Test Report: Kaplan

Jen Garner

April 20, 2019

1 Revision History

Date	Version	Notes
December 11, 2018 (Tuesday)	1.0	first report

2 Symbols, Abbreviations and Acronyms

symbol	**description**
T	Test
SMILES	Simplified molecular-input line-entry system
VMD	Visual Molecular Dynamics
IDE	integrated development environment

Contents

List of Tables

List of Figures

This document discusses the results of the unit testing that took place for the Kaplan program. The testing was based on the UnitVnVPlan document that can be found in the github repository under docs/VnVPlan/UnitVnVPlan. https://github.com/PeaWagon/Kaplan. This report is a partner to the System VnV Report that is located in the docs/VnVReport/SystVnVReport directory.

3 Unit Testing

During testing, an assertion error was raised when the num_atoms and size of the parser.coords attribute was found to be non-equal. This led to the discovery that SMILES strings are case sensitive. The verification of the mol_input module had to be changed to accommodate this discovery. Previously, all input was changed to lowercase to avoid case-sensitivity-related errors (such as trivial matching errors raised by an incorrect case). Now, all input except the SMILES string is changed to lowercase.

Also during testing, an error was found in the vetee program for the write_xyz method of the Xyz class. The method was missing a newline character, and so all of the coordinates were being incorrectly written to one line. A new issue was raised in the vetee repository on github, and the co-author Kumru Dikmenli was able to make a new pypi package with the newline character included. Now the output for Kaplan is computer-readable (tested using Visual Molecular Dynamics - VMD). See Figure 1 for an example of the geometry generated with a kaplan output xyz file.

When the tests were run, there would be constant convergence issues from the psi4 package. To get around this issue, if a calculation did not work (regardless of the error message), the energy would be set to zero. Therefore, a more in depth investigation is needed into the types of errors raised by psi4, and when they might actually be errors (instead of just convergence problems caused by really poor geometry specification).

From the Travis build (see https://travis-ci.org/PeaWagon/Kaplan/builds/465876614), the unit tests that have been written thus far have all passed (see Table 2). Since not all of the unit tests have been written, this measure of success is minimal (see the Appendix 8 for details on which exact tests were run). The linters, on the other hand, are causing the Travis build to fail as they are not satisfied with the current state of the code. Mainly, the errors are coming from incorrect whitespace and missing docstrings. The

author plans on addressing these problems by opening the code using the PyCharm IDE (integrated development environment). The author can use PyCharm to automatically highlight all of the linting errors such that they can be fixed. A search will also be done to find such tools online (so they don't have to be fixed manually) if PyCharm cannot automatically fix the linting problems. The docstrings for the modules also have to be written.

4 Automated Testing

An example of some of the output can be found in the kaplan/kaplan_output directory of the repository on github https://github.com/PeaWagon/Kaplan. This output was generated by calling the run_kaplan function in the gac module (genetic algorithm control module) with the inputs for example2_mol_input_file.txt and example2_ga_input_file.txt (which are located in the kaplan/test/testfiles directory).

Based on visual inspection alone of the xyz coordinates, the geometry is clearly non-optimal. Since this run only had 5 mating events during which to find the optimal solution, the program was not run for long enough such that a reasonable geometry could be found.

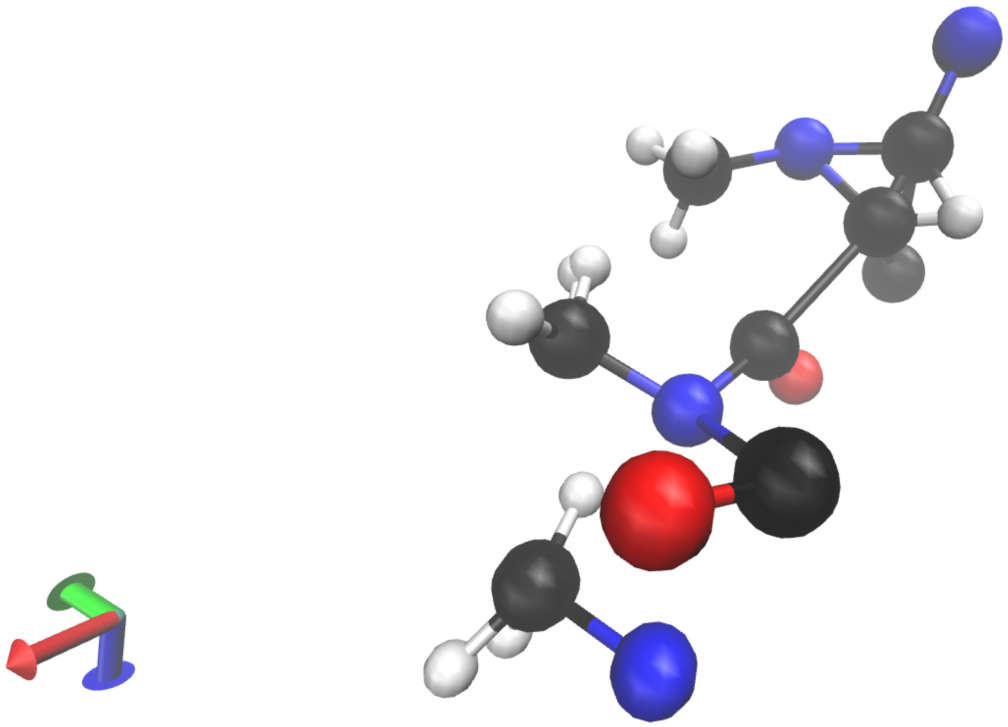

Figure 1: The output of the caffeine molecule optimization with only 5 mating events. Figure generated using VMD (Humphrey et al. (1996), Stone (1998)).

5 Trace to Requirements

Req.	**Modules**
R1	mol_input.py, ga_input.py
R2	fitg.py, pmem.py
R3	fitg.py, rmsd.py, energy.py
R4	fitg.py, energy.py, geometry.py, mol_input.py
R5	output.py, geometry.py, pmem.py
R6	Satisfied in kaplan/test/testfiles
R7	tournament.py, mutations.py, ring.py, gac.py
NFR1	fitg.py
NFR2	all modules, but especially gac.py
NFR3	all modules
NFR4	mol_input.py, ga_input.py
NFR5	energy.py, rmsd.py, mol_input.py, output.py

Table 1: Trace between requirements and modules, with the specific python file names given (as found in the kaplan directory). A similar table is found in the module guide document.

6 Trace to Modules

Module	Test File	Comments
M1 (Hardware Hiding)	-	Out of scope
M2 (GA Input)	test_ga_input.py	complete
M3 (Molecule Input)	test_mol_input.py	complete
M4 (GA Control)	test_gac.py	missing assertions for output generation
M5 (Fit_G)	tests-for-energy-summation.ipynb	more tests will be written in test_fitg.py
M6 (Tournament)	test_tournament.py	incomplete
M7 (Crossover & Mutation)	test_mutations.py	mostly complete
M8 (Ring)	test_ring.py	complete for current spec
M9 (Pmem)	-	test will be written in test_pmem.py
M10 (Output)	-	test will be written in test_output.py, tests by visual inspection have been done
M11 (Geometry)	test_geometry.py and pybel-xyz-zmatrix.ipynb	incomplete
M12 (Energy)	tests-for-energy-summation.ipynb and test_avail_psi4.ipynb	incomplete
M13 (RMSD)	rmsd-tests-for-hydrogen.ipynb and test_rmsd.py	mostly complete (visual inspection still needed)

Table 2: Trace between modules and tests, with the specific python file names given (as found in the kaplan/test directory), or (if no python test was written) a comment or reference to a jupyter notebook (located in the kaplan/test/jupyter-notebooks directory).

7 Code Coverage Metrics

See the System VnV Plan for code coverage, or see the Travis build at this link: https://travis-ci.org/PeaWagon/Kaplan/builds/465876614.

References

William Humphrey, Andrew Dalke, and Klaus Schulten. VMD – Visual Molecular Dynamics. *Journal of Molecular Graphics*, 14:33–38, 1996.

John Stone. *An Efficient Library for Parallel Ray Tracing and Animation.* Master's thesis, Computer Science Department, University of Missouri-Rolla, April 1998.

8 Appendix

Here is a list of the tests that have been run for Kaplan using nosetests.

```
kaplan.test.test_generate_children ... ok
kaplan.test.test_run_kaplan ... ok
kaplan.test.test_calc_rmsd ... ok
kaplan.test.test_generate_parser ... ok
kaplan.test.test_Ring ... ok
kaplan.test.test_run_tournament ... ok
kaplan.test.test_read_ga_input ... ok
kaplan.test.test_read_mol_input ... ok
kaplan.test.test_select_pmems ... ok
kaplan.test.test_get_zmatrix_template ... ok
kaplan.test.test_select_parents ... ok
kaplan.test.test_update_zmatrix ... ok
kaplan.test.test_verify_ga_input ... ok
kaplan.test.test_verify_mol_input ... ok
kaplan.test.test_zmatrix_to_xyz ... ok
kaplan.test.test_Ring_fill ... ok
kaplan.test.test_Ring_getitem ... ok
kaplan.test.test_ga_input.test_read_ga_input ... ok
kaplan.test.test_ga_input.test_verify_ga_input ... ok
```

```
kaplan.test.test_gac.test_run_kaplan ... ok
kaplan.test.test_geometry.test_generate_parser ... ok
kaplan.test.test_geometry.test_get_zmatrix_template ... ok
kaplan.test.test_geometry.test_update_zmatrix ... ok
kaplan.test.test_geometry.test_zmatrix_to_xyz ... ok
kaplan.test.test_mol_input.test_read_mol_input ... ok
kaplan.test.test_mol_input.test_verify_mol_input ... ok
kaplan.test.test_mutations.test_generate_children ... ok
kaplan.test.test_ring.test_Ring ... ok
kaplan.test.test_ring.test_Ring_fill ... ok
kaplan.test.test_ring.test_Ring_getitem ... ok
kaplan.test.test_ring.test_Ring_setitem ... ok
kaplan.test.test_ring.test_Ring_update ... ok
kaplan.test.test_rmsd.test_calc_rmsd ... ok
kaplan.test.test_tournament.test_run_tournament ... ok
kaplan.test.test_tournament.test_select_pmems ... ok
kaplan.test.test_tournament.test_select_parents ... ok
```