Quantum Economic Advantage Calculator

This framework is designed to allow users to explore different combinations of algorithmic problems and quantum hardware. Users can freely deviate from known projections and default parameters to derive their own insights as to the general timeline of when certain scenarios may become economically advantageous to run on a quantum machine.

Models

A model in the framework refers to an instance of a specific problem-hardware combination being examined. Each model tracks its own parameters and displays its own graphs.

Making Changes

Upon opening the framework, there will be a template model already created which is ready to edit. Changing any input to the model will have its effects immediately displayed on the corresponding graphs (unless the parameter was changed in a popup menu in which case the user must hit save).

Users can add new models by creating from the template model or duplicating existing models. The latter option is useful if users would like to see how slight changes would affect the graphs' plots and view the results at the same time. Users can also freely delete models.

Simple vs Advanced View

There are two ways of displaying a model on the user interface. The first is a simple view in which only the two major inputs of problem and hardware can be seen and selected. All the graphs are still visible and this view allows for easy comparison between them across multiple models. The second option is an advanced view in which the user is able to freely manipulate all variables associated with the computation of the graphs. It is more complex and may overwhelm the user if they are not too familiar with quantum hardware so the framework shows the simple view by default and allows users to choose their own level of involvement with the additional parameters.

Something worth noting is that two separate models could have the same options selected for problem and hardware, yet have very different choices for the remaining parameters. This could result in two models with the same options visible for their simple views but with drastically different graphs.

Inputs

Each input into the models can be classified as either related to the problem or the hardware. We will refer to the problem and roadmap inputs themselves as major inputs and the inputs dependent on them as minor inputs. Each selectable

option for the major inputs has default values for all of its corresponding minor inputs; as a result, changing a model's major inputs will automatically override the values of all its associated minor inputs. Changing a minor input has no effect on the other parameters.

Problem Inputs

Problem The problems contain the runtimes of the best known classical and quantum algorithms to solve them to be compared in the model. We will refer to these runtimes as $C(n)$ and $Q(n)$ respectively. #### Connectivity Penalty The connectivity penalty is a function which acts as a multiplicative factor to the quantum runtime being compared. If this value is set to "None", then $Q(n) * \text{Penalty}(n)$ will be evaluated as $Q(n)$ in the resulting expressions. This slowdown is inspired by how quantum algorithms actually face slowdowns when implemented on physical circuits in practice. #### Qubit To Problem Size Qubit to Problem Size (QPS) is a function which maps logical qubits to the maximum problem size achievable with said amount of qubits. The inverse of this function (QPS^{-1}) is also used in some calculations. This is still a minor input as different problems may deviate from a typical mapping of $2^{\#\text{ of qubits}}$.

Hardware Inputs

Roadmap This is the major input for the hardware. The roadmap specifies physical qubit projections with associated years for each target. Each projection in the framework is pulled directly from the hardware providers' published timelines, but users are free to edit the roadmap as well. They can add/remove qubit-year pairs from the projections in addition to changing the type of extrapolation that is done in between data points on the plots. #### Hardware Slowdown Hardware slowdown (hws) represents how many classical operations could be performed in the time it takes the hardware to perform a single quantum operation. Users can set this value directly or instead choose to edit each of the individual factors whose product is the slowdown:

$$\text{Speed Ratio} * \text{Gate Overhead} * \text{Algorithmic Constant} = \text{Hardware Slowdown}$$

The speed ratio represents the ratio of how much faster a classical computer is than a quantum computer. Much like hardware slowdown, the user can choose to manually edit its individual components:

$$\text{Average Quantum Operation (ns)} * \text{Classical CPU Rate (GHz)} = \text{Speed Ratio}$$

(Note that we're multiplying the terms instead of dividing them and still achieving the Classical/Quantum Speed Ratio. This is because of the choice of GHz and ns as units which allows for this simpler equation.)

The hardware slowdown also features another input: the quantum improvement rate (qir). The framework assumes the hardware slowdown improves yearly by

a factor of the improvement rate:

$$\text{hws}_{t+1} = \text{hws}_t * (1 - \frac{qir}{100})$$

(Note it is possible for the improvement rate to be negative implying that quantum devices are becoming relatively slower instead.)

Physical to Logical Qubit Ratio This final input is the physical to logical qubit ratio (PLQR) which represents how many physical qubits are needed to encode a single logical qubit. And similar to hardware slowdown, the ratio also features a ratio improvement rate (rir). How the ratio changes with its rate over time is exactly the same as the hardware slowdown equation shown above.

Graphs

Minimum Problem Size for Quantum Algorithmic Advantage

This graph plots both the classical and quantum runtimes with problem size on the x-axis and classical time steps on the y-axis. It's purpose is to visualize the minimum problem size such that the quantum algorithm (including the penalty and hardware slowdown) would take less time to run than the classical counterpart. We denote said problem size with n^* which occurs when the runtime trajectories intersect in the graph. Or mathematically, n^* is the smallest problem size (n) such that

$$C(n) = Q(n) * \text{Penalty}(n) * \text{hws}$$

Economic Advantage of Quantum Computing This graph answers perhaps the most important information in the framework: when will the problem become economically advantageous to run on quantum hardware? The x- and y-axes are time (in years) and problem size respectively. The two lines plotted are called the feasibility line (Feas(t)) and the quantum advantage line (Adv(t)).

The feasibility line represents how the maximum achievable problem size on the hardware increases over time. That is, all points which lie below this curve on the graph could have their problem sizes executed on the quantum hardware (although it might not yet be efficient to do so). This is influenced by the roadmap's trajectory of physical qubits, the value/rate of the physical to logical qubit ratio, and the function mapping logical qubits to problem sizes. If we denote the amount of physical qubits expected from a roadmap at year t as the function $R(t)$, we can express Feas(t) as

$$\text{Feas}(t) = \text{QPS}(\frac{R(t)}{PLQR * (1 - \frac{rir}{100})^{t-2024}})$$

(Note that the year 2024 is included into the equation because we assume PLQR is the value for the current year. The equation could be made more general by replacing it with an additional variable denoting the current year.)

The quantum advantage line represents how the minimum problem size to achieve advantage diminishes over time (although the size could actually increase if the quantum improvement rate is negative). All points which lie above this line would be advantageous to run on a quantum device (although it might not yet be possible to do so). Similar to how the first graph finds n^*, this function plots the value of n^* as the hardware slowdown changes over time.

$$\text{Adv}(t) = n \mid C(n) = Q(n) * \text{Penalty}(n) * \text{hws} * (1 - \frac{qir}{100})^{t-2024}$$

The area above the feasibility represents problem sizes that are still too large for the progress of quantum devices in said year. The area below the advantage line represents problem sizes that are too small to actually achieve a practical advantage for said year. Once these two lines intersect, the area contained between them then represents the space of problem sizes which are both feasible and advantageous for quantum computers. This intersection occurs at the year t^* such that $\text{Feas}(t) = \text{Adv}(t)$, or equivalently:

$$n \mid (C(n) = Q(n) * \text{Penalty}(n) * \text{hws} * (1 - \frac{qir}{100})^{t-2024}) = \text{QPS}(\frac{R(t)}{PLQR * (1 - \frac{rir}{100})^{t-2024}})$$

Logical Qubit Quantum Economic Advantage

This graph provides practically the same information as the previous one except in terms of different units. Whereas the previous graph plotted problem size over time, this one graphs logical qubits over time. The year marking the start of quantum economic advantage will remain exactly the same, but users are now able to see the trajectories in terms of logical qubits if they were more interested in that metric instead. Formally, the feasiblity line plotted would be $\text{QPS}^{-1}(\text{Feas}(t))$ and the advantage line would be $\text{QPS}^{-1}(\text{Adv}(t))$. (Note that the former evaluates to an expression of the form $\text{QPS}^{-1}(\text{QPS}(\dots))$ which mathemetically makes the two function calls useless, however since the $\text{QPS}(\dots)$ must be computed regardless, both computations are still performed in the framework. The results are the same to several decimal points as if no transformation had been done to the data.)