stabilizer-decomp-search User Manual

Eva Feng

August 2021

This user manual aims to provide installation instructions, system overview and command documentation for the stabilizer-decomp-search package.

1 Installation

1.1 Software Requirements

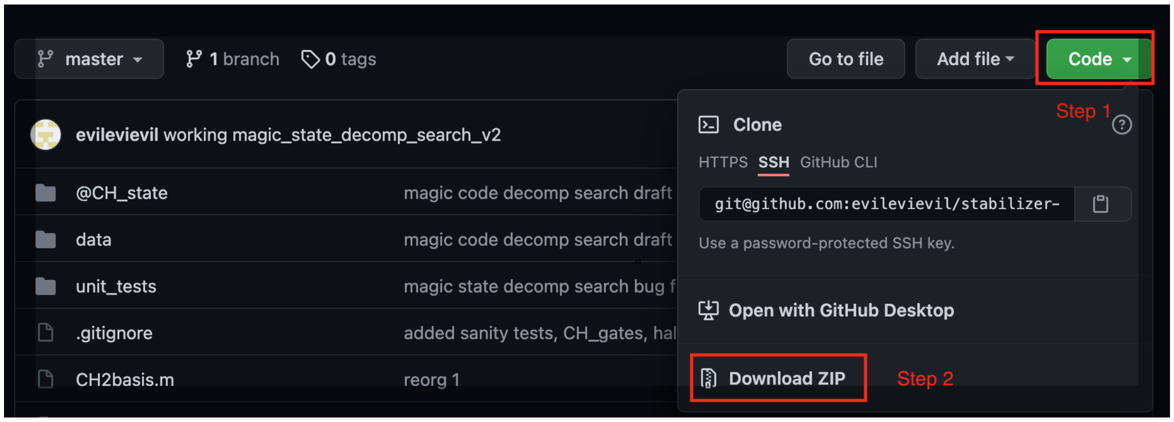
The stabilizer-decomp-search is written completely in Matlab. Please ensure your Matlab version is at least R2021a. Older Matlab versions are likely compatible, but this is not guaranteed.

1.2 Getting Your Local Copy

To get a copy of the stabilizer-decomp-search package on your local machine, you can either download the package as a zip file or clone the git repository. To do either of these, first go to the github repository:

`https://github.com/evilevievil/stabilizer-decomp-search`

Download ZIP 1. Click on the green 'Code' button. 2. Click on 'Download ZIP' button in the drop down.



evilevievil working magic_state_decomp_search_v2

@CH_state magic code decomp search draft
data magic code decomp search draft
unit_tests magic state decomp search bug f
.gitignore added sanity tests, CH_gates, hal
CH2basis.m reorg 1

Figure 1: Download ZIP

Clone Repo See the following github page on how to clone a repo:

`https://docs.github.com/en/github/creating-cloning-and-archiving-repositories/cloning-a-repository-from-github/cloning-a-repository`

1.3 Open and Run

Open the directory (folder) containing stabilizer-decomp-search in Matlab and add all files and folders to Matlab path. The package is now ready to go!

2 System Overview

2.1 File Organization

\stabilizer-decomp-search (root directory)

↪ \@CH_state (CH-form class definition and methods)
↪ \data (found stabilizer decompositions)
↪ \unit_tests (tests for major subroutines)
— CH utility functions:
· superpos2circuit.m
· CH2basis.m
· CH_CH_inner_product.m
· CH_basis_inner_product.m
— general utility functions:
· approx.m
· bitsum.m
· const.m
· magic_state_vec.m
· parity.m
· tensor_exp.m
— objective value functions:
· reverse_format_amp.m
· CH_decomp_project.m
· CH_decomp_project_memoize.m
— magic state generator functions:
· add_generator.m
· get_commuter.m
· generator_search_v2.m
— search functions:
· fixed_rank_stab_decomp_search.m
· magic_code_decomp_search.m (WIP)
· magic_state_decomp_search_v2.m
— example scripts:
· profile_time.m

2.2 References and Design Decisions

CH-form We choose to represent all stabilizer states using CH-form because it requires less space and is easier to implement. CH-forms are stored using bit matrices and bit arrays for improved efficiency. The CH-form class and its utility functions are implemented following Section 4.1 in 'Simulation of quantum circuits by low-rank stabilizer decompositions '. [1]

Objective Function Evaluation The objective function $< \text{target}|\Pi_{\text{stab decomp}}|\text{target} >$ is evaluated using Gram matrix. This conveniently allows us to use memoization for objective function evaluation after each Pauli walk step.

Simulated Annealing Search The simulated annealing search meta-heuristic is implemented following Appendix B in 'Trading classical and quantum computational resources' [2]. The same numerical search was previously implemented in python and C using state vector representation to generate upper bounds in Figure 6 in 'Simulation of quantum circuits by low-rank stabilizer decompositions'. Our new Matlab version of the simulated annealing search uses CH-form and is able to reproduce all previous upper bounds.[2, 1]

Magic Code State Search In effort to find lower-rank decompositions, we enhance the simulated annealing search using magic code states proposed in 'Improved upper bounds on the stabilizer rank of magic states'. In addition to the original simulated annealing search, the magic code state search also randomly selects generators, corresponding to random magic code states, which can help convert between search spaces to improve stabilizer rank. This enhancement is inspired by low-rank decomposition for the 6-qubit cat state in 'Improved upper bounds on the stabilizer rank of magic states' [3]. Currently, the magic code state search does not achieve, but is close to achieving the state-of-art upper bound.

3 Command Documentation

Although the main purpose of this package is to search for low-rank stabilizer decompositions, it also includes several useful standalone classes and functions. This section provides documentation and example commands for these classes and functions.

3.1 Configuration

To configure the max number of qubits, first go to 'const.m' and update init_max_qubits to 8 or 16 or 32. Then, reopen the stabilizer-decomp-search directory (or simply reopen Matlab) to apply new config globally.

3.2 Running Unit Tests

Unit tests are stored under 'unit_tests' folder. Functions are tested using state vector simulations. To run unit test, simply open the desired test file and click 'Run' button in the Matlab console.

3.3 CH-form Utilities

1. Allocate new state in CH-form
   requires: #qubits <= init_max_qubits.

   ```
    stab = CH_state(#qubits)
   ```

2. Initialize allocated state
   requires: type_of_initialization ∈ {'zero' (|00..0>),'rand' (random stab state)}.

   ```
   stab.CH_init('type_of_initialization')
   ```

3. Apply gate
   requires: type_of_gate ∈ {'CXL' (apply control-X gate from the left),'CXR','CZL','CZR','HL','SL','SR'}.
   target_qubits ∈ {[control_bit,target_bit],[trarget_bit,dont_care]}.

   ```
   stab.CH_gate('type_of_gate', target_qubits)
   ```

4. Apply Pauli projector
   CAVEAT: the function normalizes the result for you; no need to renormalize.
   requires: is_neg = positive→0 ; negative→1
   x_bit[i] = 0→I ; 1→X
   z_bit[i] = 0→I ; 1→Z.

   ```
   stab.CH_pauli_proj(is_neg,x_bit,z_bit)
   ```

5. Pretty Printing
   requires: option ∈ {'ch' (print CH-form),'conj' (print conjugated tableaux),'basis' (print state vector)}.

   ```
   state.pp_CH(option)
   ```

6. Compute inner product of standard basis state and stabilizer state
   CAVEAT: Please follow reverse bit string convention for basis state. ie. basis state '1011' should be inputted as '1101' in base 10.

   ```
   CH_basis_inner_product(reversed_basis_state_as_integer,stab)
   ```

7. Compute inner product of two stabilizer states
   CAVEAT: The function conjugates |stab_state1> for you: please input |stab_state1>, |stab_state2> as is.

```
CH_CH_inner_product(stab_state1,stab_state2)
```

8. Compute projection onto a stabilizer decomposition
   CAVEAT: Serves as base case for its memoized version 'CH_decomp_project_memoize'
   requires: type_state_vec is the integer representation of reverse bit string.

   ```
   [projection,memoize_G,memoize_target_inner_prod] =
   CH_decomp_project(target_state_vec,stab_decomp,#qubit,decomp_len)
   ```

3.4 Stabilizer Decomposition Search

1. Make state vector for magic state
   requires: type_of_magic ∈ {'T', 'H', 'catT','r_1_3','12gencat'}.
   notes: 'r_1_3' stands for r(1,3) Reed-Muller code and '12gencat' stands for 12-qubit generalized cat state.

   ```
   state_vec = magic_state_vec(type_of_magic, #qubits)
   ```

2. Search for stabilizer decomp for target_state and target_rank
   notes: Each SA step executes a random Pauli walk for rand_walk_steps.
   The search terminates either when a decomposition is found or when max_SA_steps is reached.

   ```
   fixed_rank_stab_decomp_search(target_state,#qubits,target_rank,
             init_temp_inverse,final_temp_inverse,
             max_SA_steps,rand_walk_steps)
   ```

3. Search for stabilizer decomp for fixed target_rank and magic code state with variable number of generators
   notes: Stabilizer decomposition search performs a generator search when objective value does not improve for max_idle_steps.
   Set max_num_generators = -1 to automatically compute the maximum number of generators to improve the state-of-art stabilizer rank.
   Search repeats generator search for max_generator_search_step times for each fixed generator dimension.

   ```
   magic_state_decomp_search_v2(#qubits,rank,
             init_temp_inverse,final_temp_inverse,
             max_SA_steps,rand_walk_steps,
             max_idle_steps,max_num_generators,max_generator_search_step)
   ```

Acknowledgments

The stabilizer-decomp-search package was developed as a summer Undergraduate Research Assistantship (URA) project funded by the Institute for Quantum Computing (IQC) at the University of Waterloo.

I thank David Gosset, Hakop Pashayan and Hammam Qassim for their support and guidance throughout the stabilizer decomposition search project. I thank Prof. David Gosset for giving me the opportunity to work as an URA at IQC.

References

[1] S. Bravyi, D. Browne, P. Calpin, E. Campbell, D. Gosset, and M. Howard, "Simulation of quantum circuits by low-rank stabilizer decompositions," *Quantum*, vol. 3, p. 181, Sep 2019.

[2] S. Bravyi, G. Smith, and J. A. Smolin, "Trading classical and quantum computational resources," *Physical Review X*, vol. 6, Jun 2016.

[3] H. Qassim, H. Pashayan, and D. Gosset, "Improved upper bounds on the stabilizer rank of magic states," 2021.