We thank the reviewers for their helpful comments and constructive criticism. We would like to respond to the reviewers' questions and correct misunderstandings, focusing on comments by R4.

R1

In response to your questions:

1. While it could be expected that our method would perform at least as well as the meta-greedy policy, we also found that our method performed significantly better than the blinkered approximation (and its generalization to sequential decision problems).
2. We believe that our paper's lasting contribution will be to bring machine learning methods to bear on meta-reasoning problems. This is the first time that machine learning has been applied to learn an approximation to rational metareasoning.

We will correct the typos you identified, and clarify that the statement that our method is applicable to a wider range of problems refers to the original blinkered approximation whereas our generalization of this method appears applicable to a similar range of problems.

R2

Thank you; we will make the examples in the Discussion more concrete. For instance, our method could be applied to metareasoning about whether to make the decision to issue an evacuation warning to an area that might get hit by a hurricane based on an initial coarse weather simulation (terminate deliberation) or to run additional more accurate but time-consuming weather simulations to decrease the risk of a costly false alarm or costly false negative.

R3

Thank you for your positive feedback! We will clarify our mathematical notation and correct the typos you identified.

R4

Thank you for your thorough feedback! We believe that your 3 most serious concerns are based on unfortunate misunderstandings:

1. We apologize we were not sufficiently clear about the distinction between observation vs. computation. The two are easy to conflate because learning fromthrough external observations vs. simulationnternal simulations would be formally identical if the agent had a perfect world model. However, the two problems are different in that the outcomes of actions are rewards whereas simulations generate information only.

2. You expressed the concern that the savings from choosing computations according to the learned meta-level policy might be offset by the time it takes to learn the meta-level policy such that the sum of training time and solution time would be larger than the runtime of a standard algorithm. This concern seems to be based on the implicit misunderstanding that the learned algorithm is used only once, but the truth is that it can be reused across millions of problem instances. Thus, as long as they learned policy is more efficient than the naive algorithm that would have been used instead, the initial time invested to learn it will be amortized eventually. Our paper proposes an automatic method for algorithm discovery, and it would clearly be unfair to demand that the time it takes to develop a new sorting algorithm has to be less than the amount of time it saves on sorting a single list.
3. You criticized that our evaluations do not account for the time it takes to execute the learned meta-level policy. This is a legitimate criticism: meta-reasoning is only worthwhile if the meta-level computational cost is less than the saved object-level computational cost. However, this is an open problem in metareasoning research. We see our method as a step in this direction because it is more computationally efficient than the blinkered approximation whose original formulation was intractable for the larger instances we used in our third evaluation. Furthermore, the utility of metareasoning depends on the the ratio of the costs associated with object-level and meta-level computations. Our method could improve total runtime when computational actions are very costly. For example, when predicting the path of a hurricane, a single computational action might entail running a computationally intensive weather simulation on a supercomputer.

Minor comments:

1. Thank R4 for pointing out a typo we will correct and two relevant papers we will gladly cite in the revised version.
2. Eq. 8 defines the VPI as the expected improvement in decision-quality that would be achieved by having perfect information about the environment in expectation over all environments that the agent might be in. It does not depend on the true value of theta, because the agent cannot possibly know theta before it has computed it.
3. We will gladly add that in evaluation 1 the optimization algorithm was Infinite Metric Gaussian Process Optimization (Kawaguchi, Kaelbling, Lozano-Perez, 2016) which has exponential convergence, and in evaluations 2 and 3, Bayesian optimization of the weights was performed with the function gp_minimize from the scikit-optimize library (Louppe & Kumar, 2016). Both algorithms are known to be very sample-efficient.
4. The computational complexity of our method depends on how the features are computed. The evaluations reported in the paper computed the features exactly through enumeration. This makes the runtime of our method linear with the size of the meta-level state-space. We are currently investigating whether it is possible to achieve a similar level of performance with better runtime complexity by approximating the features through Monte-Carlo integration, analytic approximations, or deep learning. This is work in progress that was not included in the submission. We will clarify this in the revision.

5. Our framework addresses the problem that deliberation may cause the agent to miss time-sensitive opportunities to collect rewards by including these opportunity costs in the cost of computation, cost(c).
6. The finite horizon in our evaluation problems serves the purpose of evaluation: It makes tractable to compute the optimal meta-level policy by backward induction and thereby allows us to determine how close the learned meta-level policy is to being optimal.
7. The term `reinforcement learning` is used differently in different communities. Our learning algorithm performs policy search which is commonly considered as a form of reinforcement learning (especially in the deep RL literature) because it interacts with an MDP environment to learn a policy from experienced rewards. This algorithm tradeoffs exploration against exploitation by trying out those policy parameters according to a combination of their expected return and its uncertainty.