Accelerating Drugs Discovery with Deep Reinforcement Learning – Review

IBN-ML-Study

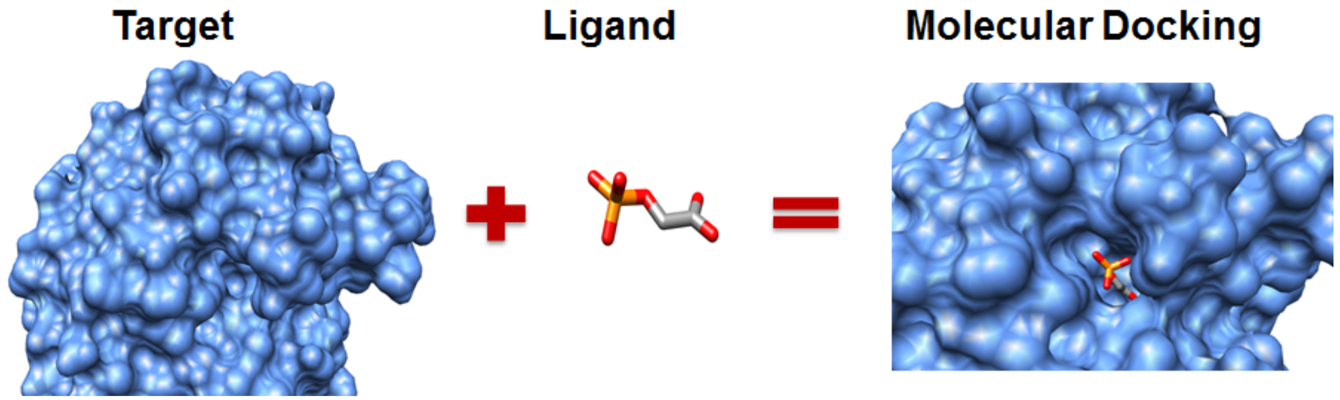
Week 13 2019/04/06

Eugene Bang

Accelerating Drugs Discovery with Deep Reinforcement Learning

- Traditional drug discovery process usually take around one decade
  - Recently shortened by the use of Virtual Screening
  - Most effective VS = Docking
    - PLDP(Protein-Ligand Docking Prediction) problem
- Field of AI has grown exponentially
  - Deep Learning – Neural Networks
  - Reinforcement Learning
    - Deep Q-network & AlphaGo
    - Deep Reinforcement Learning

- Downsides of Virtual Screening
  - Ligands can be evaluated in millions of different positions to find the best coupling location
    - This Docking process is very demanding



- We propose *DQN-Docking*
  - Covering the disadvantages of VS with DRL
  - Accelerate the Docking process

- Virtual Screening method
  - Computational techniques that analyze large libraries(millions) of ligands
    - Bigger size&diversity = higher chances of drug discovery
  - Goal : finding ligands able to bind to target(protein receptor or enzyme) involved in a given disease
- HOWEVER,
  - Fastest VS methods cannot process large biological database in reasonable time
  - Constrained by computational resource and the theory level used in scoring

- The use of high performance computing for VS is necessary
  - Multithreading programming
    - In computer architecture, multithreading is the ability of a central processing unit (CPU) (or a single core in a multi-core processor) to execute multiple processes or threads concurrently, supported by the operating system.
  - Heterogeneous computing system
    - Heterogeneous computing refers to systems that use more than one kind of processor or cores. - CPU + GPU
    - Challenges : Limits the system growth; scalability, programmability or data management

- METADOCK
  - Heterogeneous computing
  - Metaheuristic scheme applied to the PLDP problem
    - a metaheuristic is a higher-level procedure or heuristic designed to find, generate, or select a heuristic (partial search algorithm) that may provide a sufficiently good solution to an optimization problem, especially with incomplete or imperfect information or limited computation capacity
  - Evaluates the ligand in millions of positions by varying translational and rotational degrees of freedom around the surface of the receptor for the selected heuristic

- Reinforcement Learning
  - At a given timestep *t*, the agent observes a state *st*, takes an action *at*, gets a reward *rt* from the environment, and transitions to a new state *st+1*
  - **Q-learning approach** to select the best action in a given time-step that maximizes the reward
    - Updates the Q-values using the Bellman equation

$$Q(s,a) = Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a))$$

    - DQN follows an e-greedy strategy to manage the exploration/exploitation trade-off
      - First : randon actions / Later : takes action based on Q-values

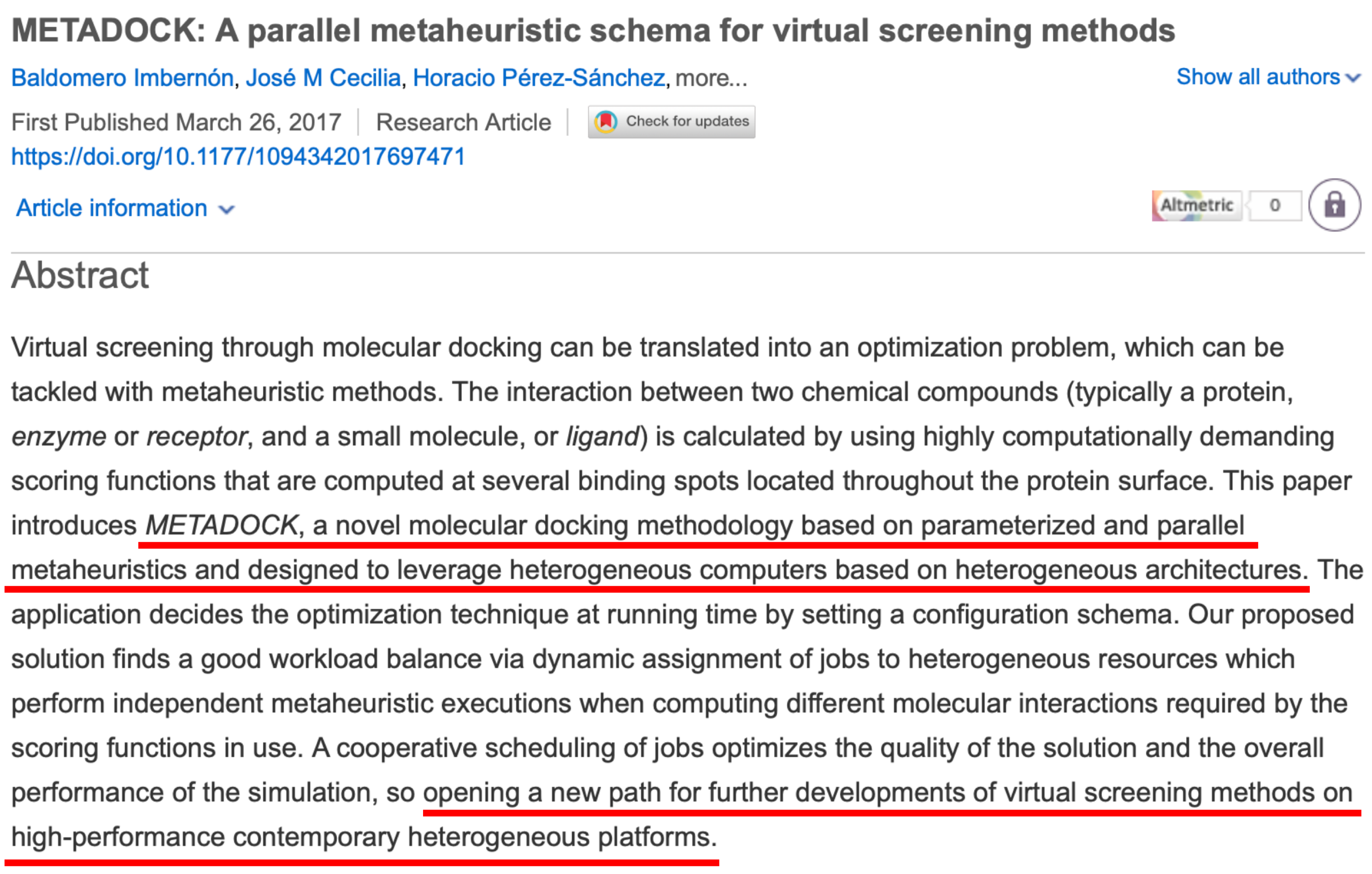
- Reinforcement Learning
  - During training : minimize the loss function

$$L(s,a|\theta_i) = (r + \gamma \max_{a'} \hat{Q}(s',a'|\theta_i^-) - Q(s,a|\theta_i))^2$$

- The key contributions of ‘Deep Mind’
  - 1) use of an experience replay dataset
  - 2) target Q-network to compute the loss
  - 3) reward clipping

- DOCKING = NP-complete problem
  - Although a solution to an NP-complete problem **can be verified "quickly,"** there is **no known way to find a solution quickly**. That is, the time required to solve the problem using any currently known algorithm increases rapidly as the size of the problem grows. As a consequence, determining whether it is possible to solve these problems quickly, called the P versus NP problem, is one of the fundamental unsolved problems in computer science today.

- METADOCK : efficient heuristic-based software
  - apply translations and rotations to the ligand in the Euclidean space
  - report the quality of the movement by using a scoring function – *calculation of three major terms*
    - 1) Electrostatic interactions
    - 2) Potential of Lennard-Jones(Van der Waals' foeces)
    - 3) The hydrogen-bonding



METADOCK: A parallel metaheuristic schema for virtual screening methods

Baldomero Imbernón, José M Cecilia, Horacio Pérez-Sánchez, more...   Show all authors

First Published March 26, 2017 | Research Article | Check for updates
https://doi.org/10.1177/1094342017697471
Article information
Altmetric 0

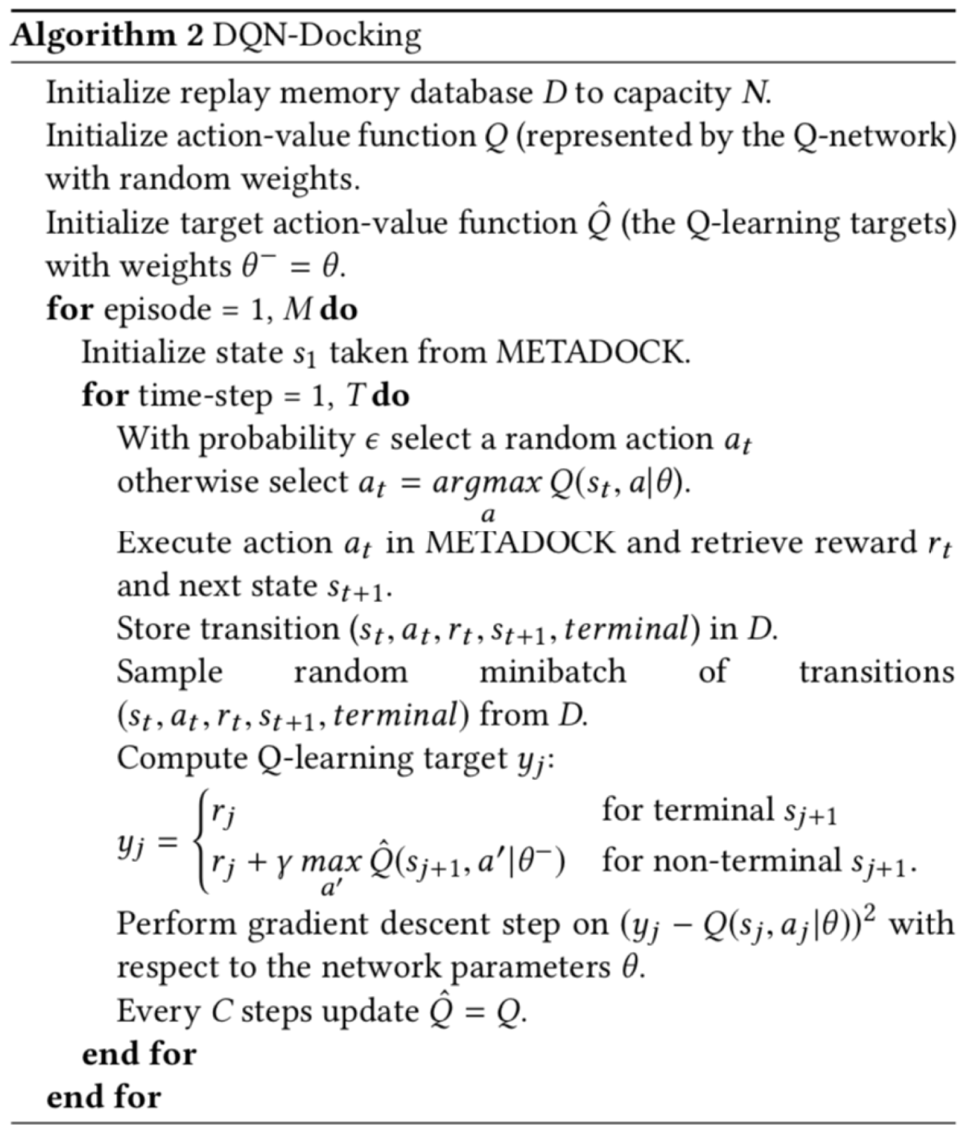
Abstract

Virtual screening through molecular docking can be translated into an optimization problem, which can be tackled with metaheuristic methods. The interaction between two chemical compounds (typically a protein, *enzyme* or *receptor*, and a small molecule, or *ligand*) is calculated by using highly computationally demanding scoring functions that are computed at several binding spots located throughout the protein surface. This paper introduces *METADOCK*, a novel molecular docking methodology based on parameterized and parallel metaheuristics and designed to leverage heterogeneous computers based on heterogeneous architectures. The application decides the optimization technique at running time by setting a configuration schema. Our proposed solution finds a good workload balance via dynamic assignment of jobs to heterogeneous resources which perform independent metaheuristic executions when computing different molecular interactions required by the scoring functions in use. A cooperative scheduling of jobs optimizes the quality of the solution and the overall performance of the simulation, so opening a new path for further developments of virtual screening methods on high-performance contemporary heterogeneous platforms.

- DRL : an efficient computation of Docking method with METADOCK
  - Ligand = the agent in DRL
  - Policy = NN estimating the Q-values
  - Environment represented by METADOCK
  - Reward = change in the score computed by METADOCK

- Original DQN vs DQN-Docking
  - No need of using image as input
    - Internal state of METADOCK is given as raw input to NN
    - Partially Observed Markov Decision Process(PO-MDP) problem
  - The game score is always positive vs METADOCK score is negative most of the time and can drop sharply
  - METADOCK has no stop conditions – manually included
    - Restricted movement area by the Euclidean distance of ligand-target
    - If ligand goes to deep inside the target, gives big negative score and terminates if the scores from 20 time-steps are smaller than threshold

- Pseudo-code of DQN-Docking

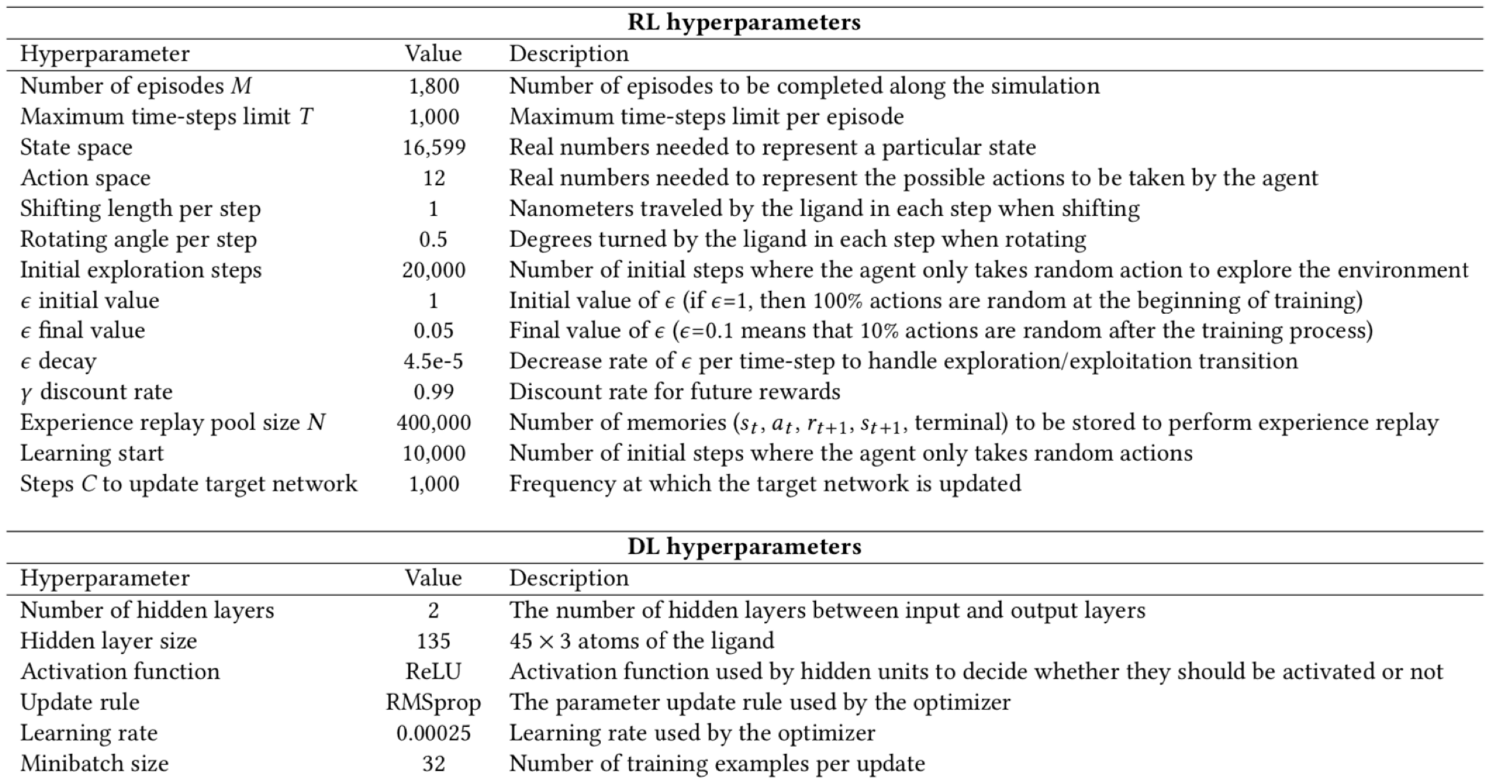


Algorithm 2 DQN-Docking

```
Initialize replay memory database $D$ to capacity $N$.
Initialize action-value function $Q$ (represented by the Q-network) with random weights.
Initialize target action-value function $\hat{Q}$ (the Q-learning targets) with weights $\theta^- = \theta$.
**for** episode = 1, $M$ **do**
    Initialize state $s_1$ taken from METADOCK.
    **for** time-step = 1, $T$ **do**
        With probability $\epsilon$ select a random action $a_t$
        otherwise select $a_t = \underset{a}{argmax}\, Q(s_t, a|\theta)$.
        Execute action $a_t$ in METADOCK and retrieve reward $r_t$ and next state $s_{t+1}$.
        Store transition $(s_t, a_t, r_t, s_{t+1}, terminal)$ in $D$.
        Sample random minibatch of transitions $(s_t, a_t, r_t, s_{t+1}, terminal)$ from $D$.
        Compute Q-learning target $y_j$:
        $y_j = \begin{cases} r_j & \text{for terminal } s_{j+1} \\ r_j + \gamma \underset{a'}{max}\, \hat{Q}(s_{j+1}, a'|\theta^-) & \text{for non-terminal } s_{j+1}. \end{cases}$
        Perform gradient descent step on $(y_j - Q(s_j, a_j|\theta))^2$ with respect to the network parameters $\theta$.
        Every $C$ steps update $\hat{Q} = Q$.
    **end for**
**end for**
```

- Technical implementations
  - Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40GHz
  - 128 GB of RAM, 1 TB SSD Hard Disk
  - NVIDIA GeForce GTX 780 GPU (Kepler).
  - TensorFlow 1.7.0 and Keras 2.1.5

Table 1: Values of the hyperparameters used in DQN-Docking



RL hyperparameters		
Hyperparameter	Value	Description
Number of episodes M	1,800	Number of episodes to be completed along the simulation
Maximum time-steps limit T	1,000	Maximum time-steps limit per episode
State space	16,599	Real numbers needed to represent a particular state
Action space	12	Real numbers needed to represent the possible actions to be taken by the agent
Shifting length per step	1	Nanometers traveled by the ligand in each step when shifting
Rotating angle per step	0.5	Degrees turned by the ligand in each step when rotating
Initial exploration steps	20,000	Number of initial steps where the agent only takes random action to explore the environment
ϵ initial value	1	Initial value of ϵ (if ϵ=1, then 100% actions are random at the beginning of training)
ϵ final value	0.05	Final value of ϵ (ϵ=0.1 means that 10% actions are random after the training process)
ϵ decay	4.5e-5	Decrease rate of ϵ per time-step to handle exploration/exploitation transition
γ discount rate	0.99	Discount rate for future rewards
Experience replay pool size N	400,000	Number of memories (s_t, a_t, r_{t+1}, s_{t+1}, terminal) to be stored to perform experience replay
Learning start	10,000	Number of initial steps where the agent only takes random actions
Steps C to update target network	1,000	Frequency at which the target network is updated

DL hyperparameters		
Hyperparameter	Value	Description
Number of hidden layers	2	The number of hidden layers between input and output layers
Hidden layer size	135	45×3 atoms of the ligand
Activation function	ReLU	Activation function used by hidden units to decide whether they should be activated or not
Update rule	RMSprop	The parameter update rule used by the optimizer
Learning rate	0.00025	Learning rate used by the optimizer
Minibatch size	32	Number of training examples per update

- DQN-Docking with 2BSM for 1,800 episodes
  - Set a fixed number of time-steps and tracked the average maximum predicted Q value
  - (theoretically) Q-value across episode should gradually increase
  - BUT Q-values per episode **start to gradually decline**.
    - "We cannot assure convergence"

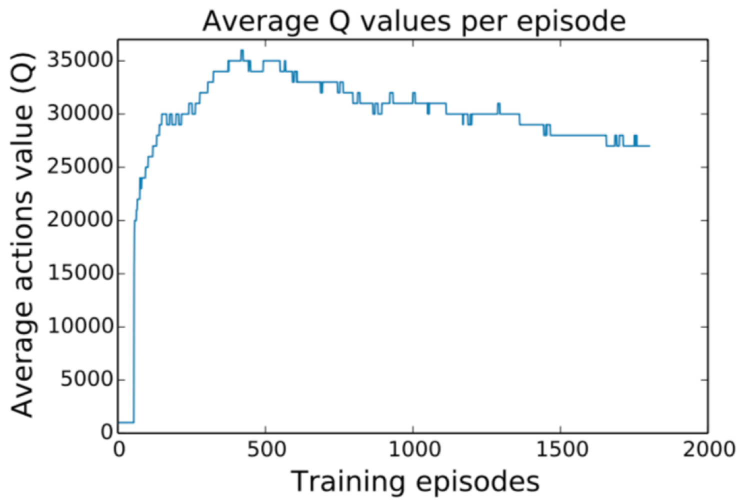


Figure 4: Training curve tracking the average predicted action-value.

- Declining of Q-values per episode
  - Communication between DQN-Docking/environment is slow
  - Do not have enough episodes to know if the algorithm will converge to the optimal solution
  - On the positive side…
    - The main components of RL in PLDP and METADOCK in a DQN system is identified to be able to move the ligand based on Q-value prediction
    - Need further exploration to fully guarantee convergence of DQN-Docking and achieve the goals for VS

- Current implementation of DQN-Docking has several limitations
  - 1) Communication between the algorithm and METADOCK entails to write two separate files in disc and DQN-Docking reads those files to store memories in the experience replay dataset
  - 2) DQN-Docking was tested only with the receptor-ligand pair known as 2BSM(which is very small protein)
    - Using internal states from METADOCK – if input size grows exponentially according to the number of atoms
    - Can be extended by using images and CNN instead of MLP
  - 3) Worked with a fixed ligand(no rotation in flexible bonds)
    - In 2BSM, ligand can fold in 6 bonds
  - 4) Applied the standard DQN but new versions exists. It is worth exploring these alternatives(DDQN …)