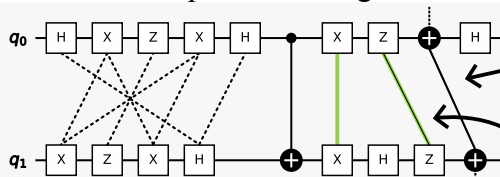


1. Create set of potential merges

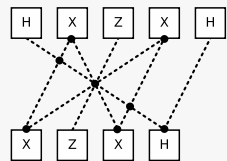


The set of potential merges is the bipartite graph between logical gates of the same type, partitioned by dependencies. In the example above, many single qubit gates can merge with each other but cannot with the gates after the CNOT.

In lieu of doing extensive path analysis, we make the assumption that two-qubit gates that leave the qubit pair may have a path, and as a consequence, enforced dependency between them.

Potential merges that have no conflicts can be made for free with a very low likelihood of negatively affecting encoded circuit quality.

2. Identify conflicts through line crossings

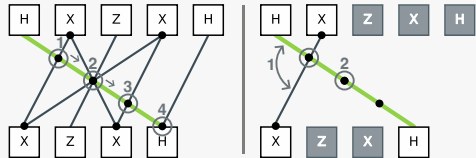


8 $\diamond$ conflicts to resolve
Note: one $\diamond$ represents a choice among possible merges.

Conflicts can be calculated efficiently through line-crossing algorithms such as Bentley-Ottman. Potential merges sharing the same gate are considered a conflict as well. For example, X_1 and X_3 share the gate X_2 and are conflicted.

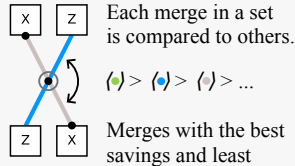
3. Iterate through conflict set

a) for each potential merge, analyze set of conflicts



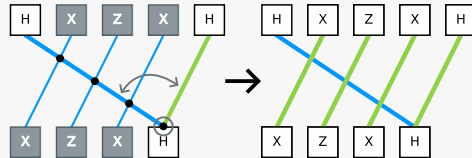
Iteration takes place left to right. It begins with the first potential merge, a Hadamard merge, and progresses through its set of conflicts then moves to the next, X.

b) comparatively group merges



Conflicts are added to the best available group. In this case, green.

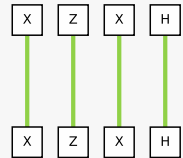
c) updated groups backpropagate



Previously scored merges in a conflict set are updated if the current scored merge is moved to a higher group. In this case, blue X and Z merges are moved to green.

4. Return best

groups are assessed.



In this example, green is the group with the most effective merges.