیه‌ساز، ابتدا الگوریتم‌های توضیح داده شده در بخش انتخاب الگوریتم‌های محک را اجرا و زمان اجرا و میزان حافظهٔ مصرفی آن را به تحریر نمودار درآورده‌ایم. نتیجهٔ حاصل شده در شکل‌های ۲.۴ و ۳.۴ نشانگر رفتار این شبیه‌ساز در اجرای الگوریتم‌های مختلف است.

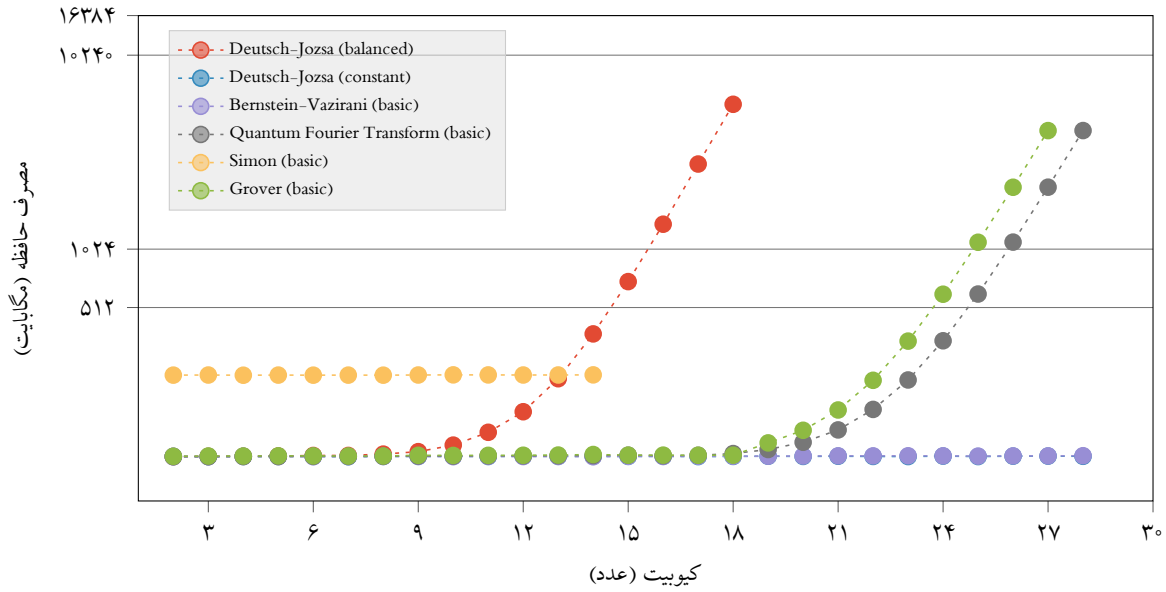
از آن‌جایی که نتایج مربوط به الگوریتم Shor رفتار متفاوتی نسبت به دیگر الگوریتم‌ها داشته است، خروجی‌های مربوط به این الگوریتم در قالب جدول گزارش خواهد شد. این تفاوت در آن‌جایی است که برخلاف دیگر الگوریتم‌ها، داده‌های ورودی این الگوریتم محدودتر هستند و افزایش تعداد کیوبیت‌ها با افزایش عدد مرکب از دو عدد اول صورت می‌پذیرد. برای مثال در جدول ۱.۴ برای تجزیه‌ی عدد مرکب ۷۷ که از حاصل ضرب اعداد اول ۷ و ۱۱ بدست می‌آید، الگوریتم Shor به ۲۴ کیوبیت احتیاج دارد تا بتواند این عدد را به عوامل اول آن تجزیه کند. به دلیل هم‌پوشانی که بین فاکتورهای اول اعداد مرکب وجود دارد، نمی‌توان تعداد کیوبیت‌ها را یکی‌یکی افزایش داد و به همین دلیل زمان اجرا و حافظه‌ی مصرفی در قالب جداول نظیر جدول ۱.۴ گزارش شده است.

عدد مرکب	تعداد کیوبیت	میانگین زمان اجرا (ثانیه)	میانگین حافظهٔ مصرفی (مگابایت)
۱۵	۱۵	۱.۶۳۳۵۳۶	۱۲۱.۷۲۳۵۶۸
۳۵	۲۱	۲.۰۳۷۱۸۸	۱۲۲.۵۲۹۹۹۰
۷۷	۲۴	۲.۰۵۶۴۹۳	۱۲۵.۲۰۷۳۴۴
۱۴۳	۲۷	۱۳.۹۸۲۸۷۳	۱۲۹.۹۹۰۸۸۵
۴۳۷	۳۰	N/A	N/A

جدول ۱.۴: جدول اطلاعات مربوط به اجرای الگوریتم Shor با Aer Simulator



شکل ۲.۴: نمودار زمان اجرا بر حسب تعداد کیوبیت الگوریتم‌های مختلف در Aer Simulator



شکل ۳.۴: نمودار مصرف حافظه بر حسب تعداد کیوبیت الگوریتم‌های مختلف Aer Simulator

از آن‌جایی که در محدودیت‌های این شبیه‌ساز تعداد ۳۲ کیوبیت ذکر شده همان‌طور که مشاهده می‌شود این شبیه‌ساز، در این محدوده، در اجرای همهٔ الگوریتم‌ها همان‌طور که انتظار می‌رود رفتار کرده است. دلیل این که محک با الگوریتمی نظیر Simon فقط تا ۱۶ کیوبیت ادامه پیدا کرده است این است که این الگوریتم برای حل مسئله به دوبرابر تعداد کیوبیت ورودی نیاز دارد. دیگر الگوریتم‌ها نظیر Grover در تعداد کیوبیت کمتر رفتاری خطی و با افزایش تعداد کیوبیت رفتاری نمایی به خود می‌گیرند، طبق آن‌چه که در [۱] هم حدس زده شده است، امکان بهینه‌سازی غیر جامع این شبیه‌سازها به‌طوری‌که برای تعداد کیوبیت کمتر، بهینه شده باشند، وجود دارد.

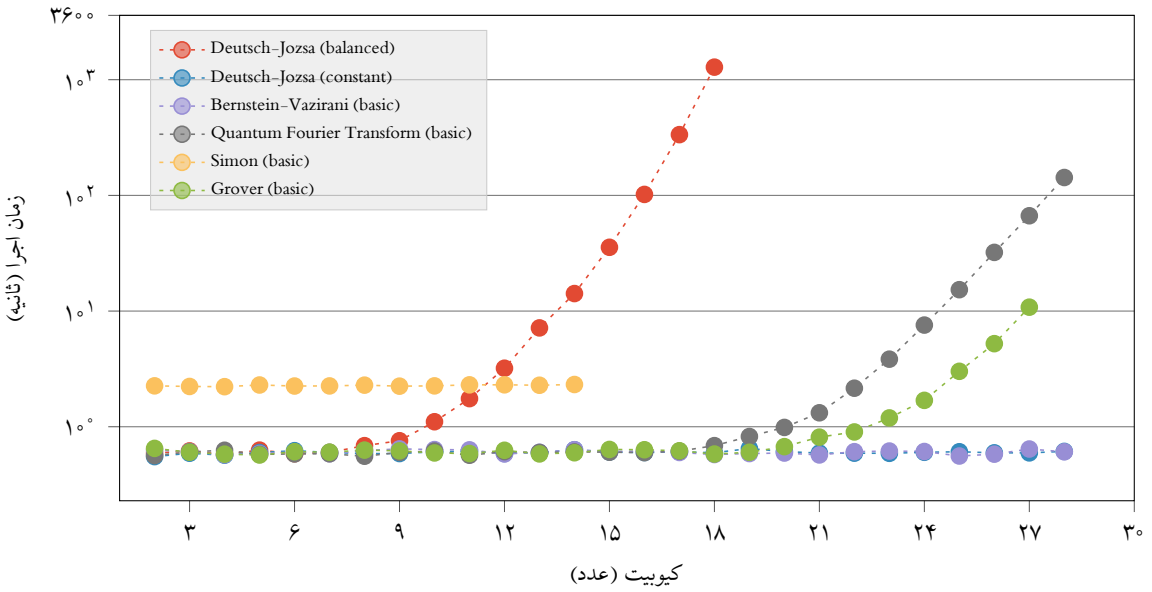
QasmSimulator: این شبیه‌ساز هم از چندین روش شبیه‌سازی و اضافه‌کردن نویز به محاسبات پشتیبانی می‌کند. مانند قبل، اجرای الگوریتم‌ها با این شبیه‌ساز، در شکل ۴.۴ برای زمان اجرا و در شکل ۵.۴ برای مصرف حافظه، آورده شده است.

از آن‌جایی که این دو شبیه‌ساز در یک بک‌اند توسعه داده شده‌اند، تفاوت چندانی در عملکرد آن‌ها مشاهده نمی‌شود و رفتار مشابهی را در مواجهه با الگوریتم‌های مختلف دارند که این رفتار موردانتظار از شبیه‌سازهای مدارهای کوانتومی است.

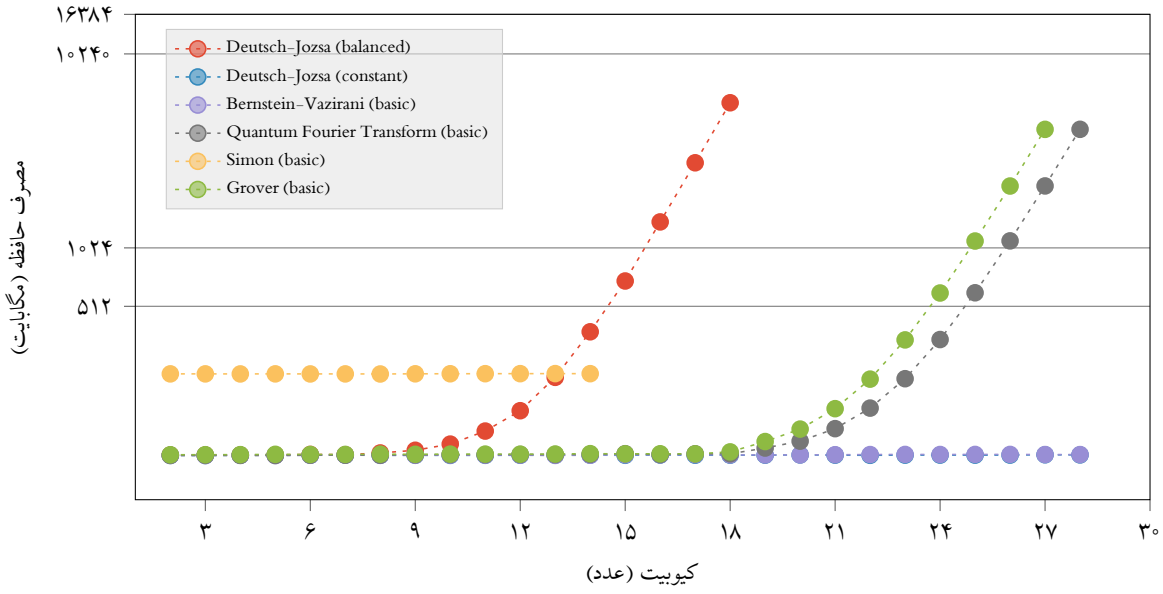
به صورت مجزا، عملکرد این شبیه‌ساز در اجرای الگوریتم Shor در جدول ۲.۴ گزارش شده است.

عدد مرکب	تعداد کیوبیت	میانگین زمان اجرا (ثانیه)	میانگین حافظهٔ مصرفی (مگابایت)
۱۵	۱۵	۱.۷۷۰۸۵۲	۱۲۱.۷۳۲۶۸۲
۳۵	۲۱	۳.۷۱۴۲۶۵	۱۲۳.۲۲۶۵۶۲
۷۷	۲۴	۲.۵۴۲۱۶۶	۱۲۵.۴۱۱۸۷۵
۱۴۳	۲۷	۱۴.۰۵۱۵۲۹	۱۳۰.۳۹۳۲۲۹
۴۳۷	۳۰	N/A	N/A

جدول ۲.۴: جدول اطلاعات مربوط به اجرای الگوریتم Shor با Qasm Simulator



شکل ۴.۴: نمودار زمان اجرا بر حسب تعداد کیوبیت الگوریتم‌های مختلف در Qasm Simulator



شکل ۵.۴: نمودار مصرف حافظه بر حسب تعداد کیوبیت الگوریتم‌های مختلف در Qasm Simulator

DDSIM

این بک‌اند چهارچوبی برای شبیه‌سازها تعیین کرده است که از یک روش مبتنی بر گراف، به‌ویژه نمودارهای تصمیم‌گیری، برای شبیه‌سازی کارآمد مدارهای کوانتومی استفاده می‌کند و به‌جای ذخیره بردارها و ماتریس‌های بزرگی که به‌صورت نمایی رشد می‌کنند، از این نمودارها برای ثبت تکرارهای موجود در محاسبات بهره می‌برد. در واقع به نحوی از برنامه‌نویسی پویا برای بهبود کارایی شبیه‌سازی استفاده می‌کند و به‌این‌ترتیب استفاده از حافظه را به میزان قابل‌توجهی کاهش می‌دهد. عملیات کوانتومی از طریق ضرب ماتریس‌ها و بردارها در این قالب به‌صورت بازگشتی انجام می‌شوند بدون این که نیاز باشد کل آن‌ها پردازش شوند. اندازه‌گیری نیز به همین شکل انجام می‌شود و تغییرات حالت به طور کارآمد در نمودارها نمایش داده می‌شود. این روش باعث می‌شود که این نوع از شبیه‌سازها در مقایسه با شبیه‌سازهای دیگر، عملکرد بهتری داشته باشند و بتوانند تعداد بیش‌تری کیوبیت را مدیریت کنند و زمان و حافظهٔ محاسباتی کمتری مصرف کنند [۲۱].

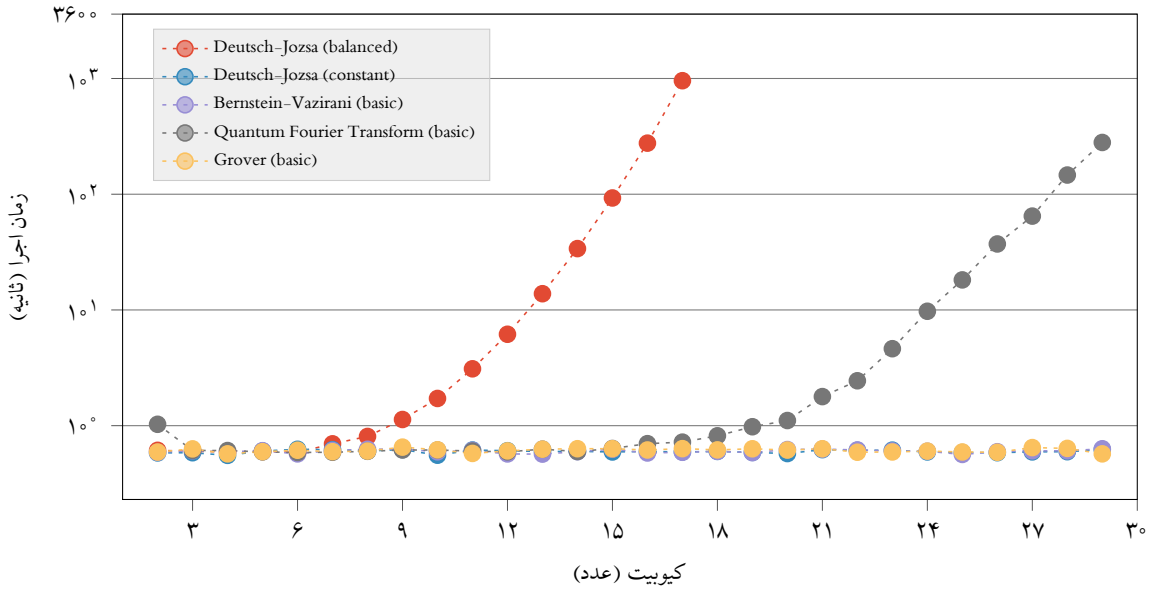
QasmSimulator: این شبیه‌ساز از روش Schrödinger استفاده می‌کند که بر اساس نمودارهای تصمیم‌گیری به‌عنوان یک ساختمان داده، طراحی شده و می‌توان از آن برای به‌دست‌آوردن بردار حالت کامل مدار کوانتومی «شبیه‌سازی قوی» [۲۱] و یا نمونه‌گیری از توزیع خروجی یک مدار کوانتومی «شبیه‌سازی ضعیف» [۲۲] بهره برد. برای این منظور، شبیه‌ساز با نمایش نمودار تصمیم‌گیری از حالت اولیه آغاز می‌شود و سپس در هر مرحله، گیت‌های مدار را یکی‌یکی اعمال می‌کند. نمایش نمودار تصمیم‌گیری از بردار حالت در هر مرحله به‌روزرسانی می‌شود. این شبیه‌ساز قادر است مدارهای کوانتومی تقریباً دلخواه، از جمله مدارهایی با اندازه‌گیری‌ها و بازتنظیم‌های میانی را مدیریت کند. برای مدارهایی که شامل عملیات غیر واحدی (به‌جز اندازه‌گیری‌ها در انتهای مدار) نیستند، شبیه‌سازی فقط یک‌بار انجام می‌شود. در این حالت، تعداد نمونه‌های درخواستی به طور متعاقب از نمودار تصمیم‌گیری نهایی گرفته می‌شود که منجر به زمان اجرای سریع شبیه‌سازی می‌گردد.

شایان‌ذکر است که این شبیه‌ساز با شبیه‌سازی که در قسمت قبل توضیح داده شده است، متفاوت است. با این که هر دو از روش Schrödinger برای شبیه‌سازی بهره گرفته‌اند؛ اما ساختمان داده‌ای که در شبیه‌ساز دوم با بک‌اند DDSIM توسعه‌یافته به طور کامل با شبیه‌ساز با بک‌اند Aer متفاوت است.

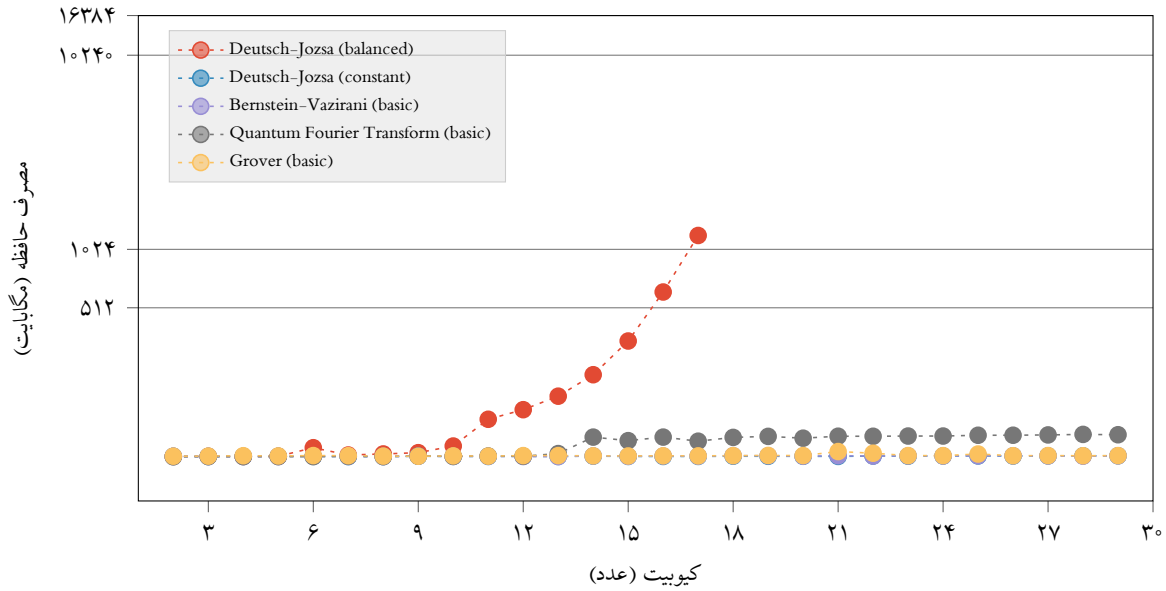
شکل‌های ۶.۴ و ۷.۴ رفتار این شبیه‌ساز را نسبت به الگوریتم‌های مختلف نشان می‌دهد که به‌طورکلی در مقایسه با شبیه‌ساز Aer در زمان اجرا و حافظهٔ کمتری شبیه‌سازی‌ها را انجام داده است که این خود نقطه‌ی قوت این شبیه‌ساز را نشان می‌دهد و این بهبود برای محققانی که قصد اجرای شبیه‌سازها بر روی رایانه‌های شخصی را دارند بسیار مؤثر خواهد بود. همان‌طور که در بخش‌های قبل گفته شد، پیاده‌سازی الگوریتم Simon، به دلیل عدم وجود ویژگی ذخیرهٔ نتایج اجرای الگوریتم‌ها، در این بک‌اند انجام نشده است و به همین دلیل در نمودارهای ذکر شده، اطلاعاتی مربوط به الگوریتم Simon وجود ندارد.

با تفاوت‌های ذکرشده در ساختار الگوریتم Shor و تفاوت‌های آن با دیگر محک‌ها، عملکرد این شبیه‌ساز در اجرای این الگوریتم به صورت مجزا در جدول ۳.۴ گزارش شده است. رفتار این شبیه‌ساز در اجرای این الگوریتم با افزایش از ۲۱ کیوبیت به ۲۴ کیوبیت، غیرقابل انتظار بوده است چرا که انتظار بر این است با افزایش تعداد کیوبیت زمان اجرا و میزان مصرف حافظه هم افزایش یابد. در توجیه این رفتار، از آن‌جایی که الگوریتم Shor در مراحل خود به صورت احتمالی به پاسخ می‌رسد، رسیدن به پاسخ در زمان محدودشده در آزمایش‌ها (۳۶۰۰ ثانیه برای هر اجرا) امکان‌پذیر نبوده است. این در حالی است که هر اجرا به بیش از ۵ بار تکرار انجام شده است. در نتیجه این کاهش زمان در قبال نرسیدن به پاسخ نهایی بوده است.

Hybrid QasmSimulator: این مدل شبیه‌ساز از رویکرد ترکیبی Schrödinger-Feynman استفاده می‌کند به‌طوری‌که از تمام حافظه و واحدهای پردازشی موجود برای شبیه‌سازی کارآمد مدارهای کوانتومی بهره ببرد. در شبیه‌سازی‌هایی که به سبک Schrödinger هستند با محدودیت‌های حافظه



شکل ۶.۴: نمودار زمان اجرا بر حسب تعداد کیوبیت الگوریتم‌های مختلف در DDSIM Qasm Simulator



شکل ۷.۴: نمودار مصرف حافظه بر حسب تعداد کیوبیت الگوریتم‌های مختلف در DDSIM Qasm Simulator

میانگین حافظهٔ مصرفی (مگابایت)	میانگین زمان اجرا (ثانیه)	تعداد کیوبیت	عدد مرکب
۱۲۴.۵۹۰۸۸۵	۱.۶۴۳۸۳۳	۱۵	۱۵
۱۳۱.۸۲۳۴۹۵	۳.۰۰۸۱۴۰	۲۱	۳۵
۱۲۸.۱۴۹۰۶۳	۱.۸۸۳۹۸۹	۲۴	۷۷
۱۴۷.۱۶۷۱۸۸	۶.۷۶۵۳۰۸	۲۷	۱۴۳
N/A	N/A	۳۰	۴۳۷

جدول ۳.۴: جدول اطلاعات مربوط به اجرای الگوریتم Shor با DDSIM Qasm Simulator

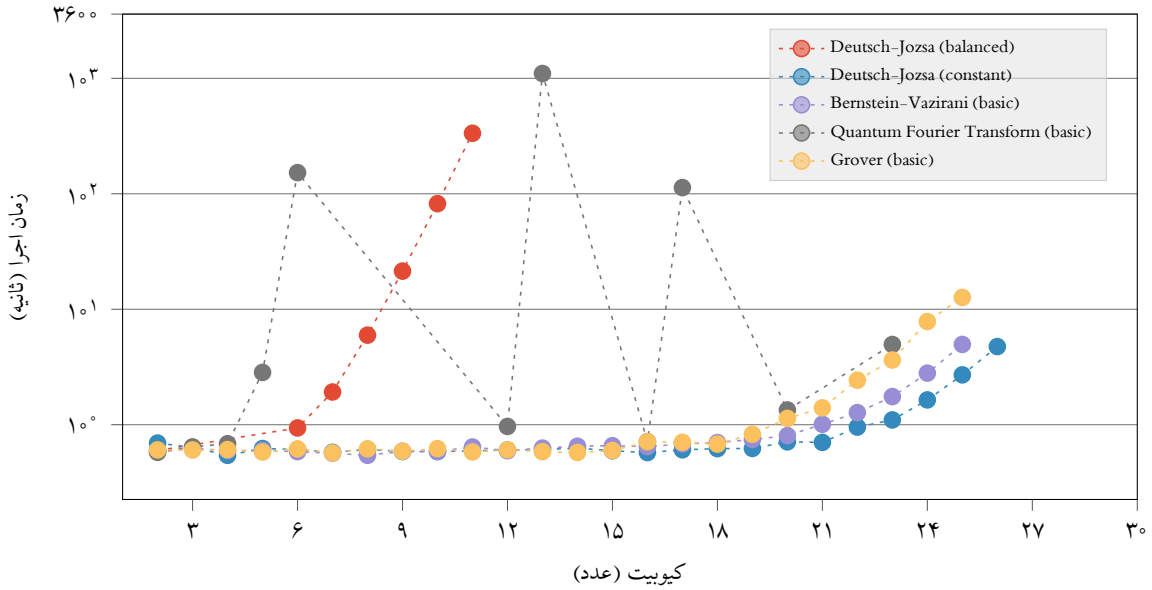
مواجه می‌شوند و یا در شبیه‌سازی‌های به سبک Feynman مدت‌زمان بسیار زیادی نیاز دارند، این شبیه‌ساز سعی می‌کند با ایجاد یک مصالحه بین الزامات حافظه و زمان اجرا بهترین شبیه‌سازی ممکن، وابسته به مدار کوانتومی ورودی را ارائه دهد [۲۳].

شکل‌های ۸.۴ و ۹.۴ رفتار این شبیه‌ساز را نسبت به الگوریتم‌های مختلف نشان می‌دهد. از آن‌جایی که این مدل شبیه‌ساز در مراحل اولیهٔ توسعهٔ خود قرار دارد، این احتمال وجود دارد که انتخاب بین استفاده از رویکردهای یاد شده، باعث چنین ناهمواری‌هایی در نمودار زمان اجرا و حافظه شود. در کل، بدون در نظر گرفتن وقت صرف شده برای توسعهٔ شبیه‌سازهای مختلف، این شبیه‌ساز عملکرد خوبی نسبت به شبیه‌سازهایی که تاکنون شرح داده شده‌اند، نداشته است.

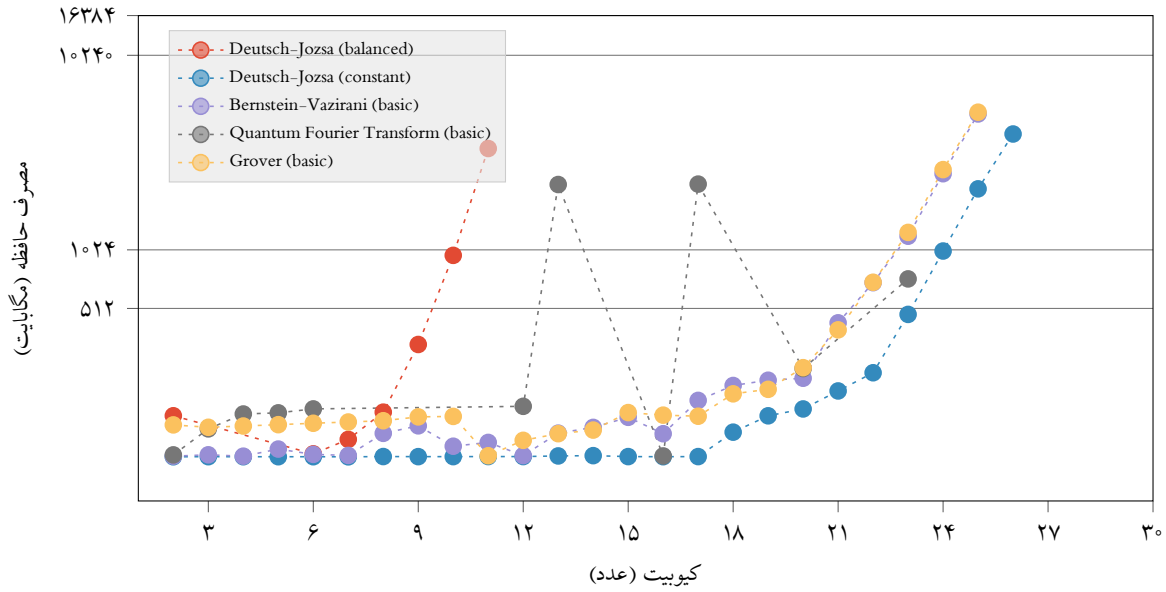
همان مشکلی که برای پیاده‌سازی الگوریتم Simon بیان شد، باعث عدم پیاده‌سازی الگوریتم Shor در این شبیه‌ساز گردیده است.

Cirq

Cirq یک پلتفرم محاسبات کوانتومی است که برای طراحی، بهینه‌سازی و اجرای مدارهای کوانتومی بر روی رایانه‌های کوانتومی و شبیه‌سازهای کوانتومی استفاده می‌شود. Cirq قابلیت اجرا بر روی پردازنده‌های کوانتومی مانند Alpine، Pasqal، Rigetti و IonQ را نیز داراست. این پلتفرم دارای شبیه‌سازهای داخلی برای آزمایش مدارهای کوچک است و از شبیه‌سازهایی با عملکرد بالا مانند Qulacs و quimb نیز پشتیبانی می‌کند. Cirq همچنین با پلتفرم‌های نرم‌افزاری دیگر مانند QC



شکل ۸.۴: نمودار زمان اجرا بر حسب تعداد کیوبیت الگوریتم‌های مختلف در DDSIM Hybrid Qasm Simulator



شکل ۹.۴: نمودار مصرف حافظه بر حسب تعداد کیوبیت الگوریتم‌های مختلف در DDSIM Hybrid Qasm Simulator

Ware Forge، Xanadu Pennylane و Zapata Orquestra می‌تواند ارتباط برقرار کند و راه‌حل‌های پیچیده‌تری را پوشش دهد. Cirq بخشی از اکوسیستم متن‌باز Google Quantum AI است که شامل ReCirq، OpenFermion و TensorFlow Quantum می‌شود.

Cirq

همان‌طور که اشاره شد پلتفرم Cirq از شبیه‌سازهای داخلی برای مدارهای کوچک‌تر را داراست. دو نوع اصلی شبیه‌سازی که Cirq پشتیبانی می‌کند، شبیه‌سازی حالت خالص و شبیه‌سازی حالت مخلوط هستند. شبیه‌سازهای حالت خالص توسط cirq.Simulator و شبیه‌سازهای حالت مخلوط توسط cirq.DensityMatrixSimulator پشتیبانی می‌شوند که در پژوهش ما فقط به شبیه‌سازی‌های حالت خالص پرداخته شده است چرا که شبیه‌سازی‌های حالت مخلوط به‌طورکلی از ماهیت متفاوتی برخوردارند. در واقع شبیه‌ساز حالت خالص و شبیه‌ساز حالت مخلوط به این واقعیت اشاره دارند که این شبیه‌سازی‌ها برای مدارهای کوانتومی هستند که در طول شبیه‌سازی اعم از اعمال انواع گیت‌ها، اندازه‌گیری‌ها و نویزها که حاصل را در حالت خالص (یک حالت کوانتومی منفرد) یا حالت مخلوط (ترکیبی از حالت‌های کوانتومی مختلف) حفظ می‌کنند. شبیه‌ساز حالت خالص از تکامل‌های نویزی پشتیبانی می‌کند، به شرطی که خلوص حالت را حفظ کنند.

به‌طورکلی، شبیه‌سازی حالت خالص، روند شبیه‌سازی و نتیجهٔ آن را در یک حالت کوانتومی حفظ می‌کند، چه باوجود نویز و چه بدون آن، درصورتی‌که شبیه‌سازی حالت مخلوط، روند شبیه‌سازی را در ترکیبی از حالت‌های کوانتومی ارائه می‌دهد.

برخی از شبیه‌سازهای دیگر با عملکرد بهتر نسبت به شبیه‌ساز قیدشده نیز یک رابط به Cirq ارائه می‌دهند. این شبیه‌سازها، به‌ویژه هنگام کار با مدارهای بزرگ‌تر، اغلب می‌توانند نتایج را سریع‌تر از شبیه‌سازهای داخلی Cirq ارائه دهند. Qsim نمونه‌ای از آن‌ها است [۲۴].

Pure: همان‌طور که گفته شد، این شبیه‌ساز داخلی پلتفرم Cirq برای مدارهای کوچک‌تر است که یک شبیه‌ساز بردار حالت با روش نمایش ماتریس پراکنده است و از کتابخانهٔ NumPy برای انجام محاسبات استفاده می‌کند.

شکل‌های ۱۰.۴ و ۱۱.۴ رفتار این شبیه‌ساز را نسبت به الگوریتم‌های مختلف نشان می‌دهد. نسبت به شبیه‌سازهایی که تاکنون بررسی شده است، عملکرد بهتری را چه در زمان اجرا و چه در مصرف حافظه نشان می‌دهد.

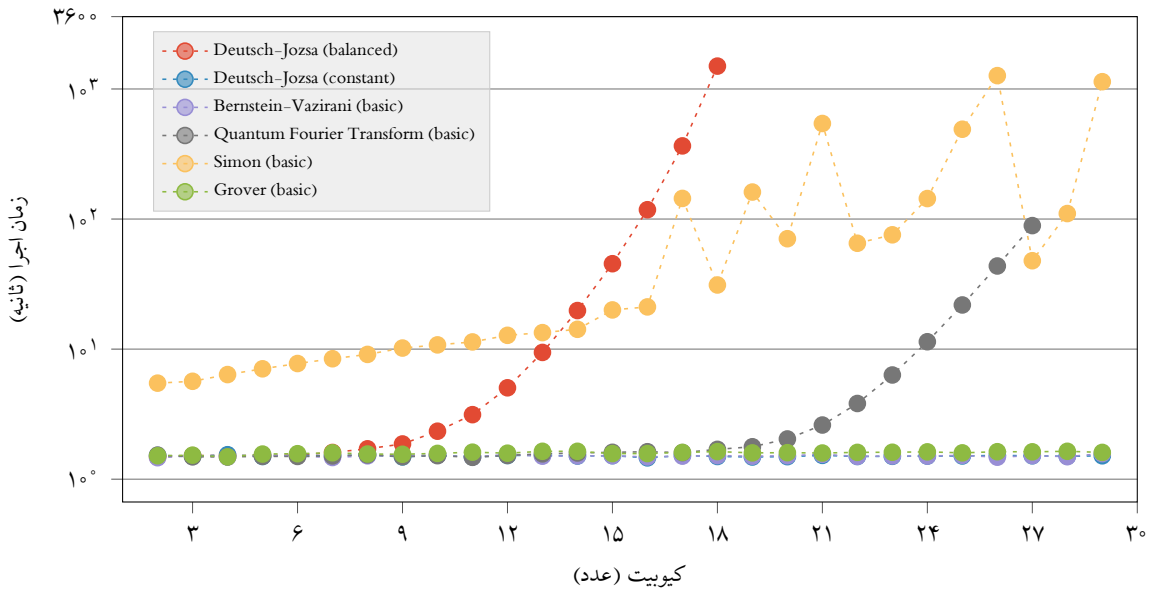
عملکرد این شبیه‌ساز در اجرای الگوریتم Shor در جدول ۴.۴ گزارش شده است.

عدد مرکب	تعداد کیوبیت	میانگین زمان اجرا (ثانیه)	میانگین حافظهٔ مصرفی (مگابایت)
۱۵	۱۵	۳.۸۹۳۹۸۵	۱۵۶.۱۸۳۰۷۳
۳۵	۲۱	۵.۲۱۲۱۲۹	۱۵۶.۶۸۴۴۶۲
۷۷	۲۴	۵.۴۴۷۷۲۹	۱۵۷.۴۳۱۲۳۴
۱۴۳	۲۷	۸.۱۸۹۲۹۱	۱۵۸.۸۸۵۷۸۵
۴۳۷	۳۰	N/A	N/A

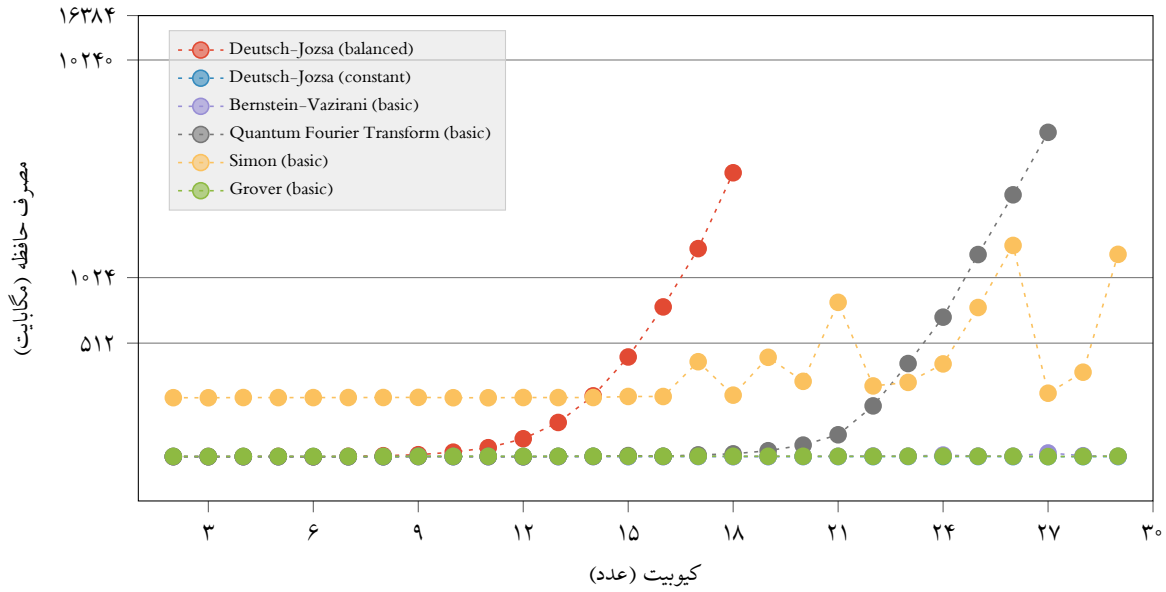
جدول ۴.۴: جدول اطلاعات مربوط به اجرای الگوریتم Shor با Cirq Simulator

Mixed: در تعریف مقایسه‌ای حالت‌های کوانتومی، یک حالت خالص کوانتومی حالتی است که می‌توان آن را با یک بردار حالت منفرد یا به‌صورت مجموعی از حالت‌های پایه توصیف کرد. یک حالت مخلوط کوانتومی، توزیع آماری‌ای از حالت‌های خالص است. در اینجا مهم است که به دو نوع میانگین‌گیری توجه کنیم: یکی میانگین‌گیری کوانتومی روی بردارهای پایهٔ حالت‌های خالص و دیگری میانگین‌گیری آماری روی مجموعه‌ای از حالت‌های خالص که مربوط حالت مخلوط کوانتومی است.

همان‌طور که ذکر شد، از آن‌جایی که شبیه‌سازی حالت‌های مخلوط کوانتومی دارای پیچیدگی بیش‌تری است و بررسی آن‌های نیازمند زمان دوچندان بوده است، از مطالعهٔ آن‌ها در پژوهش ما صرف‌نظر شده است.



شکل ۱۰.۴: نمودار زمان اجرا بر حسب تعداد کیوبیت الگوریتم‌های مختلف در Cirq Pure



شکل ۱۱.۴: نمودار مصرف حافظه بر حسب تعداد کیوبیت الگوریتم‌های مختلف در Cirq Pure

QSim

یک شبیه‌ساز کامل بردار حالت Schrödinger است. این شبیه‌ساز تمام ۲^n دامنه‌های بردار حالت را محاسبه می‌کند که n تعداد کیوبیت‌ها است. در واقع، شبیه‌ساز به طور مکرر ضرب‌های ماتریس‌ـ بردار را انجام می‌دهد. هر ضرب ماتریس‌ـ بردار معادل با اعمال یک گیت است. زمان کل اجرا با $g۲^n$ متناسب است، که g تعداد گیت‌های دو کیوبیتی است. برای سرعت‌بخشیدن به شبیه‌سازی از ادغام گیت‌ها، دستورالعمل‌های AVX/FMA برای تبدیل برداری و OpenMP برای چندرشته‌ای کردن فرایند استفاده می‌شود [۲۵].

رفتار این شبیه‌ساز در شکل‌های ۱۲.۴ و ۱۳.۴ نمایان است. برتری این شبیه‌ساز در شبیه‌سازی تعداد بیش‌تری از کیوبیت‌ها است. برای مثال الگوریتم Deutsch–Jozsa را هیچ‌کدام از شبیه‌سازهای گفته شده نتوانستند تا ۱۹ کیوبیت شبیه‌سازی کنند.

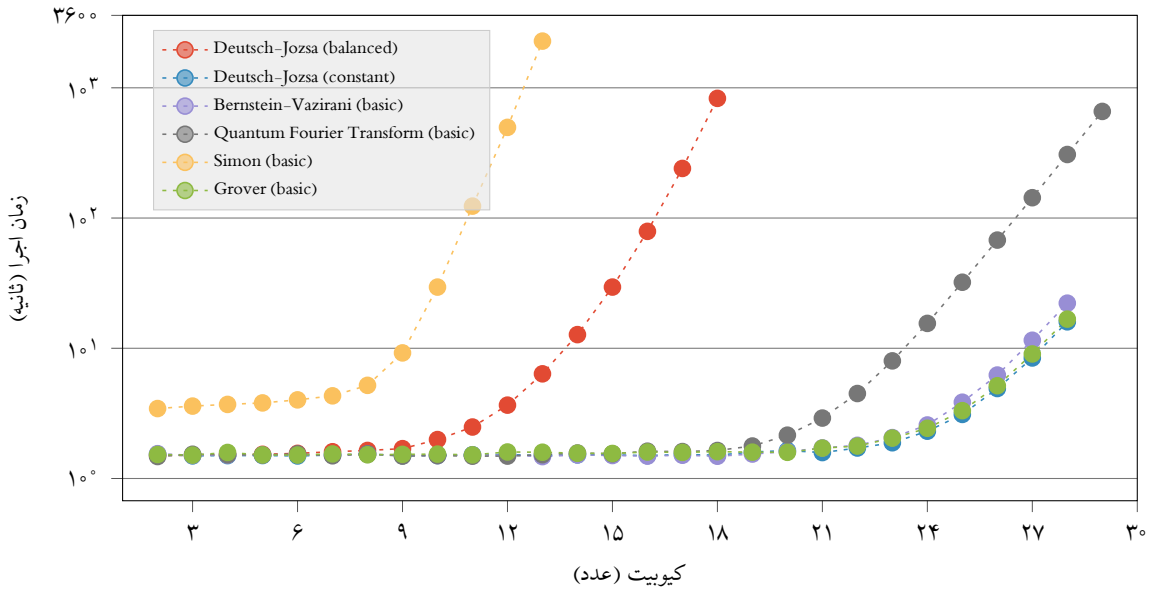
عملکرد این شبیه‌ساز در اجرای الگوریتم Shor در جدول ۵.۴ گزارش شده است. در مقایسه با دیگر شبیه‌سازها، زمان اجرا و حافظه‌ی بیشتری را مصرف کرده است اما این تفاوت جزئی است.

عدد مرکب	تعداد کیوبیت	میانگین زمان اجرا (ثانیه)	میانگین حافظهٔ مصرفی (مگابایت)
۱۵	۱۵	۵.۲۵۱۰۰۶	۱۵۸.۴۰۵۴۶۹
۳۵	۲۱	۵.۹۳۹۱۵۷	۱۵۸.۴۰۷۱۱۸
۷۷	۲۴	۶.۵۶۰۰۵۳	۱۵۸.۸۹۶۹۹۸
۱۴۳	۲۷	۹.۴۰۳۶۸۲	۱۶۰.۲۵۴۸۳۰
۴۳۷	۳۰	N/A	N/A

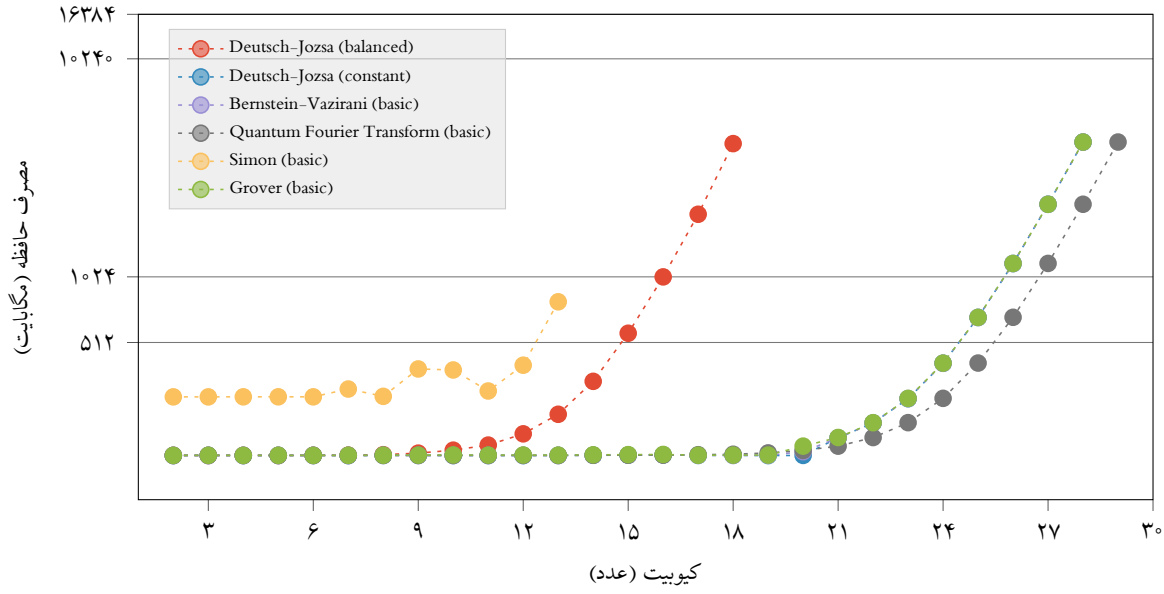
جدول ۵.۴: جدول اطلاعات مربوط به اجرای الگوریتم Shor با Qsim Simulator

QSimh

این یک شبیه‌ساز ترکیبی Schrödinger–Feynman است که شبکهٔ هندسی چیدمان کیوبیت‌ها را به دو قسمت تقسیم می‌کند که از تجزیهٔ اشمیت برای تجزیهٔ گیت‌های دو کیوبیتی در جداسازی استفاده می‌شود. روش به این صورت است که اگر رتبهٔ اشمیت هر گیت m و تعداد گیت‌ها در جداسازی k



شکل ۱۲.۴: نمودار زمان اجرا بر حسب تعداد کیوبیت الگوریتم‌های مختلف در QSim Simulator



شکل ۱۳.۴: نمودار مصرف حافظه بر حسب تعداد کیوبیت الگوریتم‌های مختلف در QSim Simulator

باشد، در آن صورت m^k مسیر وجود دارد. برای شبیه‌سازی یک مدار با بهترین نتیجهٔ شبیه‌سازی از لحاظ شباهت با خروجی موردنظر، باید تمام m^k مسیرها شبیه‌سازی شده و نتایج جمع شوند. زمان کل اجرا با $(۲^{n_۱} + ۲^{n_۲})m^k$ متناسب است، جایی که $n_۱$ و $n_۲$ تعداد کیوبیت‌ها در بخش اول و دوم هستند. مسیرهای شبیه‌سازی شده مستقل از یکدیگر هستند و می‌توانند به‌سادگی موازی‌سازی شوند تا بر روی ابررایانه‌ها یا در مراکز پردازش سریع اجرا شوند. می‌توان شبیه‌سازی‌ها را بادقت کمتر شبیه‌سازی، تنها با جمع‌کردن بر روی یک بخش از تمام مسیرها اجرا کرد.

به‌علاوه، در این شبیه‌ساز، یک روش بررسی دوسطحی برای بهبود عملکرد استفاده می‌شود. فرض کنید k گیت در یک قسمت از قسمت‌های تقسیم‌شده وجود دارد. ما آن‌ها را نیز به سه قسمت تقسیم می‌کنیم. در واقع $p + r + s = k$، که در آن p تعداد گیت‌های «پیشوند»، r تعداد گیت‌های «ریشه» و s تعداد گیت‌های «پسوند» است. اولین سطح بررسی پس از اعمال همهٔ گیت‌ها و از جمله گیت‌های پیشوند اجرا می‌شود و دومین سطح بررسی پس از اعمال همه گیت‌ها و گیت‌های ریشه اجرا می‌شود.

در واقع با تقسیم‌بندی این چنینی، به‌جای حل مسئله در ابعاد بزرگ‌تر با تقسیم آن به مسائل کوچک‌تر، مسئله قابل حل خواهد شد [۲۶].

از آن جایی که پیاده‌سازی الگوریتم‌ها در این شبیه‌ساز متفاوت از شبیه‌سازهای ذکرشده بوده است، محک الگوریتم‌ها روی این شبیه‌ساز صورت نپذیرفته است و تنها به بیان ویژگی‌های آن بسنده شده است.

بحث و نتیجه‌گیری

با توجه به شکل ۱۴.۴ مقایسهٔ شبیه‌سازها در بررسی زمان اجرای الگوریتم‌ها واضح‌تر است. به صورت کلی شبیه‌ساز QasmSimulator (DDSIM) عملکرد بهتری داشته است اما با توجه به ضعف ساختاری و آن چه که در بخش‌های قبل گفته شد، پیاده‌سازی الگوریتم Simon برای آن ممکن نبود

در نتیجه عملکرد آن در اجرای این الگوریتم مشخص نیست. شبیه‌ساز Qsim به صورت کلی عملکرد ضعیف‌تری نسبت به دیگر الگوریتم‌ها داشته است چرا که در اجرای اکثر الگوریتم‌ها زمان اجرای بیش‌تری را صرف کرده است.

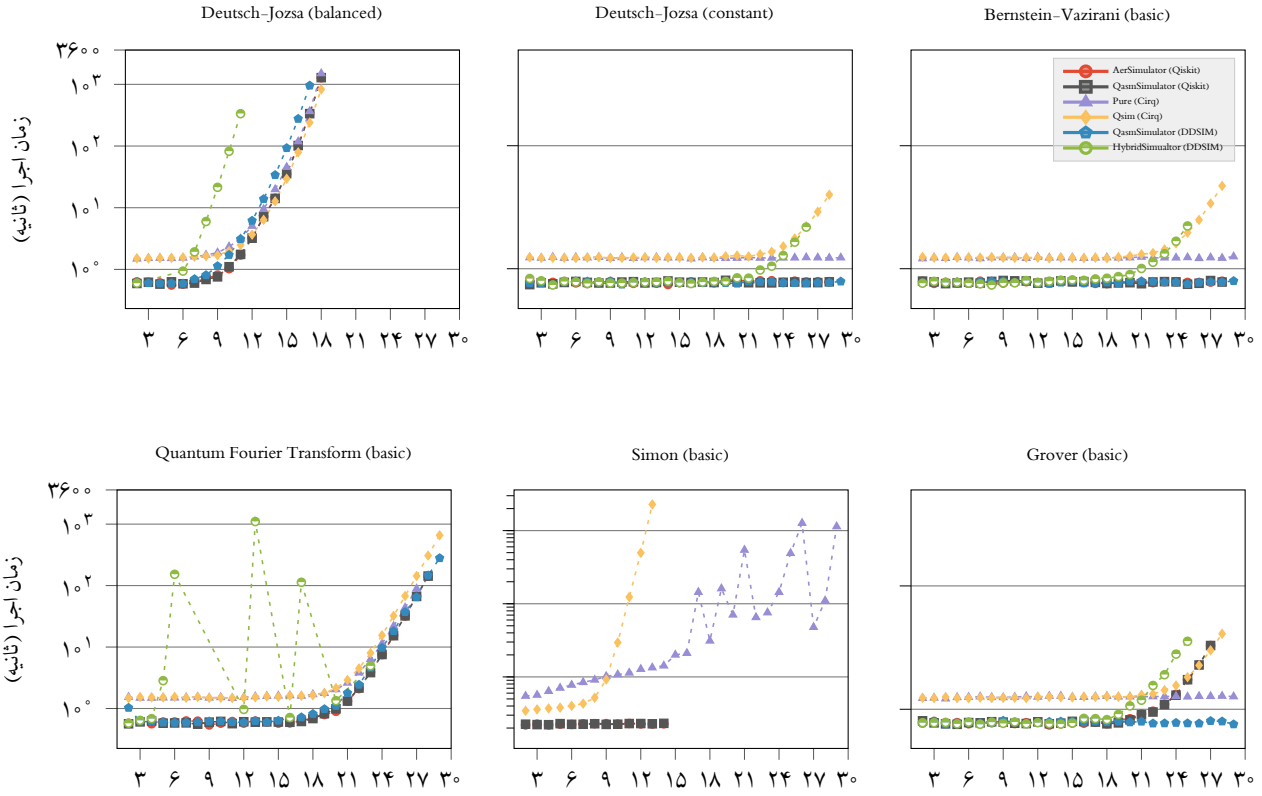
در مقایسه‌ی میزان مصرف حافظهٔ نیز شبیه‌ساز QasmSimulator (DDSIM) بهترین عملکرد را داشته است. رفتار شبیه‌سازها در مصرف حافظه و زمان اجرا تقریباً یکسان بوده است.

با توجه به شواهد موجود، شبیه‌سازی که با استفاده از نمودارهای تصمیم با اعمال مرحله‌ای گیت‌های موردنظر الگوریتم‌ها عملیات شبیه‌ساز را انجام داده است، بهترین عملکرد را به ثبت رسانده است. با این تفاسیر، می‌توان به این نتیجه رسید که دیگر روش‌های شبیه‌سازی پیچیده‌تر که توسط دیگر پژوهش‌گران مورد بررسی قرار گرفته است لزوماً عملکرد بهتری نداشته‌اند و پیچیدگی بیش‌تر باعث بهبود عملکرد نشده است. دیگر نتیجه‌ای که می‌توان گرفت، با مقایسه با نتایج حاصل‌شده در [۱] می‌توان دریافت که برای حل مسئلهٔ شبیه‌سازی کوانتومی، آن چه که در [۱۴] به عنوان یک مسئلهٔ بسیار دشوار مطرح شده است، افزایش توان محاسباتی تأثیر به نسبت مناسبی نخواهد داشت. برای مثال با دوبرابر کردن توان سخت‌افزاری، زمان اجرای شبیه‌سازها نصف نخواهد شد. تحقیق و توسعه شبیه‌سازهایی نظیر QasmSimulator (DDSIM)، کمک شایانی به پیشرفت در این حوزه خواهد بود.

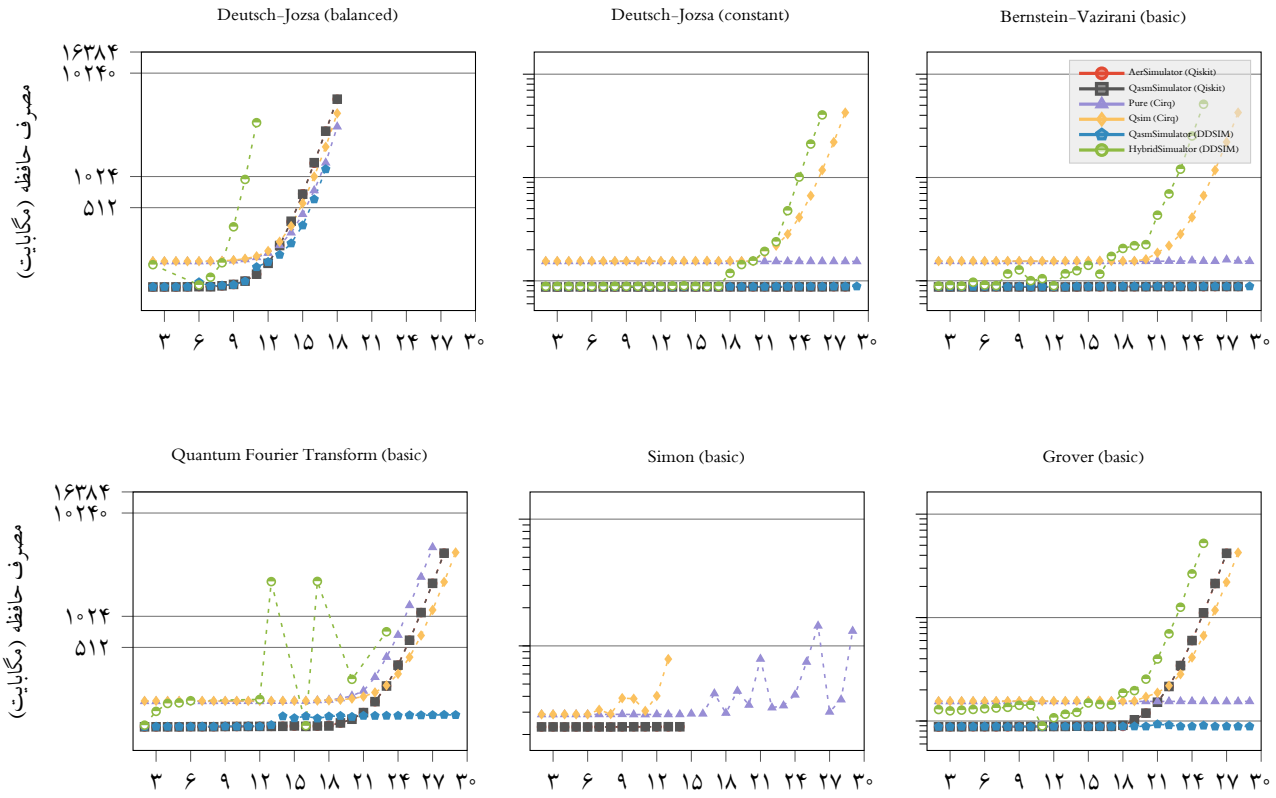
This part should be checked.

پژوهش‌های آتی

محک شبیه‌سازها فرایندی زمان‌بر و پیچیده است چرا که هر شبیه‌ساز از روش، زبان برنامه‌نویسی، تناسب با سخت‌افزار، رویکردهای بهینه‌سازی متفاوت استفاده کرده است. این که بتوان با وجود تمام این تفاوت‌ها یک ابزار و یک چارچوب معین تعیین کرد، هدف نهایی پژوهش‌های نظیر پژوهش ما است. به عنوان پژوهش آتی می‌توان بر روی طراحی دقیق یک مترجم جامع که بتواند فقط با یک بار پیاده‌سازی الگوریتم‌های محک، آن‌ها را به زبان‌های قابل فهم انواع شبیه‌سازها ترجمه کند، تمرکز کرد.



شکل ۱۴.۴: نمودار زمان اجرای الگوریتم‌های بررسی‌شده بر روی تمامی شبیه‌سازها.
توضیحات: دلیل عدم مشاهدهٔ واضح نمودار بعضی از شبیه‌سازها، هم‌پوشانی و شباهت رفتاری آن‌ها با دیگر شبیه‌سازهاست. محور افقی نشان‌دهندهٔ تعداد کیوبیت است.



شکل ۱۵.۴: نمودار مصرف حافظهٔ اجرای الگوریتم‌های بررسی‌شده بر روی تمامی شبیه‌سازها.
توضیحات: دلیل عدم مشاهدهٔ واضح نمودار بعضی از شبیه‌سازها، هم‌پوشانی و شباهت رفتاری آن‌ها با دیگر شبیه‌سازهاست. محور افقی نشان‌دهندهٔ تعداد کیوبیت است.

به علاوه، شبیه‌سازهایی که از روش نمودار تصمیم برای شبیه‌سازی بهره برده‌اند، قابلیت نویدبخشی از خود نشان داده‌اند. توسعه و بهبود آن‌ها می‌تواند پژوهش ارزشمندی را به ارمغان آورد.

This part should be checked.

کتاب‌نامه

[1] Jamadagni, Amit, Läuchli, Andreas M., and Hempel, Cornelius. Benchmarking quantum computer simulation software packages, January 2024.

[2] Lubinski, Thomas, Johri, Sonika, Varosy, Paul, Coleman, Jeremiah, Zhao, Luning, Necaise, Jason, Baldwin, Charles H., Mayer, Karl, and Proctor, Timothy. Application-Oriented Performance Benchmarks for Quantum Computing, January 2023.

[3] Li, Ang, Stein, Samuel, Krishnamoorthy, Sriram, and Ang, James. QASMBench: A Low-level QASM Benchmark Suite for NISQ Evaluation and Simulation, May 2022. arXiv:2005.13018 [quant-ph].

[4] Michielsen, Kristel, Nocon, Madita, Willsch, Dennis, Jin, Fengping, Lippert, Thomas, and De Raedt, Hans. Benchmarking gate-based quantum computers. *Computer Physics Communications*, 220:44–55, November 2017.

[5] Wright, K., Beck, K. M., Debnath, S., Amini, J. M., Nam, Y., Grzesiak, N., Chen, J.-S., Pisenti, N. C., Chmielewski, M., Collins, C., Hudek,

K. M., Mizrahi, J., Wong-Campos, J. D., Allen, S., Apisdorf, J., Solomon, P., Williams, M., Ducore, A. M., Blinov, A., Kreikemeier, S. M., Chaplin, V., Keesan, M., Monroe, C., and Kim, J. Benchmarking an 11-qubit quantum computer. *Nature Communications*, 10(1):5464, November 2019.

[6] Koch, Daniel, Martin, Brett, Patel, Saahil, Wessing, Laura, and Alsing, Paul M. Demonstrating NISQ era challenges in algorithm design on IBM's 20 qubit quantum computer. *AIP Advances*, 10(9):095101, September 2020.

[7] Mills, Daniel, Sivarajah, Seyon, Scholten, Travis L., and Duncan, Ross. Application-Motivated, Holistic Benchmarking of a Full Quantum Computing Stack. *Quantum*, 5:415, March 2021.

[8] Cornelissen, Arjan, Bausch, Johannes, and Gilyén, András. Scalable Benchmarks for Gate-Based Quantum Computers, April 2021.

[9] Quantum Computing and Shor's Algorithm.

[10] Wong, Thomas G. *Introduction to classical and quantum computing*. Rooted Grove, Omaha, Nebraska, 2022.

[11] Young, Kieran, Scese, Marcus, and Ebnenasir, Ali. Simulating Quantum Computations on Classical Machines: A Survey, November 2023.

[12] Acuaviva, Arturo, Aguirre, David, Peña, Rubén, and Sanz, Mikel. Benchmarking Quantum Computers: Towards a Standard Performance Evaluation Approach, July 2024. arXiv:2407.10941 [quant-ph].

[13] Nielsen, Michael A. and Chuang, Isaac L. *Quantum computation and quantum information*. Cambridge University Press, Cambridge ; New York, 10th anniversary ed ed. , 2010.

[14] Xu, Xiaosi, Benjamin, Simon, Sun, Jinzhao, Yuan, Xiao, and Zhang, Pan. A Herculean task: Classical simulation of quantum computers, February 2023.

[15] Dalzell, Alexander M., McArdle, Sam, Berta, Mario, Bienias, Przemyslaw, Chen, Chi-Fang, Gilyén, András, Hann, Connor T., Kastoryano, Michael J., Khabiboulline, Emil T., Kubica, Aleksander, Salton, Grant, Wang, Samson, and Brandão, Fernando G. S. L. Quantum algorithms: A survey of applications and end-to-end complexities, October 2023.

[16] Li, Jinyang, Li, Ang, and Jiang, Weiwen. QuApprox: A Framework for Benchmarking the Approximability of Variational Quantum Circuit, February 2024. arXiv:2402.08261 [quant-ph].

[17] Miessen, Alexander, Egger, Daniel J., Tavernelli, Ivano, and Mazzola, Guglielmo. Benchmarking digital quantum simulations and optimization above hundreds of qubits using quantum critical dynamics, April 2024. arXiv:2404.08053 [cond-mat, physics:quant-ph].

[18] Barbaresco, Frédéric, Rioux, Laurent, Labreuche, Christophe, Nowak, Michel, Olivier, Noé, Nicolazic, Damien, Hess, Olivier, Guilmin, Anne-Lise, Wang, Robert, Sassolas, Tanguy, Louise, Stéphane, Snizhko, Kyrylo,

Misguich, Grégoire, Auffèves, Alexia, Whitney, Robert, Vergnaud, Emmanuelle, and Schopfer, Félicien. BACQ – Application-oriented Benchmarks for Quantum Computing, March 2024. arXiv:2403.12205 [quant-ph].

[19] Javadi-Abhari, Ali, Treinish, Matthew, Krsulich, Kevin, Wood, Christopher J., Lishman, Jake, Gacon, Julien, Martiel, Simon, Nation, Paul D., Bishop, Lev S., Cross, Andrew W., Johnson, Blake R., and Gambetta, Jay M. Quantum computing with Qiskit, June 2024. arXiv:2405.08810 [quant-ph].

[20] AerSimulator - Qiskit Aer 0.15.0.

[21] Zulehner, Alwin and Wille, Robert. Advanced Simulation of Quantum Computations. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 38(5):848–859, May 2019. Publisher: Institute of Electrical and Electronics Engineers (IEEE).

[22] Hillmich, Stefan, Markov, Igor L., and Wille, Robert. Just Like the Real Thing: Fast Weak Simulation of Quantum Computation. in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, San Francisco, CA, USA, July 2020. IEEE.

[23] Hybrid Schrödinger-Feynman Simulator.

[24] Simulation | Cirq.

[25] Smelyanskiy, Mikhail, Sawaya, Nicolas P. D., and Aspuru-Guzik, Alán. qHiPSTER: The Quantum High Performance Software Testing Environment, May 2016. arXiv:1601.07195 [quant-ph].

[26] Markov, Igor L., Fatima, Aneeqa, Isakov, Sergei V., and Boixo, Sergio. Quantum Supremacy Is Both Closer and Farther than It Appears, July 2018.

Abstract

The english abstract will be written in this section.

This part should be written.

Keywords: *Keyword1, Keyword2, Keyword3*

Institute for Advanced Studies
in Basic Sciences
Gava Zang, Zanjan, Iran

Benchmark Quantum Circuit Simulators

Master's Thesis

Ahmad Mahmoodian Darvishani

Supervisors: Dr. Ali Ebnenasir

Dr. Mehdi Vasighi

Advisor: Dr. Mansour Davoodi Monfared

December 5, 2024