A STUDY OF PRIMAL WASSERSTEIN IMITATION-LEARNING IN GOAL-CONDITIONED RL

SHENGDI CHEN[α]

Abstract. This project, conducted with Prof. Dr. J. Buhmann as supervisor and I. Ovinnikov as advisor, is executed within the framework of a Semester-Thesis under the catalogue identifier `401-3740-01L` at ETH Zürich.

RW/CSE, D-MATH, ETH Zürich: `shenchen␣AT␣ethz␣DOT␣ch | 17-943-358`.

Contents

Part 1. Preliminaries

1. Quick-Start

1.1. **Synopsis.** This thesis applies the Primal-Wasserstein Imitation-Learning algorithm as introduced in [Dad+20a] to a testing-framework utilizing a separate testing-environment

1.2. **Related Researches.** Key to the Primal-Wasserstein Imitation-Learning is the distance of its namesake, the Wasserstein distance, the p^{th}-order generalization of which is studied in [Vil08]. Generative Adversarial Networks (GANs) utilizing this distance is introduced in [ACB17].

1.2.1. *On Reinforcement-Learning.*

1.2.2. *On other Learning methodologies.* Behavioral Cloning

I

2. Technical Details

2.1. **Programming.** The source-code is packaged in one monolithic repository, which includes the testing-environment, training and testing modules for expert-

2.1.1. *General usages.* The programming is performed with the `Python` language with global type-hinting. Python versions no earlier than `Python 3.9` is required[1].

The source-code is consistently processed by the `black`-formatter [Wil+19] and thus implicitly conforms to the `PEP8` standard.

2.1.2. *Required packages.* The framework for the agent-interactive environment is provided by the `gym`-package developed by OpenAI [Bro+16].

Implementation of the Proximal-Policy-Optimization (PPO) algorithm [Sch+17] as provided by stable-baselines3 [Hil+18] (`sb3`), is used as the direct RL algorithms.

2.2. **Formatting and Typesetting.** This paper is compiled by the LuaTEX typesetting-engine featuring the open-source font-packages `Libertinus` and `Source-Code-Pro`.

2.3. **Version-Tracking.** The project source-code is distributed under the following repositories:

- The modules for the formulation of various agents, as well as those for training and testing are published under the repository: Link to GitHub;
- All other aspects of the Tetris-game, including the `engine` as described in Section ??, are found at: Link to GitHub;
- This thesis-document is accessible under: Link to GitHub.

[1] Version requirement of `Python`'s `typing`-Module: https://docs.python.org/3/library/typing.html#module-contents

Both source-code repositories deploy the »GNU Affero General Public License v3.0« license, else known as GNU AGPLv3.

This thesis documents the results obtained until version V?, marked with the signed git-tag of #tag-number.

Part 2. PWIL Fundamentals

3. Essentials from Primal Wasserstein IL

3.1. **The RL problem.** The Markov-Decision-Process (MDP) framework as introduced in [REF] characterizes agent-environment interaction over time-horizon $t \in T$ with action a_t drawn from the time-invariant action-space A leading to state transition:

$$S \in s_t \xrightarrow[\text{env}]{a_t} s_{t+1} \in S$$

of the environment, yielding a reward r_t.

The decision for the action a_t of an agent is provided by policy π of policy-space Π:

$$s_\star \xrightarrow{\pi} a_t$$

Central to Reinforcement-Learning (RL) is the search of the optimal policy $\pi^\star$ that maximizes the expected γ-discounted summation of all rewards r, otherwise referred to as the expected »return«.

3.2. **The Wasserstein Distance.** As discussed in [Vil08], the p^{th}-order Wasserstein[2] distance is defined as follows:

$$fff$$

$$\equiv \inf_{\theta\in\Theta} \left\{ \int \right\}$$

$$\mathcal{W} := \left(\inf_{\theta\in\Theta} \left\{ \mathop{\mathbb{E}}_{(x,y)} \left[d\left(x, y\right) \right] \right\} \right)^{1/p}$$

which, interpreted with the earth's movers analogy (Villani-2008), is equivalently formulated as follows:

$$\pi^\star := \inf_{\theta\in\Theta} \left\{ d\left(\right) \theta_\pi \right\}$$

[Dad+20a] enumerates various desirable properties of the Wasserstein distance: in particular, it is a true distance (in contrast with f-divergences used commonly in ?-Learning (IRL or adversary?)) with guaranteed smoothness.

3.3. **The PW-IL algorithm.** Solving for the optimal solution conforming to the above definition requires trajectory information of the entire episode (before the environment has to be reset for further agent inputs). This requirement is bypassed by the Primal Wasserstein IL (PW-IL) algorithm in [Dad+20a], where the following optimization problem is solved instead:

$$\pi^\star := \inf_{\theta\in\Theta} \left\{ d\left(\right) \theta_\pi \right\}$$

[2] named after Leonid Vaserštein (in Russian: Леонид Нисонович Васерштейн)

The optimizer is deliberately non-optimal with respect to the original Wasserstein distance: in fact, the underlying optimization-problem guarantees an upper-bound for the Wasserstein distance.

The non-optimal solution is based on the following »greedy-coupling«

Variation	`PointNav`-ID	Shift of First Target	Position of First Target
Default	0	$(0;\ 0)$	$(100;\ 100)$
`Var-1`	1	$(0;\ +50)$	$(100;\ 150)$
`Var-2`	2	$(+50;\ 0)$	$(150;\ 100)$

TABLE 4.2. The three different

Part 3. Configurations

4. The environment

A separate testing-environment is implemented where the agent is required to navigate from its starting position within a two-dimensional plane to reach target(s) unknown to the agent itself. This section introduces the »`PointNav`« environment and discusses its configuration options for the purpose of studying the PW-IL algorithm.

4.1. The agent and the targets.

The two-dimensional space containing the agent and the target(s) is referred to as the »board«. Without loss of generality, its size is chosen as 200×200 and internally represented by an `np.ndarray` of the same shape.

For this thesis, the starting position of the agent is fixed at $(10, 10)$. Random initialization, readily configurable through a boolean-flag, is left for future work.

In this study, two targets non-random positions are used, with the first located at the board center and the second at the $(190, 190)$. The ideal path traverses thus diagonally across the board from the lower, left corner to first reach the middle of the middle, followed by moving towards second target in the upper, right corner.

4.1.1. *Reward and Termination.*

The reward at time-step t at is calculated as the negative L^2 distance between the agent's position and that of the current target:

$$\text{reward}^{\texttt{PointNav}} := -\left(\left(x_t^{\text{agent}} - x^{\text{curr-target}}\right)^2 + \left(y_t^{\text{agent}} - y^{\text{curr-target}}\right)^2\right)^{1/2}$$

Implicitly, the reward-value is non-negative, with higher values indicating better performance.

An episode is terminated if both targets have been found or if the internal timer of 1000 time-steps has elapsed. Justification for this specific choice of setting the maximal episode-length to 1000 time-steps is provided below in Section .

4.2. The variants.

4.2.1. *Shifts.*

For controlled modifications of the environment `PointNav`, the first target is shifted off-center. In this project, two such variations are made. Table 4.2 documents the three variations with the original version without shifts of the first target.

Note that the default variant (marked green) without shifting the first target is the (only) environment in which that the PW-IL algorithm is trained on and evaluated against. One also

observes that Variation-1 and Variation-2 (marked blue) demonstrate symmetry with respect to the shifts.

4.2.2. *Action-Continuity.* Two modes of continuity for action-input are available to agents interacting with `PointNav`: the discrete and the continuous action-space. In the discrete mode, 5 possible actions are available to the agent, which are mapped to two-dimensional movement-inputs as follows:

$$A^{\text{discrete}} := \begin{bmatrix} \texttt{gym.spaces.Discrete(5)} \\ \triangleq \begin{bmatrix} 01234 \end{bmatrix}^\top \end{bmatrix} \longrightarrow \begin{bmatrix} (0;+2) \\ (0;-2) \\ (+2;0) \\ (-2;0) \\ (0;0) \end{bmatrix}$$

For the continuous variant, the action-space spans the range of $[-2.5;+2.5]$ for both dimensions. The x and y inputs are then rounded (towards 0[3]) before applying to the agent's position.

$$A^{\text{cont}} := \begin{bmatrix} \texttt{gym.spaces.Box(-2.5, +2.5)} \\ \texttt{gym.spaces.Box(-2.5, +2.5)} \end{bmatrix} \longrightarrow \texttt{np.round(}\star\texttt{)}$$

The range is chosen such that the final, rounded action applied to `PointNav` is uniformly distributed among $\{-2,-1,0,+1,+2\}$ for the random agent.

Note 1. Episode-Length

Given the maximal (absolute) displacement of 2 in one direction every step and the size of 200×200 of the board, one readily sees that the ideal agent would require 200 steps (100 per dimension) to reach both targets in the discrete `PointNav` environment and 100 steps (move diagonally, i.e., in both dimensions at the same time) in the continuous version, irrespective of the shift values of the first target.

The episode-length of 1000 time-steps as mentioned in Section 4.1.1 allows a considerable margin of tolerance for non-optimal behavior while guaranteeing baseline performance, as later discussed in Definition 4.

4.2.3. *The* 6 *variants.*

5. Expert-Demonstration Setup

5.1. **The expert-demonstrations.** As setup for deploying the PW-IL method, the »demonstration« from the expert-agent must first be generated. This section defines such demonstrations and describe the hyper-parameters available to the process of the generation.

Definition 1. Episode-Trajectories of an agent

[3]rounding performed with `np.round()`

An »episode-trajectory«, or simply »trajectory«, is an iterable collection of per-time-step recording of all state-action pairs generated by an agent during one entire episode, where such a state-action pair s-a is the concatenation of the environments state s and the agent's action a.

$$\texttt{traj} \coloneqq \left(\texttt{sa}_1, \text{s-a}_2, \cdots \text{s-a}_{\text{episode-over}}\right)$$

Per [Dad+20a], the »demonstration« for PW-IL transports information from expert trajectories.

Definition 2. Demonstration for PW-IL

For practical implementation, [Dad+20b] encodes this as an iterable collection of state-action pairs, as previously seen in the definition of the episode-trajectories.

The above definition hints at the usage of multiple expert-trajectories to generate PW-IL's demonstration. Indeed, the number of trajectories forming the pool for the demonstration constitutes a configurable hyper-parameter $h^{\text{\#-traj}}$. Inspired by the choice of 1 and 10 as values of $h^{\text{\#-traj}}$ in the original paper of PW-IL [Dad+20a], this thesis expands upon this with the addition of an intermediate value 5:

$$h^{\text{\#-traj}} \in \{1, 5, 10\}$$

For each of these trajectories, sub-sampling at frequency $h^{\text{s-smp}}$ is performed by selecting state-action pairs once every certain number of time-steps. Assume thus that within a selected episode-trajectory, the state-action pair at the k^{th} time-step is chosen, the next state-action to pick is the one at the $(k + h^{\text{s-smp}})^{\text{th}}$ time-step.

While the original thesis [Dad+20a] uses the uniform sub-sampling frequency of 20 to simulate data scarcity, this study deploys the following value-range

$$h^{\text{s-smp}} \in \{1, 2, 5, 10, 20\}$$

The sub-sampled state-action pairs of all $h^{\text{\#-traj}}$ are finally concatenated to form the demonstration as input for executing the PW-IL algorithm.

5.2. **The source.** Besides the number of trajectories $h^{\text{\#-traj}}$ and the sub-sampling frequency $h^{\text{s-smp}}$, the quality of the demonstration itself is configurable, as also studied in [Bro+19]; [Jac+19]. In this thesis, the quality of the expert-demonstration is controlled by adding experts to the trajectory pool trained on *variations* of the `PointNav`. Allowing the presence of such variation trajectories in the pool constitutes the non-optimal demonstration quality. One further distinguishes between the »mixed« pool, where expert-trajectory on the default, non-shifted variation-0 is included and the »distant« pool, where only the shifted variations are taken into consideration. By subsequently varying the specific constellation within the optimal, the mixed and non-optimal pools, one observes the 7 demonstration-sources, identified with integer 0 to 6 respectively in Table 5.2.

5.3. **Summary of hyper-parameters.**

Demo-ID	Pool of Experts ID	Demo-Quality
0	$\{0\}$	optimal
1	$\{0,1\}$	non-optimal: Mixed
2	$\{0,2\}$	
3	$\{0,1,2\}$	
4	$\{1\}$	non-optimal: distant
5	$\{2\}$	
6	$\{1,2\}$	

TABLE 5.2. Time-stamps of key activities

Definition 3. Configuration-Space of PW-IL

Three The configurable hyper-parameters of PW-IL is thus understood as follows:

$$h := \begin{bmatrix} h^{\text{source}} \\ h^{\#\text{-traj}} \\ h^{\text{s-smp}} \end{bmatrix} \in \begin{bmatrix} \{0,1,\cdots,6\} \\ \\ h^{\text{s-smp}} \end{bmatrix}$$

Assessment	Episode-Length m^{Length}	Reward m^{Reward}	Assessment
Insufficient	> 1000	$< -1.2\,\text{e}5$	Insufficient
Baseline	< 1000	$< -7.0\,\text{e}4$	Meaningful

TABLE 6.1. tab: Performance Metrics

Part 4. Evaluation and Comparison

6. Analysis

6.1. **Framework.** The following *a priori* performance-metrics are used to evaluate the performance of an agent interacting with the testing-environment `PointNav`.

Definition 4. Performance metrics and baselines

The baseline is defined as reaching both targets before episode terminates, i.e., the episode length $m^{\text{L}} \leq 1000$, where lower m^{Length} values indicate less time required to attain the target positions, thus better performance.

Alternatively, the reward m^{Reward} returned by the environment is used to gauge the agent's performance. However, as the reward of `pointnav` is calculated as the negative L^2 distance of the agent to the target (see Section 4.1.1), achieving higher reward value does not necessarily indicate actually reaching the targets: an agent »hovering« around the target in close proximity will effectively reduce the L^2 distance close to 0, thus leading to high reward-values. Thus, instead of a baseline, the reward m^{Reward} constitutes a loose guideline for training effectiveness.

Empirical observations suggest that a random agent displays episode-reward $m^{\text{Reward}} \lessapprox -1.2\text{e}5$, whereas signs of »meaningful« training are found for reward-values greater than $-7\,\text{e}4$.

6.2. **The experts.** Using the PPO-algorithm as introduced in [Sch+17] and implemented by the reference RL-library `sb3` [Hil+18], 6 expert-agents are trained: 3 on the discrete-action `PointNav` environment, 3 on the continuous counterpart. It shall be noted that other algorithms from RL, such as A2C [Mni+16] or DQN [Mni+15] can be utilized. The experts

Per the assessment criteria nominated previously, the experts evidently solve the `PointNav` problem. Further more, despite being ultimately discretized with rounding of the action, the continuous `PointNav` is solved more robustly than the discrete version, in particular, the variation-2 of the discrete model demonstrates high standard-deviation with respect to episode-length m^{Length}.

7. Results of PW-IL

7.1. **The optimal.** The optimal performance of the PW-IL agents are presented in Figure 7.1.

One observers that

Continuity	`PointNav` Variation	Ep-Length m^{Length}	Reward m^{Reward}
Discrete	0	$-1.284\,\text{e}4 \pm 573.6$	239.1 ± 44.73
	1	$-1.427\,\text{e}4 \pm 526.3$	257.8 ± 22.86
	2	$-1.918\,\text{e}4 \pm 2.016\,\text{e}3$	406.1 ± 75.37
Continuous	0	$-5.221\,\text{e}3 \pm 329.3$	88.2 ± 0.4
	1	$-1.194\,\text{e}4 \pm 498.2$	244.2 ± 57.4
	2	$-9.065\,\text{e}3 \pm 321.5$	188.3 ± 38.26

TABLE 6.2. Performance of the experts with respect to the two metrics m^{Length} (lower is better) and m^{Reward} (higher is better), formatted as $(\text{avg} \pm \sigma)$.

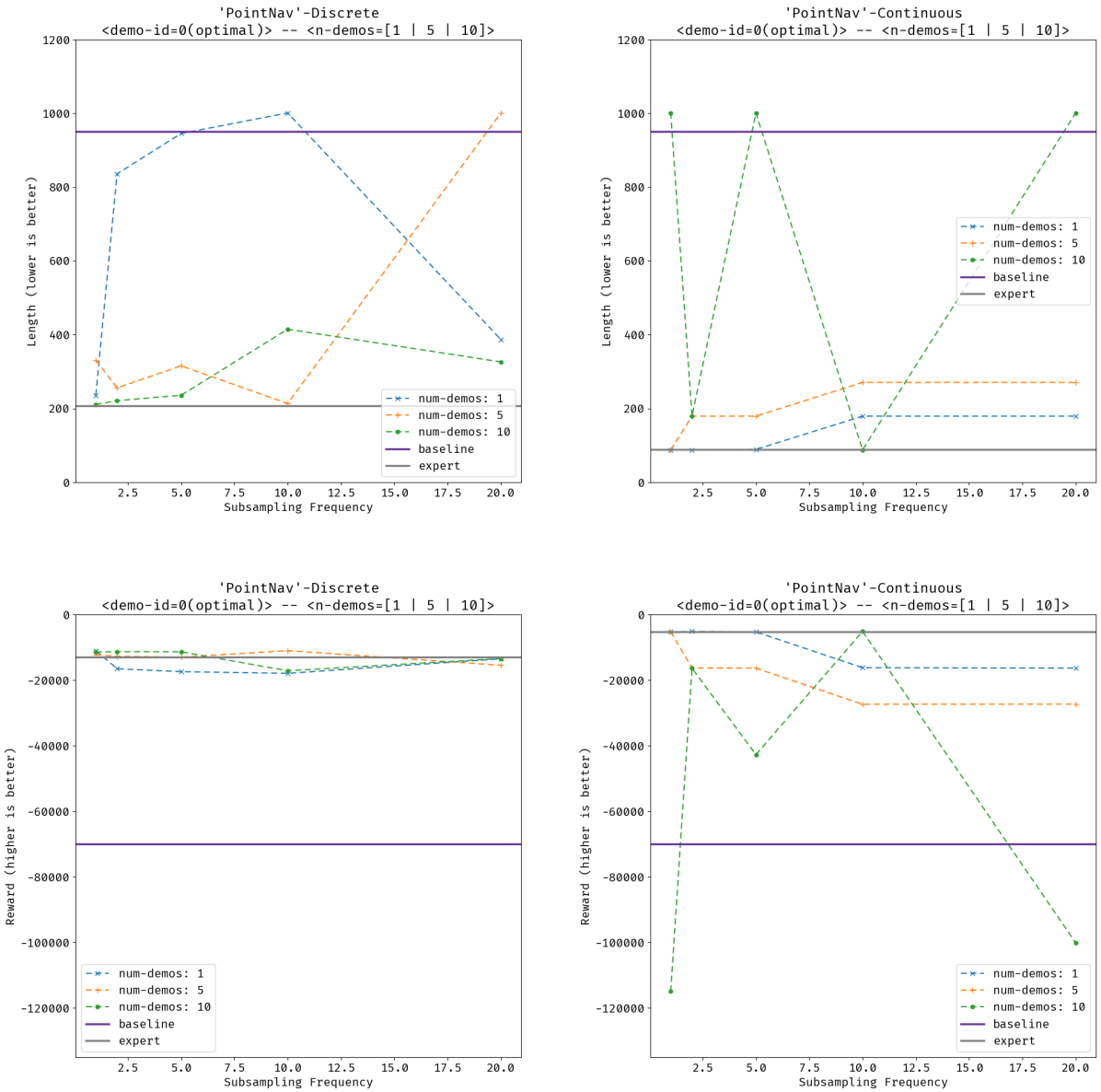

FIGURE 7.1. PW-IL agents

Activities	Duration (×week)
literature survey; project structuring	0.5
testing of relevant source-code and algorithm	1
development of testing framework	1.5
source-code refactoring; expansion of testing	1
collection of results; composition of thesis	1

TABLE 8.2. Time-stamps of key activities

Part 5. **Epilogue**

8. EXECUTIVE SUMMARY

8.1. **The gist.** This project studies the performance of the Primal Wasserstein IL as introduced in [Dad+20a] under varying conditions of:

(1) Quality of demonstrations by expert;
(2) Number of demonstrations available to the algorithm
(3) Sample scarcity with sub-sampling frequency

To this end, a testing-framework targeted at varying these essential configurations is created. A

8.2. **Timeline.** The administrative details of this thesis can be retrieved under the course-catalogue of ETH Zürich under the registration number `401-3740-01L`. Its total duration is estimated at approximately 5 weeks of full-time work, including the drafting of the thesis and the composing of the source-code. Specifics are found in Table 8.2.

8.3. **Future work.**

Definition 5. Part 5

(1) Future work:
    (a) more environments with this framework (cartpole)
    (b) further testing of PWIL
        (i) with IRL (AIRL)

9. ACKNOWLEDGEMENTS

To the supervisor of this thesis, Prof. Dr. J. Buhmann: my deepest appreciation for the acceptance of the original idea of this thesis in its earliest infancy, for the inspiring and philosophical approach to the field of machine-learning as a whole, and, of course, for granting the much needed extension for completion;

To my advisors, Ivan, Ami and Eugene: my heart-felt thanks to the extended sessions of talks, the continuous support and the constructive suggestions;

To my family and my friends: my sincere gratitude for the unyielding support and care;

Finally, to J. Neubauer: all my respect for his natural mastery of the game and brilliant deliveries in the most competitive environments. Rest in peace among these beloved pieces, *maestro*.

References

1. Arjovsky, M., Chintala, S. & Bottou, L. *Wasserstein GAN* 2017. https://arxiv.org/abs/1701.07875.
2. Bellman, R. *Dynamic Programming* 1st ed. (Princeton University Press, Princeton, NJ, USA, 1957).
3. Bertsekas, D. P. *Dynamic programming and optimal control* (1995).
4. Brockman, G. *et al. OpenAI Gym* eprint: arXiv:1606.01540.
5. Brown, D. S., Goo, W., Nagarajan, P. & Niekum, S. *Extrapolating Beyond Suboptimal Demonstrations via Inverse Reinforcement Learning from Observations* 2019. https://arxiv.org/abs/1904.06387.
6. Chen, S. *This Thesis: A Study of Primal Wasserstein Imitation-Learning in Goal-Conditioned RL* https://github.com/shengdichen/sem_tss/blob/main/src/main.pdf.
7. Dadashi, R., Hussenot, L., Geist, M. & Pietquin, O. *Primal Wasserstein Imitation Learning* 2020. https://arxiv.org/abs/2006.04678.
8. Dadashi, R., Hussenot, L., Geist, M. & Pietquin, O. *Primal Wasserstein Imitation Learning Source-Code Repository* https://github.com/google-research/google-research/tree/master/pwil.
9. Dulac-Arnold, G., Mankowitz, D. & Hester, T. *Challenges of Real-World Reinforcement Learning* 2019. https://arxiv.org/abs/1904.12901.
10. Hill, A. *et al. Stable Baselines*
11. Jacq, A., Geist, M., Paiva, A. & Pietquin, O. *Learning from a Learner* in *International Conference on Machine Learning* (2019).
12. Mnih, V. *et al.* Asynchronous Methods for Deep Reinforcement Learning. https://arxiv.org/abs/1602.01783 (2016).
13. Mnih, V. *et al.* Human-level control through deep reinforcement learning. *Nature* **518,** 529–533 (2015).
14. OpenAI. *OpenAI Spinning Up in Deep RL* eprint: arXiv:1606.01540.
15. Schulman, J., Wolski, F., Dhariwal, P., Radford, A. & Klimov, O. *Proximal Policy Optimization Algorithms* 2017. https://arxiv.org/abs/1707.06347.
16. Silver, D. *et al.* Mastering the game of Go without human knowledge. *Nature* **550,** 354–359 (2017).
17. Sutton, R. S. & Barto, A. G. Reinforcement Learning: An Introduction. *IEEE Transactions on Neural Networks* **16,** 285–286 (2005).
18. Tesauro, G. Temporal Difference Learning and TD-Gammon. *J. Int. Comput. Games Assoc.* **18,** 88 (1995).
19. Villani, C. in, xxii+973 (Jan. 2008).
20. Willing, C. *et al. The uncompromising Python code formatter*