Tal’s set-optimization idea:

Usually, VQAs work like this:

* Start with a parametrized circuit (which spans the entire space), and a **random** initial guess of parameters .
* Run the circuit, and get an initial cost evaluation (based on ).
* Use an optimizer to calculate the new parameters: .
* Repeat until convergence of the optimizer.

Tal’s idea (in its simplest form) was to work like this:

* Start with a parametrized circuit (which spans the entire space), and a **random** initial guess of parameters .
* Run the circuit, and get an initial cost evaluation (based on ).
* Start with a parametrized circuit (which spans the entire space), and a **random** initial guess of parameters .
* Run the circuit, and get an initial cost evaluation (based on ).
* Use a SPECIAL optimizer, that takes the TWO evaluations, to calculate the new parameter: .
* Repeat until convergence of the optimizer, using multiple values for .
  + Idea 1: use ALL thetas (the optimizer needs to be able to work on a variable amount of cost function evaluations).
  + Idea 2: use the last/best two thetas to calculate the next one (a-la-momentum GD).

Tal’s MUB-VQE idea:

* Start with a parametrized circuit (which spans the entire space), and an initial choice of parameters , **yielding an MUB state** .
* Run the circuit, and get an initial cost evaluation (based on ).
* Start with a parametrized circuit (which spans the entire space), and an initial choice of parameters **yielding an MUB state** .
* Run the circuit, and get an initial cost evaluation (based on ).
* In the case of a small number of qubits, repeat for all MUB states.
* Use a SPECIAL optimizer, that takes ALL evaluations, to calculate new parameters: .
* Repeat until convergence of the optimizer:
  + Idea 1: use ALL thetas (the optimizer needs to be able to work on a variable amount of cost function evaluations).
  + Idea 2: use a constant amount of cost function evaluations using a new parameter and the best old parameters.

Note: if working with a larger # of qubits, repeating for all MUB states can be replaced with:

* randomly-chosen MUB states
* purposefully-chosen MUB states

Tal’s idea (Ittay’s variant/phrasing) was to work like this:

* Start with a parametrized circuit (which spans the entire space), and TWO **random** initial guesses of parameters: and .
* Run the circuit, and get TWO initial cost evaluations: one based on and one based on .
* Use a SPECIAL optimizer, that takes the TWO evaluations, to calculate the new parameters: and .
* Repeat until convergence of the optimizer.

Note: all ideas that use TWO cost evaluations can be substituted to cost evaluations.