We thank the reviewers for their valuable feedback. We are glad that they found our approach novel and promising, and agree that further details would facilitate the understanding of Algo. 1. Next, we answer to the presented comments: positioning in the literature for curriculum learning, scalability of our approach, experiments details, and applicability.

Finding ways to automatically design a sequence of learning tasks that are increasingly *harder* to solve for the agent is a challenge in curriculum learning. For *supervised* learning, [7][1] showed that gradually increasing the *entropy* of the training distribution helped. However in RL, breaking down a task in sub-problems that can be ordered by difficulty is non trivial [2]. In robotics, [1, 3] proposed to start from the *goal* (*e.g.*, open a door) and give a starting state that is gradually further from that goal. These methods assume at least one known goal state that is used as a seed for expansion. For video games, [4] adapted the concept with a starting state increasingly further from the end of a *demonstration*. However, here, the goal is not to "reach a particular end state": there is no goal state at all. Rather, what we want is to "take an optimal decision at each time step", and, particularly, take optimal decisions at the beginning of the game, *i.e.*, at the highest level of the multilevel optimization problem, given that all subsequent actions *will* be optimal. Thus, contrary to [1, 3, 4], we do not "reverse time" to artificially build a sequence of tasks starting further from a goal state and subsequently harder to solve in the hope of learning how to reach this goal from all possible starting states, but rather *stack* new optimization problems on top of previous ones, which gradually increases the *computational complexity* of the task, in order to learn to act optimally in optimization problems with an increasing number of levels. For example, in the case of MCN$_w$, solving the last stage (protection) is NP-hard whereas solving the bilevel min-max (attack knowing the subsequent protection will be optimal) is Σ_2^p-hard, and the trilevel problem (vaccination knowing the attacker will react optimally knowing we will be able to find an optimal protection after) is Σ_3^p-hard [45]. Thus, contrary to most problems in RL, here we are faced with a task *naturally* constituted of a hierarchy of sub-problems ordered by their position in the Polynomial Hierarchy, which motivates a curriculum. The one we devised is based on an *afterstate value function*, raising scalability issues as mentioned in our discussion: this is exactly what we leave as a future direction. The paper's methodology will benefit its pursuer. So we must stress the impact of this work as the first looking to such problems and setting the ground for less expensive curricula.

In Operations Research, most of the multilevel combinatorial problems studied have less than 4 levels in practice: finding exact methods to solve bilevel and trilevel problems is still an active area of research; note that an MBC with L levels is potentially Σ_L^p-complete. Thus, even-though scaling our method to more levels is straightforward, we did not tested it as finding reference problems for such situations is rare. This also explains the scarcity of benchmarks for trilevel problems and our choice of focusing on the MCN: there is a methodology to solve the problem along with a publicly available dataset of exactly solved instances. We used this dataset to *evaluate* MultiL-Cur in Table 2: these instances were never seen before by our agent. To *train* our agent, we generated our own dataset of instances, and used as targets the *approximate* values given by the `Greedy Rollout` procedure. For the validation, we arbitrarily set T_{val} to a relatively low value of 20. In Table 2, only 71% of the 120 instances of size 100 generated for [1] were solved by MCN^{MIX} under their threshold of 2h; we only considered those as they are the only ones with a solution. But, if we had added a 2h lower bound for each non solved instance in our time average for MCN^{MIX}, the entry for graphs of size 100 would report 2690s instead of the 848s. Thus, generating a test set of exactly solved instances of larger size would take time with the existing methods, explaining why we did not try to benchmark the abilities of our heuristic on significantly larger graphs. Finally, in Table 2, we did not compare our curriculum to DA-AD in MCN$_{dir}$ and MCN$_w$ as [1] did not have results on this; we only adapted their exact method (Appendix C). Plus, from the complexity point of view, these cases should be much more expensive for DA-AD [45]. So our goal was to show a meaningful comparison for the simplest case. To gain more hindsight on the metrics η and ζ, one can look at Fig.2 of [30] and Fig.5 of [33].

Regarding the usefulness of such problems for practical scenarios, the MCN could fit on several applications, *e.g.* to limit the fake news spread in social networks or in cyber security for the protection of a botnet against malware injections. In the latter, the attacker infects nodes by introducing a malware in some bots, the defender vaccinates and protects nodes by disconnecting them, stopping the spread of the malware.

References

[1] Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. *arXiv preprint arXiv:1707.05300*, 2017.

[2] Alex Graves, Marc G. Bellemare, Jacob Menick, Remi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks. *arXiv preprint arXiv:1704.03003*, 2017.

[3] Boris Ivanovic, James Harrison, Apoorva Sharma, Mo Chen, and Marco Pavone. Barc: Backward reachability curriculum for robotic reinforcement learning. *arXiv preprint arXiv:1806.06161*, 2018.

[4] Tim Salimans and Richard Chen. Learning montezuma's revenge from a single demonstration. *arXiv preprint arXiv:1812.03381*, 2018.

[1] citations in blue refer to the bibliography of the paper.