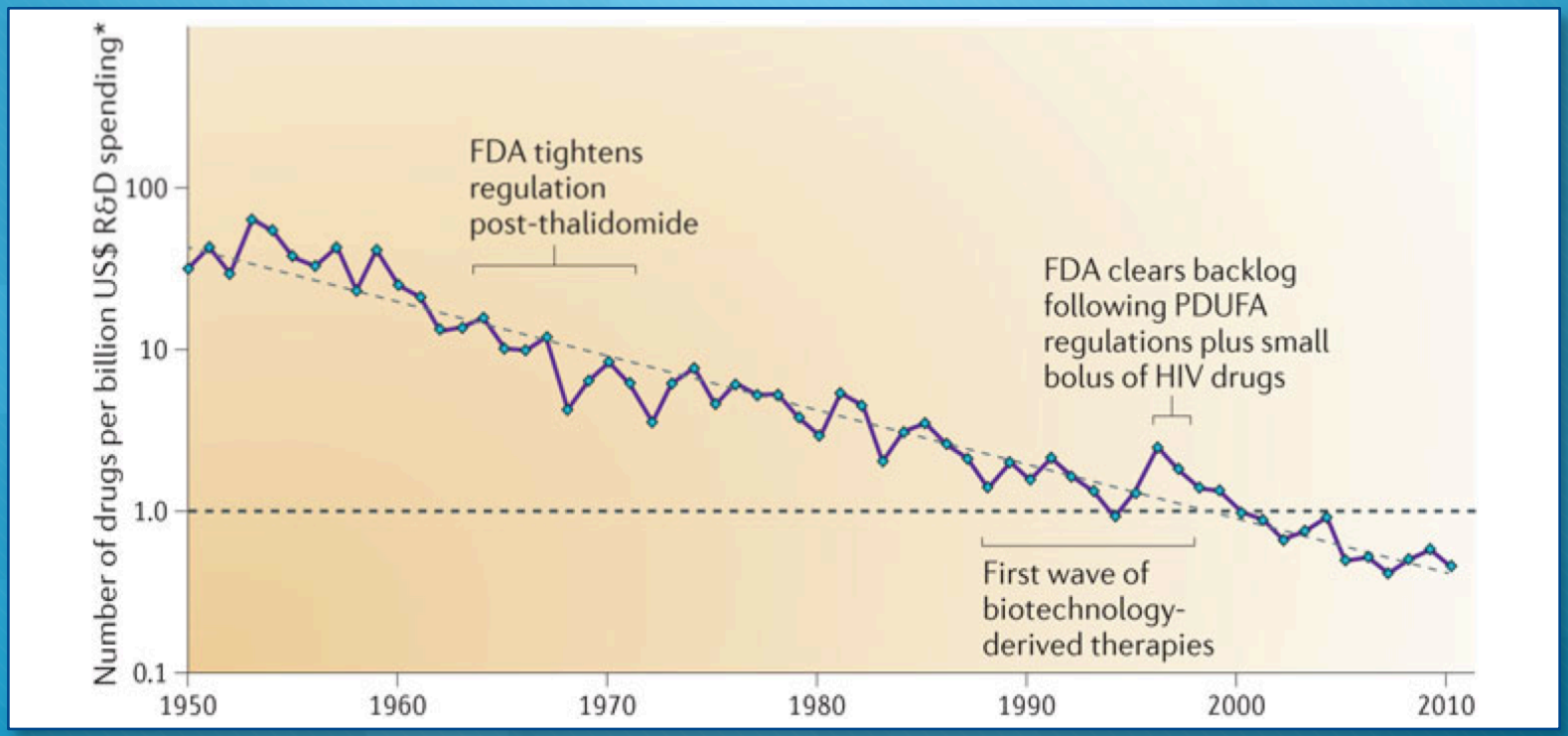
OpenCL Accelerated Molecular Docking with Historical Genetic Algorithm

Eka A. Kurniawan

Eroom's Law

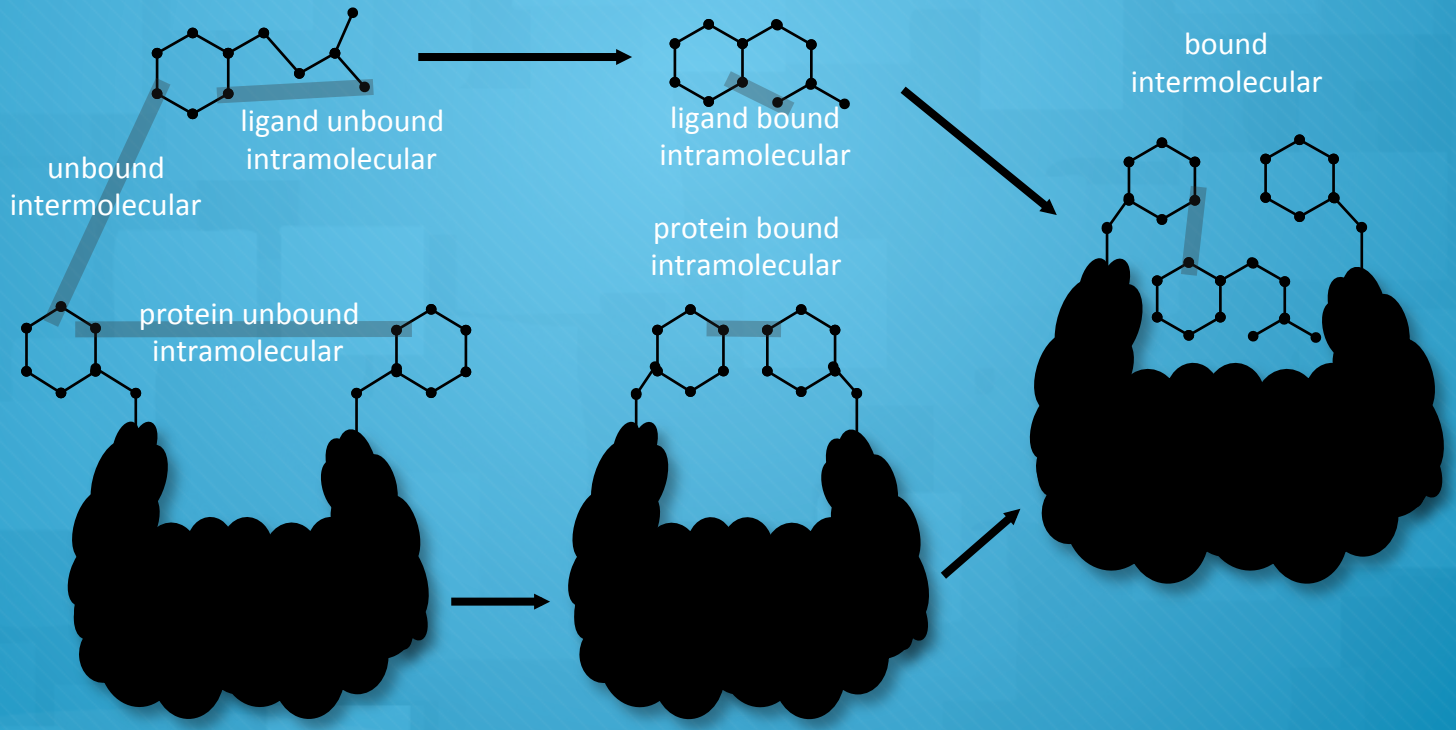


Scannell, J.W., Blanckley, A., Boldon, H. & Warrington, B., 2012. Diagnosing the decline in pharmaceutical R&D efficiency. *Nature Reviews Drug Discovery*, 11, pp.191-200.

Outline

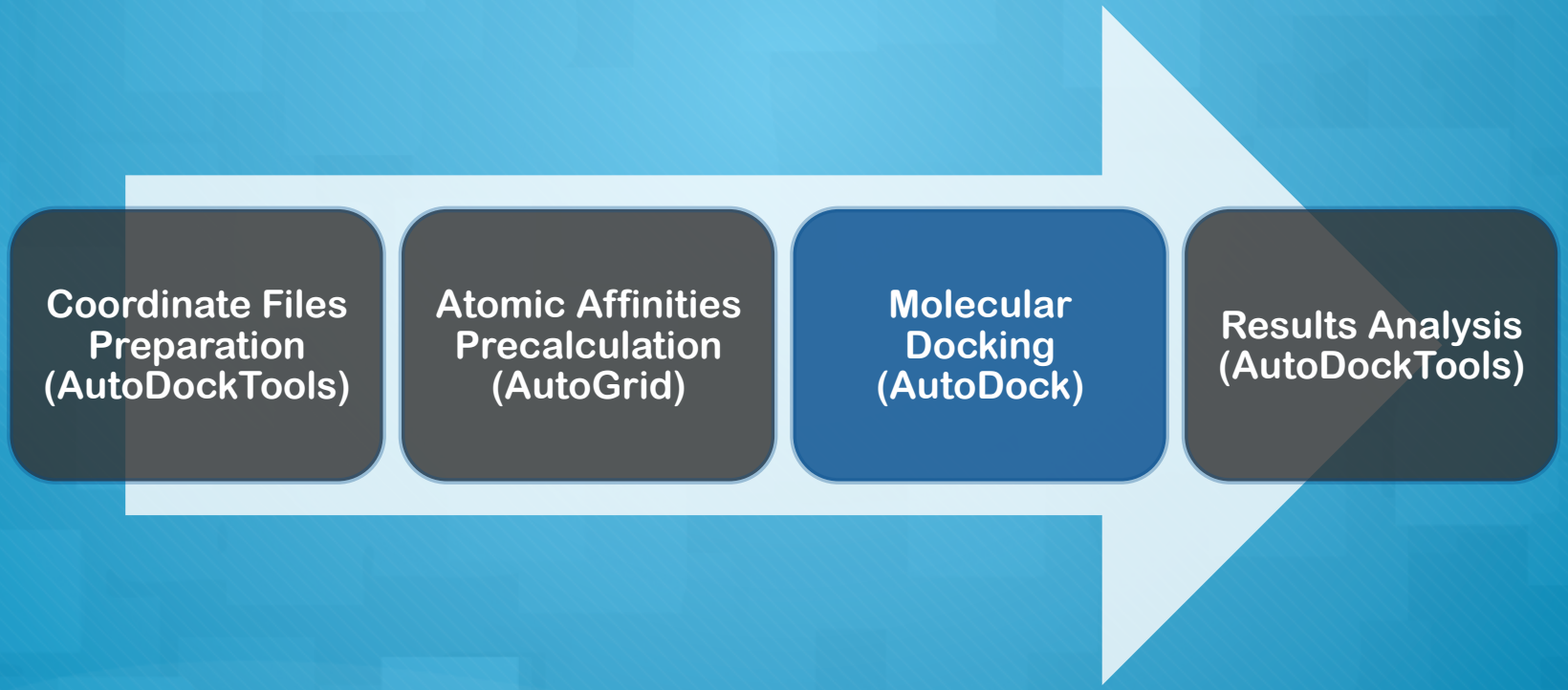
- Molecular Docking
- Sequential Python Implementation
- Parallel Python-OpenCL Implementation
- Docking Result
- Benchmark
- Conclusions and Future Works

Molecular Docking



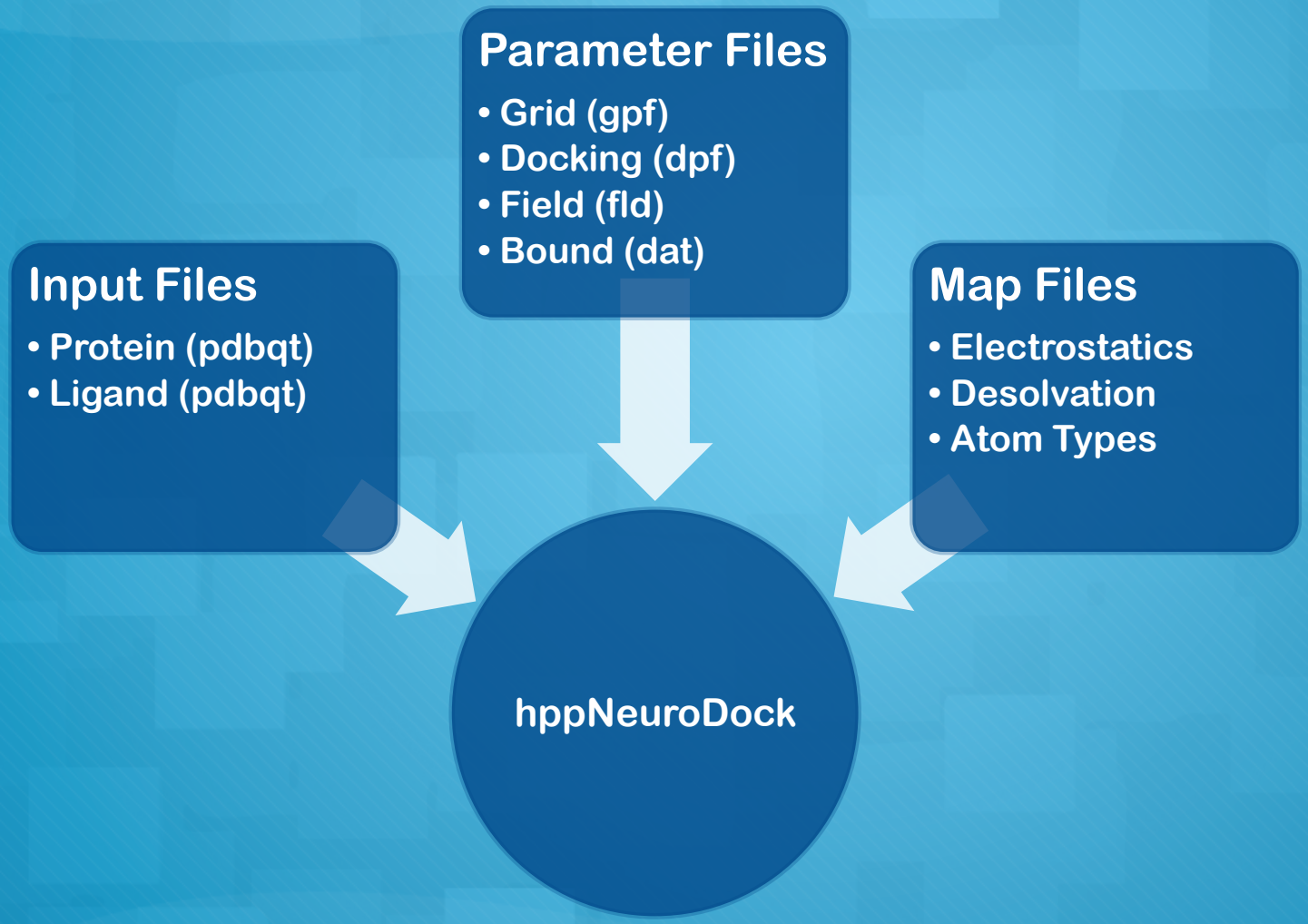
Huey, R., Morris, G.M., Olson, A.J. & Goodsell, D.S., 2007. A Semiempirical Free Energy Force Field with Charge-Based Desolvation. *Computational Chemistry*, 28, pp.1145-52.

AutoDock 4 Flow



Morris, G.M. et al., 2010. *User Guide AutoDock Version 4.2: Automated Docking of Flexible Ligands to Flexible Receptors*. [Online] Available at: http://autodock.scripps.edu/faqs-help/manual/autodock-4-2-user-guide/AutoDock4.2_UserGuide.pdf.

Inputs


Parameter Files
• Grid (gpf)
• Docking (dpf)
• Field (fld)
• Bound (dat)
Input Files
• Protein (pdbqt)
• Ligand (pdbqt)
Map Files
• Electrostatics
• Desolvation
• Atom Types
hppNeuroDock

Two Main Portions

- Energy Calculation: AutoDock 4 Semiempirical Energy Function
- Conformational Search: Historical Genetic Algorithm

AutoDock 4 Semiempirical Energy Function

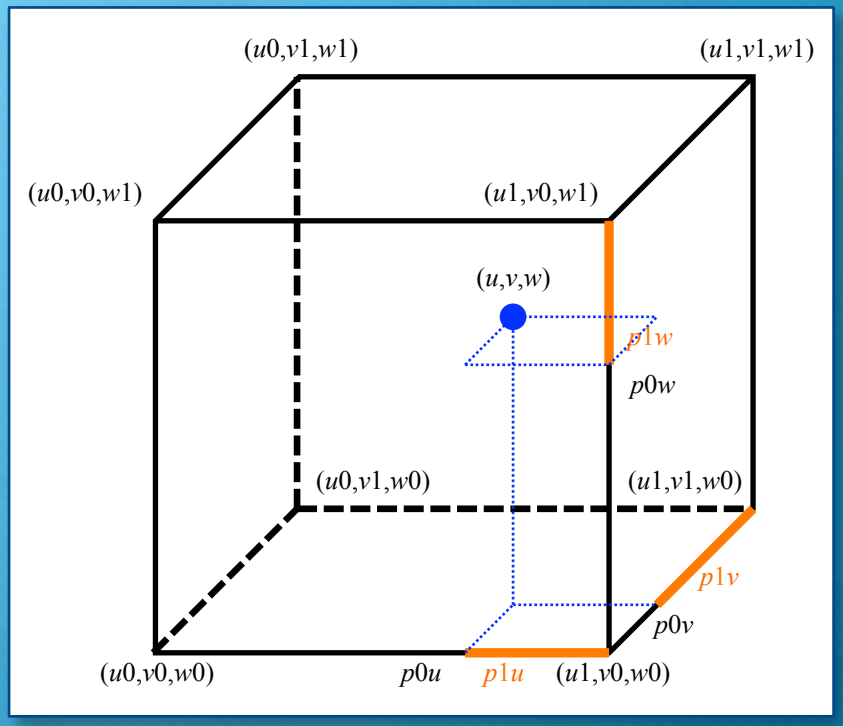
$$\Delta G = (V_{bound}^{L-L} - V_{unbound}^{L-L}) + (V_{bound}^{P-P} - V_{unbound}^{P-P}) + (V_{bound}^{P-L} - V_{unbound}^{P-L} + \Delta S_{conf})$$

$$\mathrm{V} = W_{vdw} \sum_{i,j} \left(\frac{A_{ij}}{r_{ij}^{12}} - \frac{B_{ij}}{r_{ij}^{6}} \right) + W_{hbond} \sum_{i,j} E(t) \left(\frac{C_{ij}}{r_{ij}^{12}} - \frac{D_{ij}}{r_{ij}^{10}} \right) + W_{elec} \sum_{i,j} \frac{q_i q_j}{\varepsilon(r_{ij}) r_{ij}}$$
$$+ W_{sol} \sum_{i,j} (S_i V_j + S_j V_i) e^{\left(-r_{ij}^2 / 2\sigma^2 \right)}$$

Huey, R., Morris, G.M., Olson, A.J. & Goodsell, D.S., 2007. A Semiempirical Free Energy Force Field with Charge-Based Desolvation. *Computational Chemistry*, 28, pp.1145-52.

Energy Function Implementation

- Intermolecular
- Intramolecular/Internal

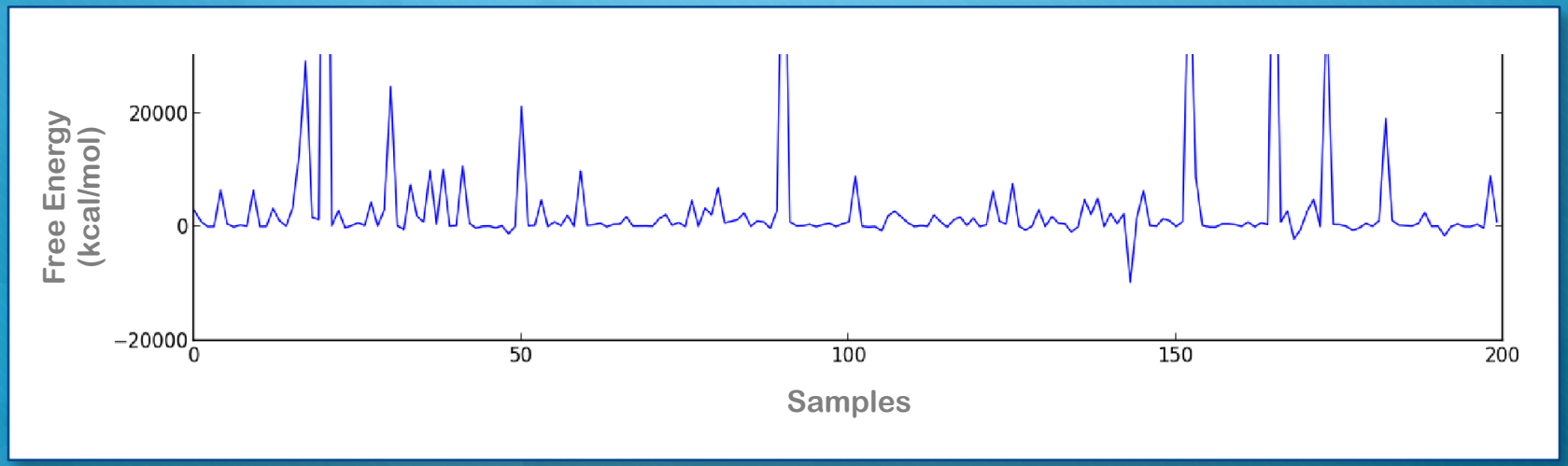


Historical Genetic Algorithm

- minimize E(s); $s = (x,y,z,a,b,c,d,t_1,t_2,\ldots,t_n)$
- Genetic Algorithm (Geometries → Coordinates)
- Free Energy Represents Individual Fitness
- Population Types
  - Nomad: High Mutation Rate
  - Settler: Low Mutation Rate

Nomad-Settler Free Energy

- 150 Individuals Over 50 Generations
- Nomad Minus Settler Free Energy



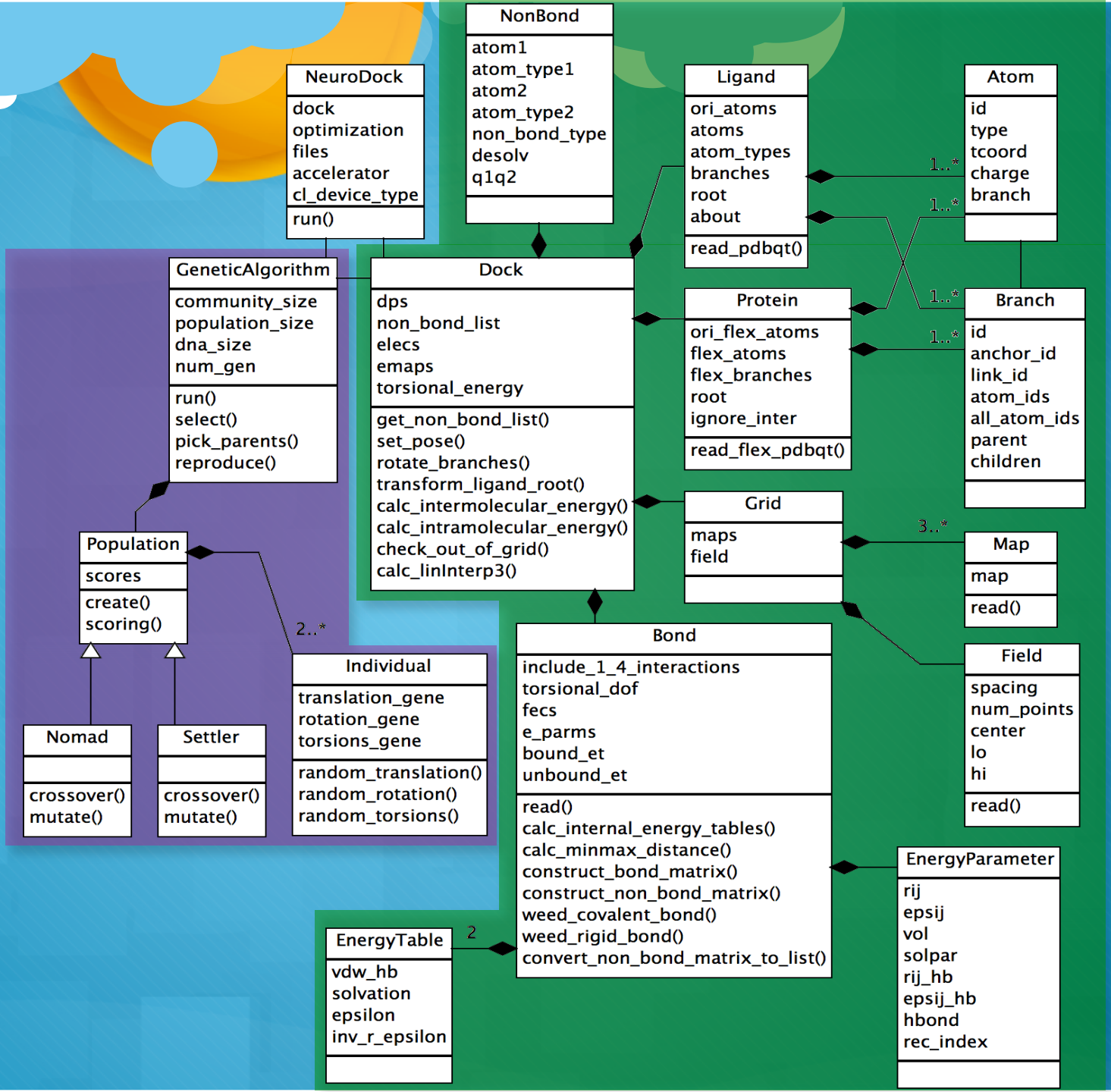
Python

- Free and Open Source Programming Language
- Focus on Productivity and System Integrity
- Notable Programming Languages Among Scientists
- Supports Multiple Scientific Packages
- Packages Allow the Core to be Written in C/C++
- Supports Object-Oriented Programming (OOP)

Sequential Python Implementation

- Pure Python (No Additional Module Required)
- Double-precision Floating-point
- NeuroDock, Dock and GeneticAlgorithm Classes

Class Diagram



OpenCL (Open Computing Language)

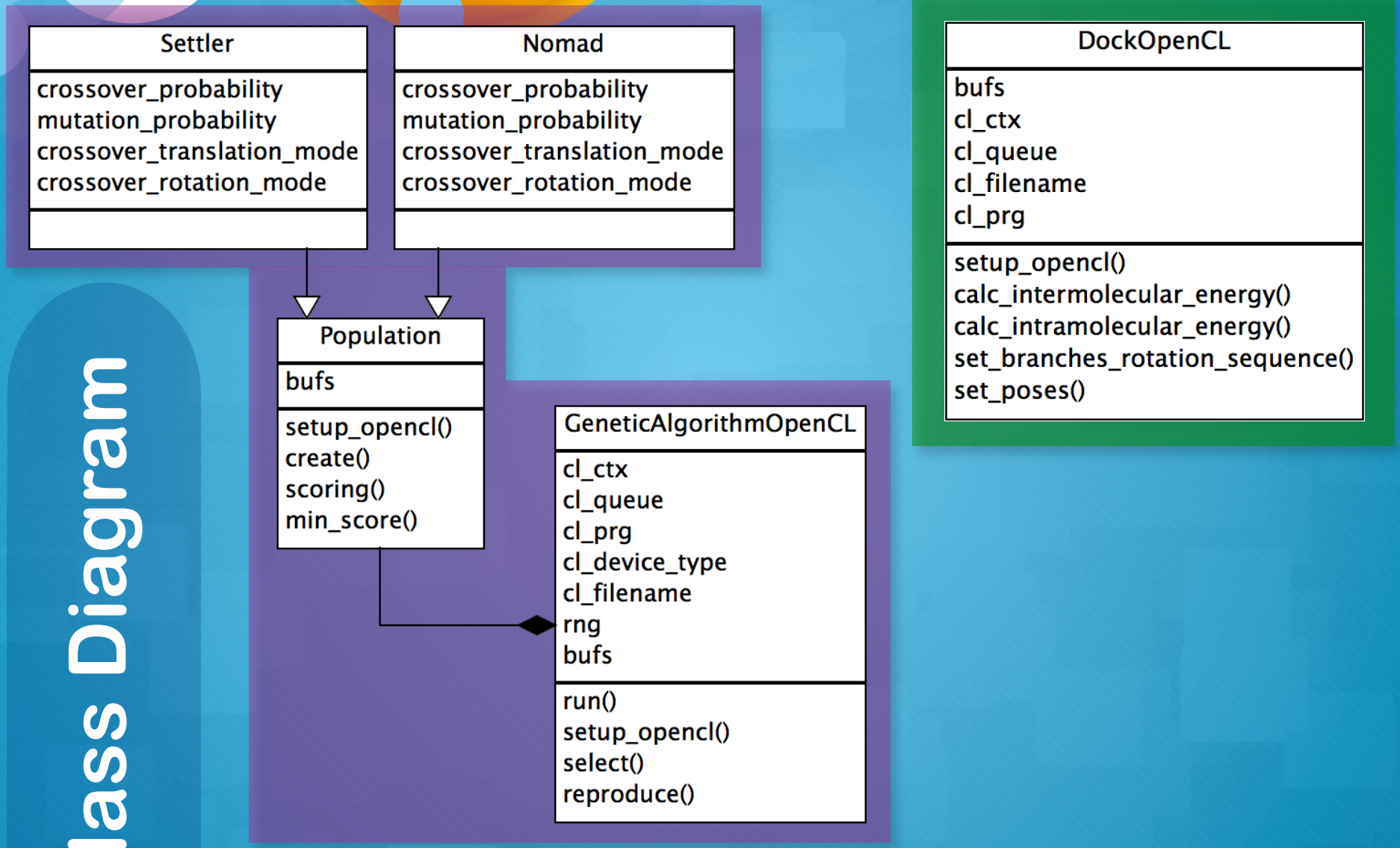
- Develop by Khronos Group (the Developer of OpenGL)
- Open and Royalty-Free Standard for Parallel Programming
- Runs on Multiple Devices (Intel, AMD, NVIDIA and Altera FPGA)
- Efficient Data Exchange with OpenGL
- PyOpenCL Module is an Open-Source Wrapper Between Python and OpenCL

Parallel Python-OpenCL Implementation

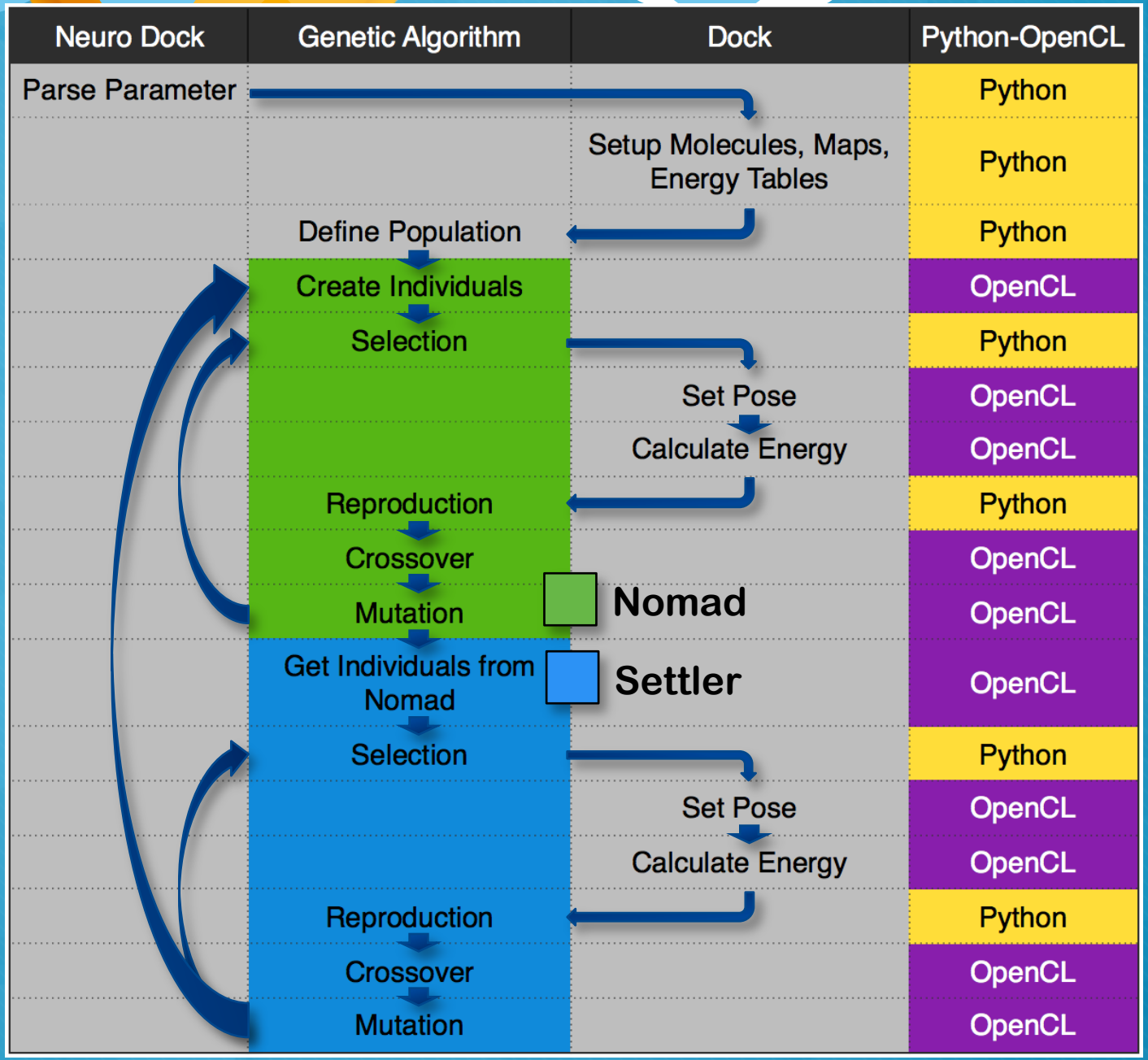
- Requires NumPy and PyOpenCL Modules
- Double-precision Floating-point
- NeuroDock, DockOpenCL and GeneticAlgorithmOpenCL Classes
- Fine-grained Parallelization and Offload Approach*

* Ouyang, X. & Kwoh, C.K., 2012. GPU Accelerated Molecular Docking with Parallel Genetic Algorithm. In *Parallel and Distributed Systems (ICPADS), 2012 IEEE 18th International Conference on*. Singapore, 2012. CPS.

Class Diagram

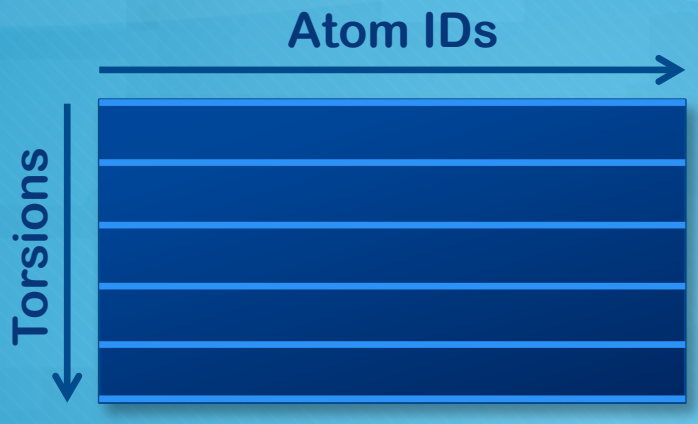


Execution Flow

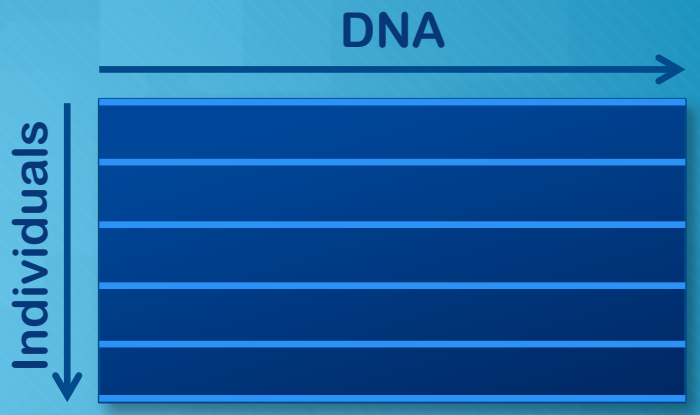


Parallel Data Structure

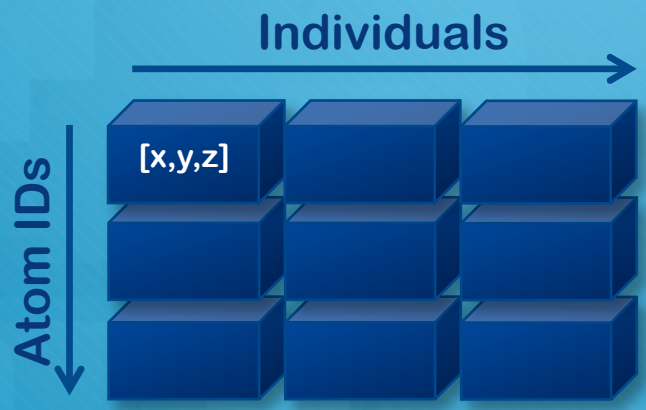
Branch Rotation Sequence



Population



Atom Coordinates

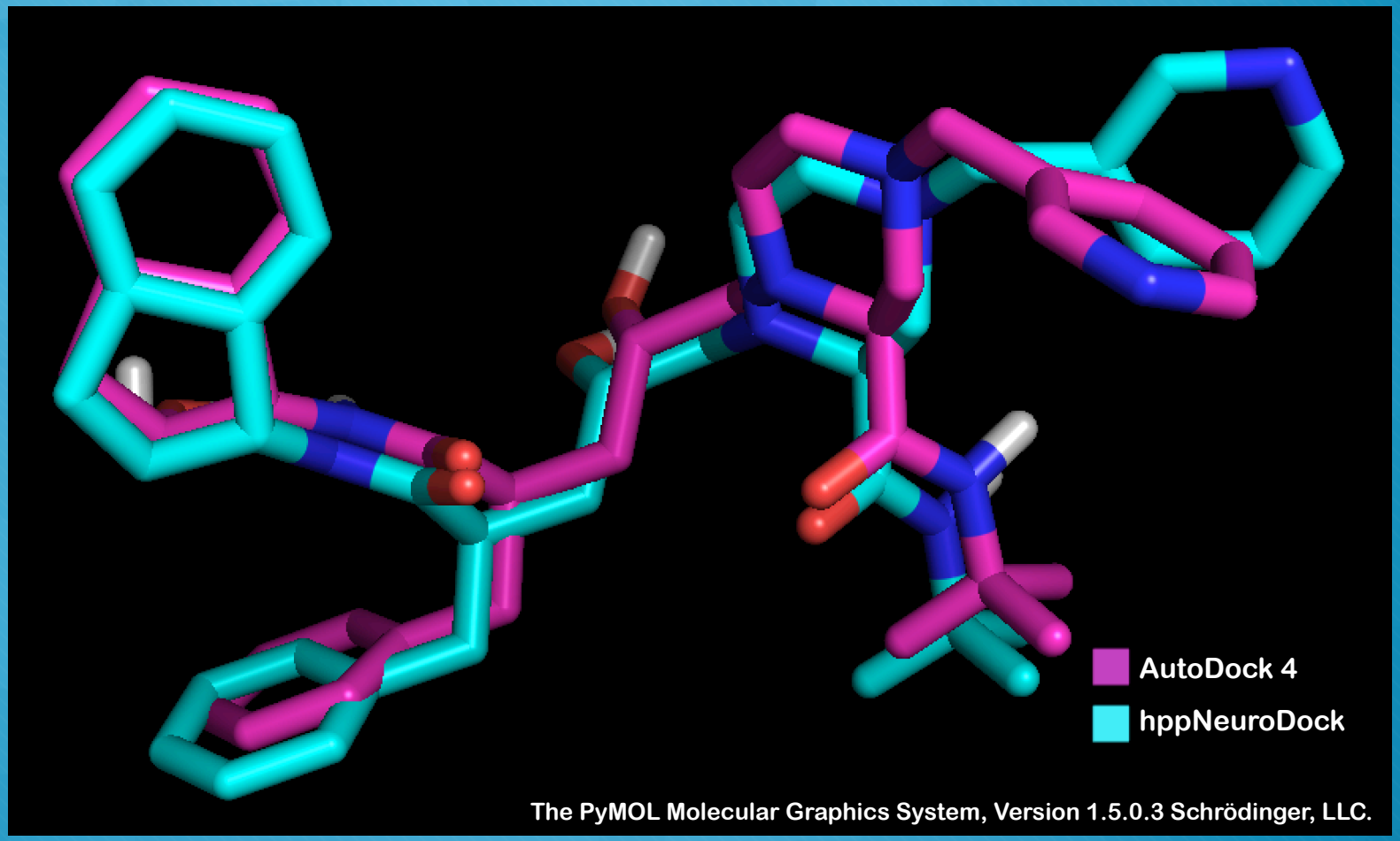


Maps

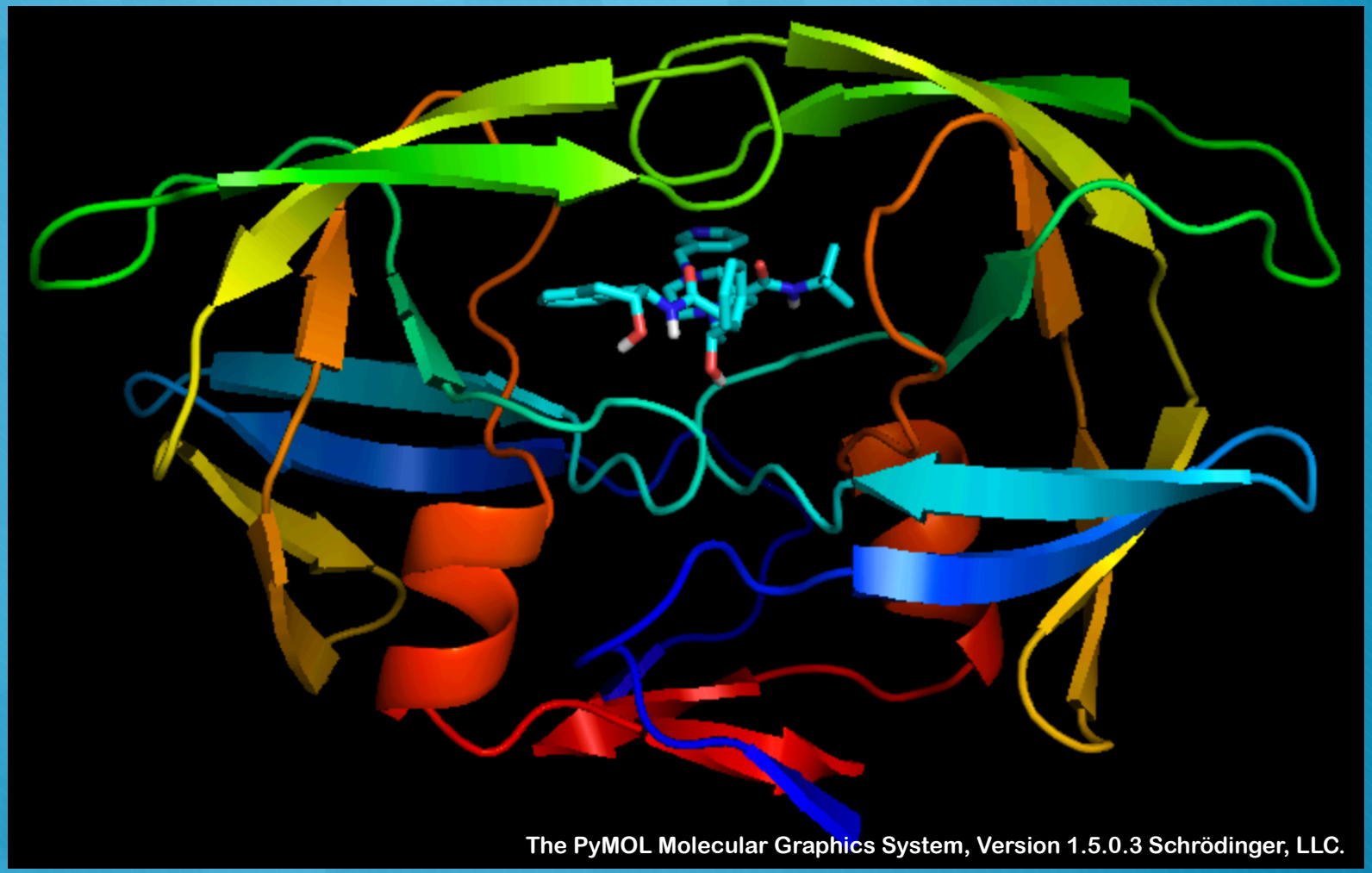
Docking Result

- HIV-1 Protease and Indinavir Interactions
- Free Energy AutoDock 4: -15.45 kcal/mol
- Free Energy hppNeuroDock: -14.27 kcal/mol
- RMSD: 1.53 Å

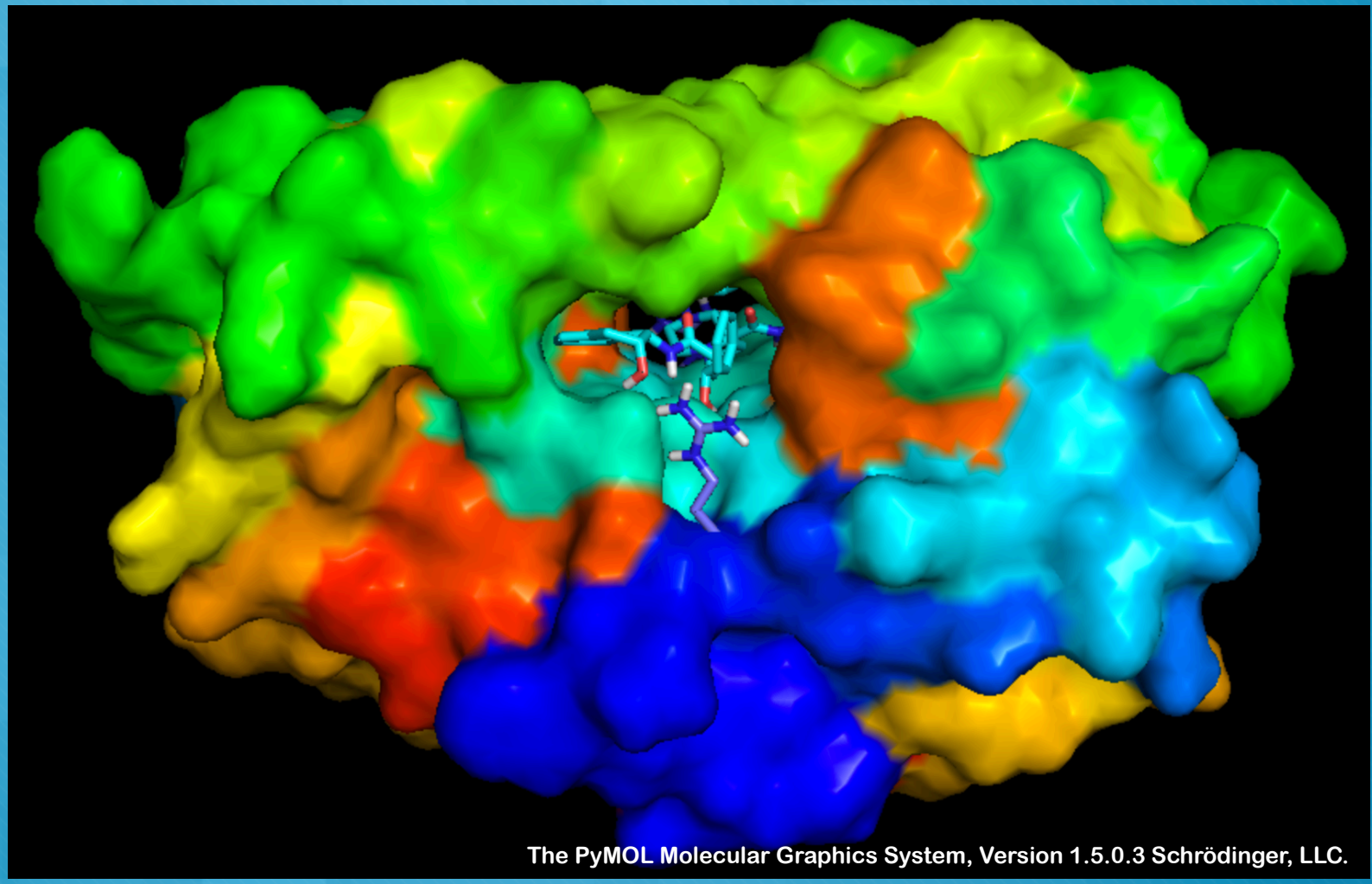
Ligand Comparison



Binding Mode



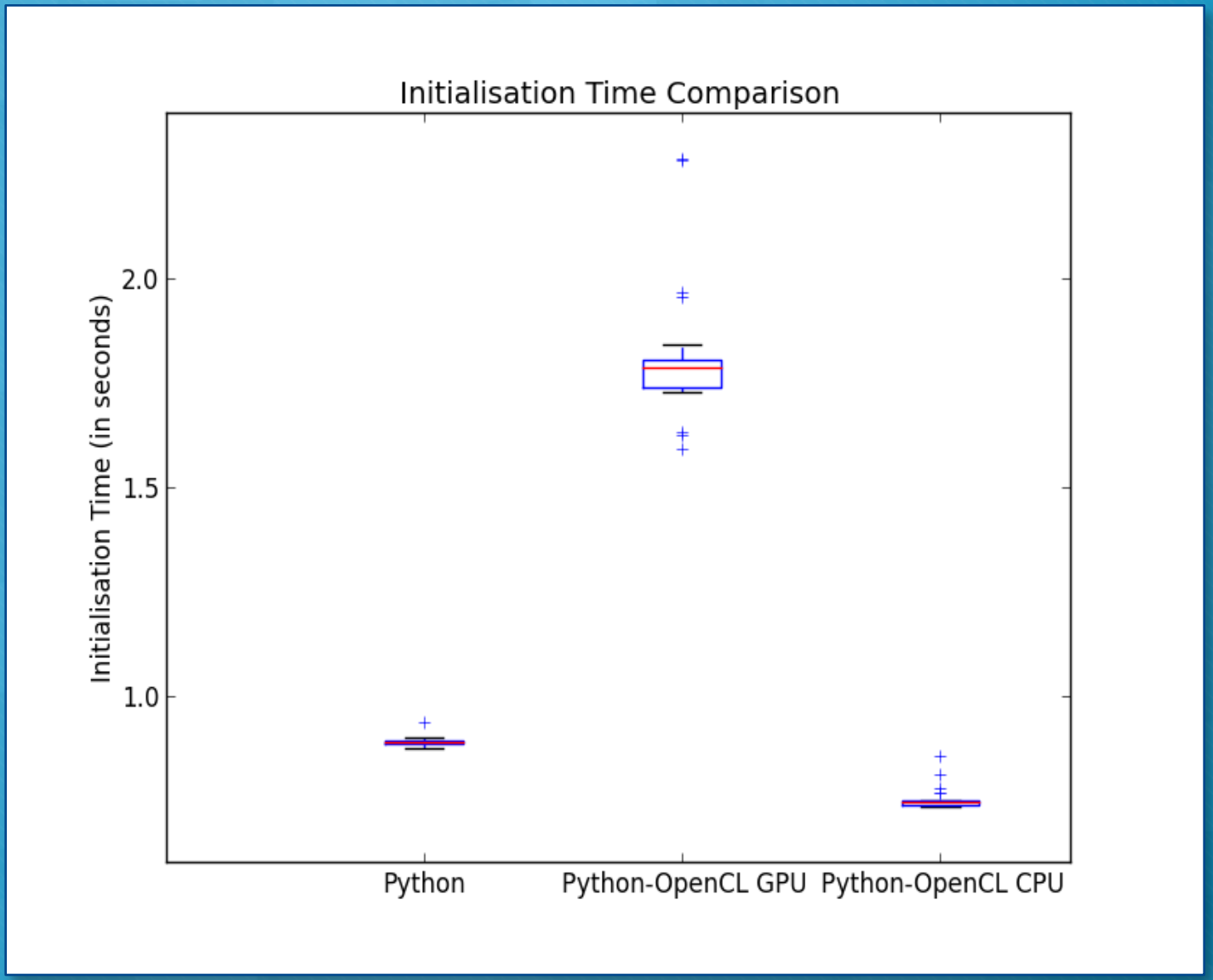
Binding Mode



Benchmark

- Hardware
  - Processor: 2.3GHz Intel Core i7 (4 Cores, 8 Threads)
  - Graphic: NVIDIA GeForce GT 650M 1GB (384 Cores)
  - Memory: 8GB 1600MHz DDR3
- Software
  - Python 2.7.3
  - NumPy 1.7.1
  - PyOpenCL 2013.1

Initialization Time



Runtime

Sequential Python

Number of Generations	Population Size					
	32	64	128	256	512	1024
10	5.16	7.43	14.48	32.74	58.12	-
50	48.9	95.78	176.14	391.87	729.37	-
100	105.53	211.02	403.57	861.12	1659.55	-
500	566.68	1156.61	2338.5	4667	9363	18593
1000	-	-	-	9217	18521	37876

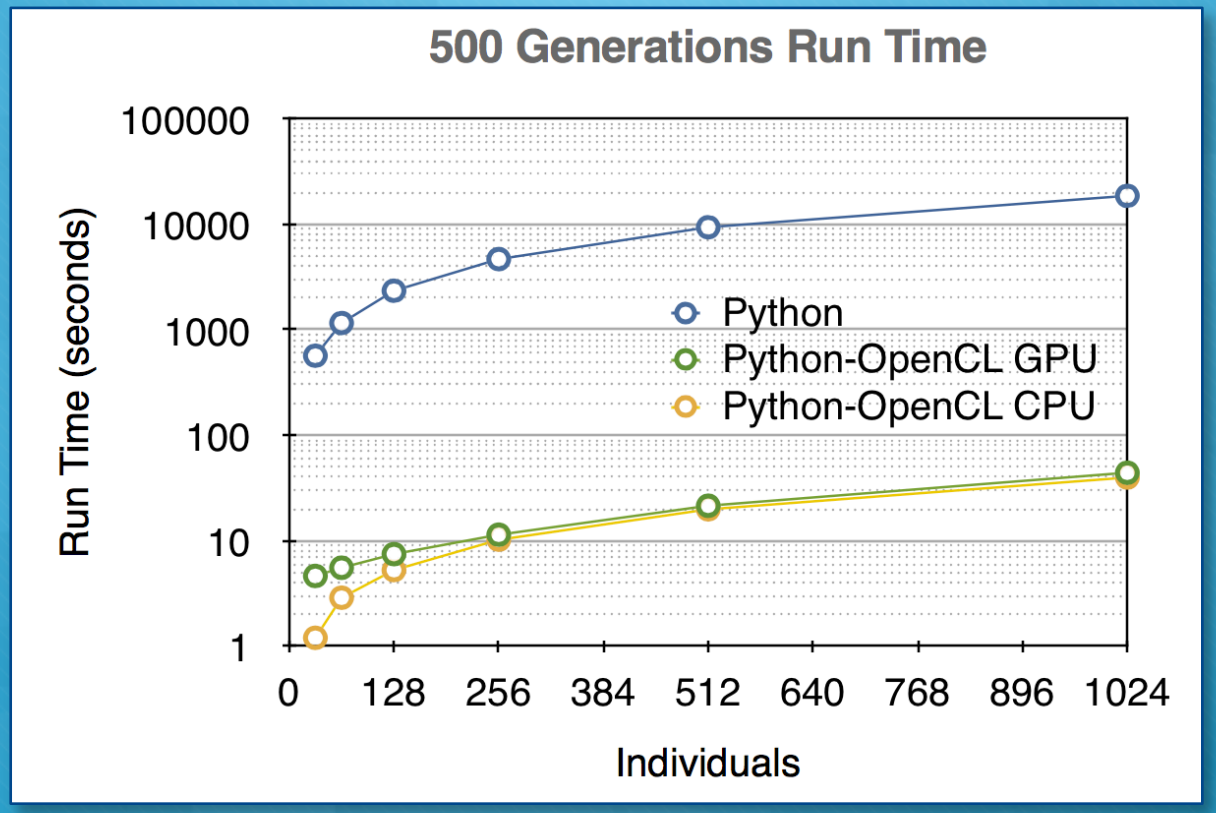
Parallel Python-OpenCL on GPU

Number of Generations	Population Size					
	32	64	128	256	512	1024
10	0.11	0.14	0.18	0.26	0.51	-
50	0.44	0.52	0.72	1.13	2.25	-
100	0.82	1.01	1.39	2.21	4.38	-
500	4.63	5.54	7.46	11.39	21.44	44.12
1000	-	-	-	21.82	42.84	88.53

Parallel Python-OpenCL on CPU

Number of Generations	Population Size					
	32	64	128	256	512	1024
10	0.03	0.06	0.11	0.22	0.44	-
50	0.12	0.29	0.54	1.03	2.04	-
100	0.23	0.58	1.07	2.06	4.03	-
500	1.2	2.88	5.26	10.2	19.82	39.7
1000	-	-	-	20.11	39.78	79.27

Runtime for 500 Generations



Speedup

Sequential Python over Parallel Python-OpenCL on GPU

Number of Generations	Population Size					
	32	64	128	256	512	1024
10	46.91	53.07	80.44	125.92	113.96	-
50	111.14	184.19	244.64	346.79	324.16	-
100	128.70	208.93	290.34	389.65	378.89	-
500	122.39	208.77	313.47	409.75	436.71	421.42
1000	-	-	-	422.41	432.33	427.83

Sequential Python over Parallel Python-OpenCL on CPU

Number of Generations	Population Size					
	32	64	128	256	512	1024
10	172.00	123.83	131.64	148.82	132.09	-
50	407.50	330.28	326.19	380.46	357.53	-
100	458.83	363.83	377.17	418.02	411.80	-
500	472.23	401.60	444.58	457.55	472.40	468.34
1000	-	-	-	458.33	465.59	477.81

Parallel Python-OpenCL on GPU over CPU

Number of Generations	Population Size					
	32	64	128	256	512	1024
10	3.67	2.33	1.64	1.18	1.16	-
50	3.67	1.79	1.33	1.10	1.10	-
100	3.57	1.74	1.30	1.07	1.09	-
500	3.86	1.92	1.42	1.12	1.08	1.11
1000	-	-	-	1.09	1.08	1.12

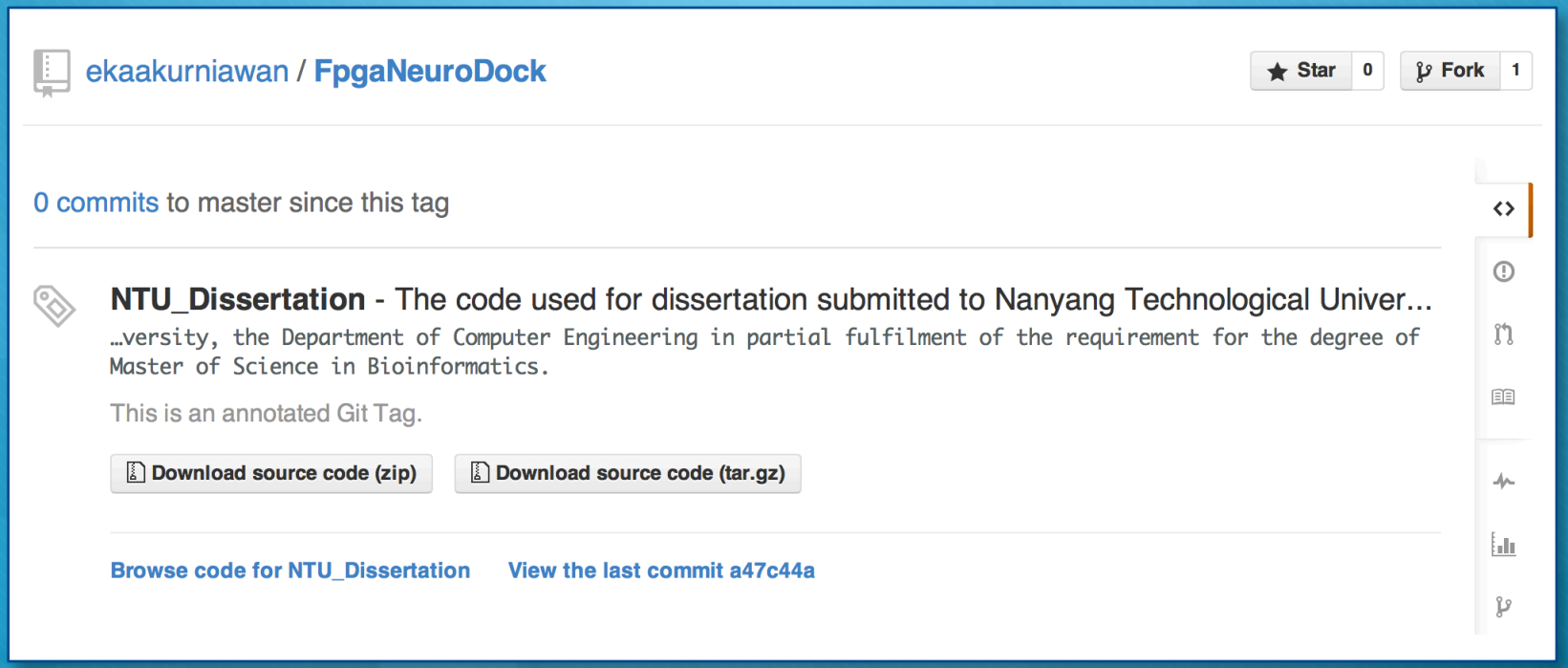
Conclusions

- Implemented AutoDock 4 semiempirical energy function in Python (which originally written in C)
- Implemented new algorithm for conformational search called historical genetic algorithm
- Implemented heterogeneous parallel program by integrating Python and OpenCL
- hppNeuroDock produces ligand conformation with RMSD 1.53 Å deviated from AutoDock 4
- hppNeuroDock produces negative docking energy of -14.27 kcal/mol compared to -15.66 kcal/mol produced by AutoDock 4
- Benchmark result shows that parallel Python improves the runtime up to 477 times faster than sequential Python

Future Works

- Implement Lamarckian Genetic Algorithm
- Using FPGA as the Accelerator
- Implement Python-MPI-OpenCL

Source Code



https://github.com/ekaakurniawan/FpgaNeuroDock/releases/tag/NTU_Dissertation

Special Thanks

- Prof. Kwoh Chee Keong
- Ouyang Xuchang