OpenCL Accelerated Molecular Docking with Historical Genetic Algorithm

Eka A. Kurniawan-NTU, Ouyang Xuchang-NTU, and Kwoh Chee Keong-NTU

***Abstract*— Molecular docking tools are developed to help scientists to discover new drug efficiently. The main contributions of this work are implementing AutoDock 4 semiempirical energy function in Python (which originally written in C), implementing new algorithm for conformational search called historical genetic algorithm, and implementing heterogeneous parallel program by integrating Python and OpenCL to achieve runtime improvement up to 477 times compared to sequential program.**

I. Introduction

This work focuses on two main portions of molecular docking workflow: *semiempirical energy calculation* and *conformational search*. The objective is to implement molecular docking tool using Python programming language that is well known to natural scientists so they can work on their own representation of semiempirical energy function. Also, for computer scientists so they can easily implement new optimization algorithm for the conformational search.

II. Implementation

The implementation (called hppNeuroDock) consists of three major classes; they are NeuroDock, Dock, and GeneticAlgorithm. NeuroDock class acts like the main program. It instantiates both Dock class for energy calculation and GeneticAlgorithm class for conformational search.

For energy calculation, this work implements AutoDock 4 semiempirical energy function that includes Lennard-Jones (6-12) potential for van der Waals interaction, 10-12 potential for hydrogen bond, Coulomb's law for electrostatic interaction of charged biochemical molecules, and desolvation model using Wesson and Eisenberg approach [1].

For conformational search, this work implements new historical genetic algorithm. It is based on human population history evolved from nomad population that lived from place to place into recent society called settler that occupies certain area permanently. Nomad population is modeled with higher mutation rate has the ability to explore entire search space efficiently but it has difficulty to get down into local minimum. Then, the population is remodeled into settler population with lower mutation rate in the hope of reaching local minimum. From a community that consists of multiple populations with different local minima, the one that has the lowest energy is considered to be the global minimum.

Having completed Python implementation, functions that require intensive computing power are translated into OpenCL. Figure 1 shows the runtime taken by sequential and parallel implementations run on both NVIDIA GeForce GT 650M GPU and Intel Core i7 CPU.

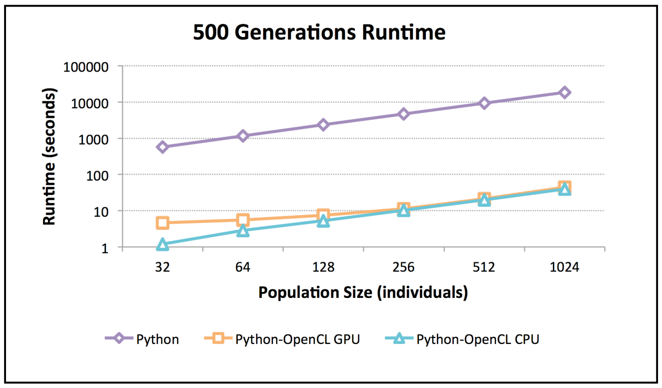

Figure 1. Combined sequential Python, parallel Python-OpenCL using GPU, and parallel Python-OpenCL using CPU runtimes in 500 generations based on different population size.

III. Result

Protein and ligand used to evaluate the implementation are HIV-1 and Indinavir respectively. Numerically, it achieves lowest free energy of -14.27 kcal/mol as compared to AutoDock 4 with -15.66 kcal/mol. The ligand location and conformation RMSD of the two are 1.53 Å apart. Visually, the result is shown in Figure 2.

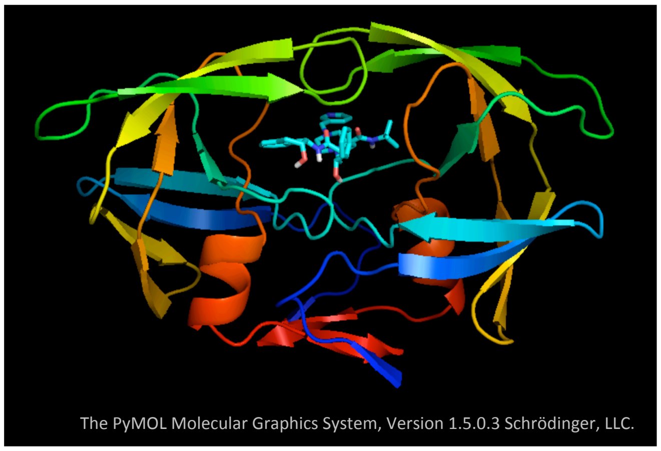
The PyMOL Molecular Graphics System, Version 1.5.0.3 Schrödinger, LLC.

Figure 2. Binding mode of Indinavir ligand to the rigid part of HIV-1 protease as the receptor.

References

[1] R. Huey, G. M. Morris, A. J. Olson, and D. S. Goodsell, "A Semiempirical Free Energy Force Field with Charge-Based Desolvation," *Computational Chemistry*, 28, pp. 1145-1152, 2007.