Optimal Direct Policy Search

Tobias Glasmachers and Jürgen Schmidhuber

IDSIA, University of Lugano
6928, Manno-Lugano, Switzerland
{tobias,juergen}@idsia.ch

Abstract. Hutter's optimal universal but incomputable AIXI agent models the environment as an initially unknown probability distribution-computing program. Once the latter is found through (incomputable) exhaustive search, classical planning yields an optimal policy. Here we reverse the roles of agent and environment by assuming a computable optimal policy realizable as a program mapping histories to actions. This assumption is powerful for two reasons: (1) The environment need not be probabilistically computable, which allows for dealing with truly stochastic environments, (2) All candidate policies are computable. In stochastic settings, our novel method Optimal Direct Policy Search (ODPS) identifies the best policy by direct universal search in the space of all possible computable policies. Unlike AIXI, it is computable, model-free, and does not require planning. We show that ODPS is optimal in the sense that its reward converges to the reward of the optimal policy in a very broad class of partially observable stochastic environments.

1 Introduction

Reinforcement learning (RL) algorithms are often categorized into model-based and model-free approaches. Model-based methods typically learn a model of the environment and its dynamics, to be used for decision making in a second step. Advantages include potentially flexible adaptation to new tasks expressed by different reward functions within the same environmental dynamics. Disadvantages include the high sample complexity of model learning: typically, many interactions with the environment are needed to estimate the underlying dynamic process with sufficient certainty.

Value function approaches do not require a full predictive model; instead they compress knowledge about the environment into a single number per state or state-action-pair, to model the only task-specific quantity of interest in most RL settings, namely, expected future reward. Since single number estimation is often more efficient and the value function is a sufficient statistics for learning the optimal policy, such methods can often reduce the sample complexity.

Model-free algorithms search for good policies without ever building a model of the environment. A particular branch of model-free methods is direct policy search [4], which includes policy gradient algorithms or evolution strategies applied to a fitness function based on the reward of an episode or a fixed time

J. Schmidhuber, K.R. Thórisson, and M. Looks (Eds.): AGI 2011, LNAI 6830, pp. 52–61, 2011.

frame. Such algorithms are relatively blind to the particularities of the environment (e.g., state transitions) and the reward signal (they only care for accumulated reward). One of the advantages of this blindness is that they are not affected by classical problems such as partial observability. Direct policy search is particularly well suited for complex environments solvable by relatively simple policies.

All of the above paradigms can be found in nature at various levels of complexity. Higher-developed animals, in particular humans, use anticipation and planning to solve many complex tasks, which to some extent requires a model of the environment. On the other hand, findings in neuroscience studies have been connected to value-function learning [8]. Many simpler animals exhibit complex but inflexible patterns of basically pre-programmed behavior, which can be understood as the result of evolutionary direct policy search.

For most complex tasks in a human-dominated and dynamic world, model-based approaches are intuitively expected to excel in the long run. However, when confronted with highly unfamiliar situations, humans are able to resort to trial and error strategies that often allow them to act better than chance, until having collected enough experience to build a sufficiently good predictive model enabling them to resort to the familiar planning paradigm. Thus, direct policy search may have its place as a bootstrapping method even in elaborate value-function or model-based approaches.

Optimal planning is hard, but identifying a computational model of an arbitrarily complex but computable environment is even harder. The AIXI model [1] does it in a provably optimal way, identifying the correct environmental model through a minimal number of interactions with the environment, but this requires a continual and exhaustive and computationally intractable systematic search in all of program space, necessarily ignoring all issues of computational complexity. (Note, however, recent serious efforts to scale down AIXI for realistic applications [9].)

The main contribution of this paper is a theoretical, conceptual one. While direct policy search is sometimes viewed as a 'last resort' heuristic, here we show that one can use it as a general and asymptotically optimal RL method *for all episodic POMDP problems*, asymptotically on the same level as the model-based AIXI scheme, but, unlike AIXI, incrementally computable (although not necessarily practically feasible). It covers any other behavior-generating searcher, such as realistically downscaled AIXI variants, by systematically enumerating *all* such algorithms. In fact, many direct policy search algorithms can trivially be understood as (non-optimal) approximations of ours.

2 Reinforcement Learning in POMDPs

We consider an agent interacting with an environment in discrete time steps $t \in \mathbb{N}$. In each time step the environment (including the agent) is in a Markov state $s_t \in \mathbb{S}$. Let $\mathcal{D}(X)$ denote the space of probability distributions over a measure space X. In each time step the agent perceives an observation $o_t \in \mathbb{O}$ of the state s_t, described by $o_t \sim \Omega_{s_t} \in \mathcal{D}(\mathbb{O})$.

The agent performs an action $a_t \in \mathbb{A}$, which results in a transition to state s_{t+1}. The dynamics of the system are described by $s_{t+1} \sim P_{s_t,a_t} \in \mathcal{D}(\mathbb{S})$. After acting the agent obtains a task-specific feedback in terms of a scalar reward signal $r_{t+1} \in \mathbb{R}$, possibly depending on s_t, a_t, and s_{t+1}. Having arrived in the new state s_{t+1}, it also obtains the next observation o_{t+1}, and the process starts over.

By $h_t = (o_1, a_1, r_1, \ldots, o_{t-1}, a_{t-1}, r_{t-1}, o_t)$ we denote the history at time t. Let $\mathbb{H}$ denote the space of all possible histories. Then a (behavior) policy is a description of how the agent acts in its environment, expressed as a mapping $\pi : \mathbb{H} \to \mathcal{D}(\mathbb{A})$. We collect all such mappings in the set $\mathit{\Pi}$ of policies.

Assume that the subset $\mathbb{T} \subset \mathbb{S}$ of terminating states is non-empty. We denote the time of termination by the random variable T. We call a task episodic if the suprema of expectation and variance of T are finite: $\sup\left\{\mathbb{E}^\pi[T] \,\middle|\, \pi \in \mathit{\Pi}\right\} < \infty$ and $\sup\left\{\mathbb{E}^\pi[(T - \mathbb{E}^\pi[T])^2] \,\middle|\, \pi \in \mathit{\Pi}\right\} < \infty$. These prerequisites corresponds to the standard technical assumption of ergodicity.

2.1 MDPs and POMDPs

The underlying Markov decision process (MDP) is described by the tuple $(\mathbb{S}, \mathbb{T}, \mathbb{A}, S, P, R)$, where $S \in \mathcal{D}(\mathbb{S})$ is the distribution of start states, $P_{s,a}$ denotes the probability distribution of transitioning from state s to the next state $s' \sim P_{s,a}$ when taking action a, and $R(s, a, s') \in \mathcal{D}(\mathbb{R})$ is the distribution of rewards obtained for this transition. In what follows we make the standard assumptions that $\mathbb{S}$ and $\mathbb{A}$ are finite, and that for each combination of s, a, and s' expectation and variance of $R(s, a, s')$ are finite. We are only interested in episodic tasks.

Now, a partially observable Markov decision process (POMDP) is a tuple $(\mathbb{S}, \mathbb{T}, \mathbb{A}, \mathbb{O}, S, P, \Omega, R)$, where Ω (as defined above) stochastically maps states to observations. We assume that $\mathbb{O}$ is finite. POMDPs are a rather general class of environments for reinforcement learning. Most classical RL algorithms are restricted to MDPs, because their learning rules heavily rely on the Markov property of the state description. However, in many realistic tasks observations are naturally stochastic and/or incomplete, such that the environment has to be treated as a POMDP.

2.2 Objectives of Learning

The goal of learning a policy π in an episodic task is usually defined to maximize future expected reward $\rho = \mathbb{E}^\pi\left[\sum_{t=1}^{T-1} r_t\right]$. However, in what follows we leave the choice of the objective function ρ open and just require the form $\rho(\pi) = \mathbb{E}^\pi[\eta(h_T)]$, where the 'success' function η as a function of the full episode h_T has finite expectation and bounded variance. This is a reasonable assumption automatically fulfilled for the future expected reward, since expectation and variance of both T and $R_{s,a,s'}$ are bounded.

Note that there is an important conceptual difference between the functions η and ρ: While $\eta(h_T)$ is a quantity measurable via interaction with the environment, evaluating the objective function $\rho(\pi)$ requires knowledge of the POMDP, which is assumed to be opaque to the reinforcement learner.

2.3 Direct Search in Policy Space

Direct policy search is a class of model-free reinforcement learning algorithms. A direct policy search algorithm searches a space Π of policies π by means of direct search, for example, an evolutioary algorithm. Such search schemes are particularly suitable for learning in POMDP environments, because they make no assumptions about observations being Markovian, or treatable as nearly Markovian. The search is direct in the sense that it relies solely on evaluations of the objective function, here on the stochastic version η, often stabilized by averaging over multiple episodes.

3 Optimal Direct Policy Search

The aim of the present paper is to lift direct policy search for stochastic RL tasks to the level of universal search [2] with a Turing complete programming language, encompassing all possible computable policies. Thus, our approach is closely connected to deterministic universal searchers [2,5], but lifted to the class of *stochastic* RL tasks. In contrast to universally optimal [1] approaches such as the Gödel machine [6] we do not require the concept of proof search.

The basic idea of our approach amounts to systematically applying the enumerable set of computable policies to the task in a scheme that makes the statistical evaluation of each such policy more and more reliable over time. At the same time we make sure that the fraction of time spent on exploitation (in contrast to systematic exploration) tends to one. This combination allows us to derive an optimality theorem stating that the reward obtained by our agent converges towards the optimal reward.

Consider a Turing machine that receives the current history h_t as input on its tape and outputs an action by the time it halts. This can be achieved by initializing the tape with a default distribution over actions, which may or may not be changed by the program.

The basic simple idea of our algorithm is a nested loop that simultaneously makes the following quantities tend to infinity: the number of programs considered, the number of trials over which a policy is averaged, the time given to each program. At the same time, the fraction of trials spent on exploitation converges towards 1.

Letting the number of programs go to infinity allows for finding policies encoded by programs with arbitrarily high indices. At the same time, each program is given more and more but always finite execution time, circumventing the halting problem in the style of universal search [2]. Averaging the reward over more and more episodes makes these estimates arbitrarily reliable, which is essential

to prove optimality. Finally, letting the fraction of episodes used for exploitation in online mode tend to one makes sure that the overall performance of the algorithm tends to the supremum of the performances of all computable policies.

To specify the optimal direct policy search algorithm we need the following notation:

- Let $p : \mathbb{N} \to \mathcal{P}$ be an enumeration of all programs $\mathcal{P}$ of the Turing machine in use, with p_i denoting the i-th program.
- Let π_p^c denote the policy obtained by running program p for c steps. This policy, understood as a mapping from the history $h_t \in \mathbb{H}$ to a distribution over actions, is implemented as follows: We write a 'default' distribution over actions and the history h_t onto the tape of the Turing machine. Then we run program p_i for c steps, or less, if it halts by itself. During this time the program may perform arbitrary computations based on the history, and overwrite the distributions over actions (which can, e.g., be represented as a soft-max decision). We also write π_i^c instead of $\pi_{p_i}^c$ for program p_i.
- Let $(N_n)_{n\in N}$, $(C_n)_{n\in N}$, $(E_n)_{n\in N}$, and $(X_n)_{n\in N}$ be sequences of natural numbers, all tending to infinity:
  - N_n denotes the number of programs considered in epoch n,
  - C_n denotes the time given to each program,
  - E_n denotes the number of episodes over which the reward of each policy is averaged,
  - X_n denotes the number of episodes for exploitation in online mode.

  We need the following easily satisfiable technical conditions on these sequences: (a) $\lim_{n\to\infty} N_n/E_n = 0$, (b) $\lim_{n\to\infty} (N_n \cdot E_n)/X_n = 0$.

Algorithm 1. Optimal Direct Policy Search (ODPS)

```
for n ∈ ℕ do
    v_n ← −∞; b_n ← 0;
    for i ∈ {1, ..., N_n} do
        w ← 0;
        for e ∈ {1, ..., E_n} do
            perform one episode according to π_i^{C_n}, resulting in h_T;
            w ← w + η(h_T);
        end
        R_i ← w/E_n;
        if R_i > v_n then v_n ← R_i; b_n ← i;
    end
    if online-mode then perform X_n episodes according to π_{b_n}^{C_n};
end
```

With these conventions, the pseudo-code of Optimal Direct Policy Search (ODPS) is given in Algorithm 1. The algorithm operates in epochs, each starting with an exploration phase during which a number of candidate policies is systematically evaluated. ODPS has two modes: In exploration mode it evaluates more and more

powerful policies in each epoch, without ever exploiting its findings. At the end of each episode, after the direct policy search is stopped, it assumes to have found the best so far solution for later exploitation phases. In online mode, ODPS is actually exploiting the policies it finds, achieving arbitrarily close to optimal performance in the limit. But exploitation phases are still interleaved with exploration phases, to make sure no good policy is missed by chance.

4 Formal Optimality

The following theorems formalize optimality of ODPS:

Theorem 1. *Consider the infinite sequence of episodes executed by Algorithm 1 in online-mode, and let η_j denote the success $\eta(h_T)$ of the j-th episode. Let $T_n = \sum_{k=1}^{n} N_k E_k + X_k$ denote the number of episodes at the end of epoch n. By $\bar{\eta}_n = \frac{1}{T_n} \sum_{j=1}^{T_n} \eta_j$ we denote the success averaged over the first n epochs. Then it holds $\bar{\eta}_n \xrightarrow[n\to\infty]{\Pr} \sup \left\{ \rho(\pi_p^c) \,\middle|\, p \in \mathcal{P}, c \in \mathbb{N} \right\}$, where $\xrightarrow{\Pr}$ denotes convergence in probability.*

Theorem 2. *Consider Algorithm 1 either in online or in exploration mode. Let b_n be the index of the best program found in epoch n. Then it holds $\rho(\pi_{b_n}^{C_n}) \xrightarrow[n\to\infty]{\Pr} \sup \left\{ \rho(\pi_p^c) \,\middle|\, p \in \mathcal{P}, c \in \mathbb{N} \right\}$, where $\xrightarrow{\Pr}$ denotes convergence in probability.*

Proof (Proof of Theorem 2). We define $\rho^* = \sup \left\{ \rho(\pi_p^c) \,\middle|\, p \in \mathcal{P}, c \in \mathbb{N} \right\}$. Fix constants $\varepsilon > 0$ and $\delta > 0$. We have to show that there exists $n_0 \in \mathbb{N}$ such that for all $n > n_0$ it holds $\Pr(\rho^* - \rho(\pi_{b_n}^{C_n}) > \varepsilon) < \delta$.

Let $R_i^{(n)}$ denote the average success of policy $\pi_i^{C_n}$ during the exploration phase of epoch n, and let $\rho_i^{(n)} = \rho(\pi_i^{C_n})$ denote the corresponding objective function value. Furthermore, let $\rho_n^* = \max \left\{ \rho_i^{(n)} \,\middle|\, i \in \{1, \ldots, N_n\} \right\}$ denote the best achievable objective function value in epoch n. We have $\mathbb{E}[R_i^{(n)}] = \mathbb{E}[\eta(h_T) \,|\, \pi_i^{C_n}] = \rho_i^{(n)}$ and $\mathrm{Var}(R_i^{(n)}) \leq \frac{\sigma^2}{E_n}$, where σ^2 is a bound on the variance of η. This allows us to estimate

$$
\begin{aligned}
& \Pr\left(\exists\, i \in \{1, \ldots, N_n\} \text{ s.t. } \left| R_i^{(n)} - \rho_i^{(n)} \right| \geq \frac{\varepsilon}{4} \right) \\
\leq\ & N_n \cdot \Pr\left(\left| R_i^{(n)} - \rho_i^{(n)} \right| \geq \frac{\varepsilon}{4} \right) \qquad \text{for all } i \in \{1, \ldots, N_n\} \\
\leq\ & \frac{N_n}{E_n} \cdot \frac{16\sigma^2}{\varepsilon^2} \leq \delta
\end{aligned}
$$

for all $n > n_1$. The first step follows from the union bound, the second one from Chebyshev's inequality. According to property (a) there exists $n_1 \in \mathbb{N}$ such that for all $n > n_1$ we have $\frac{N_n}{E_n} < \frac{\delta\varepsilon^2}{16\sigma^2}$, which implies the last step for this choice of n_1. Together with $R_{b_n}^{(n)} \geq R_i^{(n)}$ this implies

$$
\rho(\pi_{b_n}^{C_n}) \geq \ R_{b_n}^{(n)} - \frac{\varepsilon}{4} \geq R_i^{(n)} - \frac{\varepsilon}{4} \geq \rho_n^* - \frac{\varepsilon}{2} \ ,
$$

for all $i \in \{1, \ldots, N_n\}$ and $n > n_1$, and with probability of at least $(1 - \delta)$.

Per construction we have $\lim_{n\to\infty} \rho_n^* = \rho^*$. Thus, there exists $n_2 \in \mathbb{N}$ such that for all $n > n_2$ it holds $\rho^* - \rho_n^* < \varepsilon/2$. With $n_0 = \max\{n_1, n_2\}$ it follows that $\rho(\pi_{b_n}^{C_n}) > \rho_* - \varepsilon$ holds for all $n > n_0$ with probability of at least $(1 - \delta)$.

Proof (Proof of Theorem 1). Property (b) guarantees that $\tilde{\eta}_n - \rho(\pi_{b_n}^{C_n}) \xrightarrow[n\to\infty]{\Pr} 0$, where $\tilde{\eta}_n$ is the average performance during epoch n. It is easy to see that this also implies $\bar{\eta}_n - \rho(\pi_{b_n}^{C_n}) \xrightarrow[n\to\infty]{\Pr} 0$. Now we apply Theorem 2, which proves the assertion.

5 Discussion

Theorems 1 and 2 formally establish the asymptotic optimality of ODPS for all finite episodic POMDP problems, as outlined in the previous sections. In exploration mode, the algorithm finds arbitrarily close to optimal policies. In online mode, it exploits its previously learned knowledge, resulting in asymptotically optimal performance.

An interesting aside is that the ODPS algorithm, together with a simulation of the POMDP, is a program itself and is thus available as a subroutine in some of the programs $p \in \mathcal{P}$. This self-reference is neither helpful nor does it pose any problem to the procedure. In particular, the ODPS algorithm does not reason about itself, and in contrast to the Gödel machine [7] does not attempt to optimize its own search procedure.

Let us play a number of variations on the theme. For example, we may ask for the best policy that computes its action in *a priori* limited time. This scenario is more realistic than allowing the agent to take arbitrary long (but finite) time to make its decisions. The restricted time requirement simplifies the problem at hand considerably. Note that any program terminating after at most $n \in \mathbb{N}$ time steps can be expressed with at most n instructions, limiting the search space to a finite subset. Thus, all epochs can in principle search the full set of programs, and the sequence N_n disappears from the algorithm.

We do not claim that ODPS is a practical algorithm for solving real problems efficiently. How to scale it down? The probably most important step is to insert prior bias by giving small indices in the enumeration p_i to programs believed to be likely problem solvers. In many cases such programs would be known to halt in advance, which is why the progressively increasing sequence in computation time C_n could be set to infinity, enabling us to transfer information from early episodes to later ones in a straight-forward manner, such that the number of evaluations in episode n can be reduced from E_n to $E_n - E_{n-1}$. The language $\mathcal{P}$ can even be restricted to a non-Turing complete (typically finite) subset [3].

6 Experimental Test

The goal of our experiment is demonstrate the typical behavior of ODPS. We use a non-trivial POMDP environment, but in order to render the search practical we rely on a non-Turing complete encoding of policies tailored to the task.

The agent is confronted with the 49-state POMDP depicted in Figure 1. Its states are organized into a 6×8 grid, plus the single terminating state on the right. The two possible start states, each with probability $1/2$, are the two circled states at the top left. There are exactly two possible observations $\mathbb{O} = \{0, 1\}$ and two actions $\mathbb{A} = \{a_1, a_2\}$.

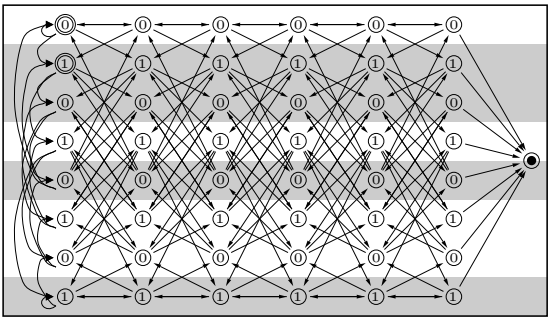

Fig. 1. Illustration of the 49-state POMDP used in the experiments. The two circled states in the top left are selected as start states with equal probability. The number in each circle is the observation the agent perceives. The two actions a_1 and a_2 have opposite effects in the gray and white background rows: In the gray rows, a_1 moves right and a_2 moves left, where each of the two possible arrows to the left and to the right are followed with equal probability. Actions are unrelibale, having the opposite effect in 30% of the cases. In the white rows the roles of actions are reversed. The goal is to reach the terminating state as quickly as possible. This is expressed by the reward uniformly distributed in the interval $[-1, 0]$ for all states.

The mapping from states to observations is deterministic; the symbol observed in each state is given in the figure. Note that the agent observes only a single bit of information per state, which makes partial observability a serious problem.

The effects of actions are highly stochastic. In rows with gray background action a_1 moves the agent to the right, and action a_2 to the left, but the effects of the actions are reversed in 30% of the cases. In rows with white background the roles of a_1 and a_2 are reversed. Furthermore, there are two possible destination states for each step, which are chosen with equal probability. In each time step the agent receives a reward drawn from the uniform distribution on the interval $[-1, 0]$. Thus, the goal of the agent is to reach the terminating state as quickly as possible.

The reason for using this seemingly over-complex POMDP is that direct policy search is particularly well suited for this type of task, because the optimal policy is relatively simple. It depends only on the last three observations (o_{t-2}, o_{t-1}, o_t) in the form $\pi(0, 0, 1) = \pi(0, 1, 0) = \pi(1, 0, 0) = \pi(1, 1, 1) = a_1$ and $\pi(0, 0, 0) = \pi(0, 1, 1) = \pi(1, 0, 1) = \pi(1, 1, 0) = a_2$. This encoding of the optimal policy is much simpler than an encoding of its value function in the MDP formulation, not to speak of the difficulties when learning in the corresponding POMDP.

Consequently we encode policies as tables, mapping tuples of most recent observations to actions. We enumerate the possible values of n_o, the number of previous observations taken into account; for each size we systematically enumerate all possible combinations of actions. There are 2^{n_o} such observations, and $2^{(2^{n_o})}$ tabular policies. In early interactions when there are fewer observations

available than processed by the policy, the observation vector is padded with zeros accordingly. This encoding scheme is not unique, which is typical when encoding programs. For example, p_1 and p_3 both always execute action a_1, in the form $p_1(\cdot) = a_1$, in contrast to $p_3(0) = a_1$, $p_3(1) = a_1$.

We ran ODPS with the settings $N_n = 10n$, $E_n = \lceil n \log(n+1) \rceil$, and $X_n = 10E_n^3$ for 100 epochs. These sequences fulfill properties (a) and (b), such that Theorems 1 and 2 hold when the number of epochs is unlimited. In the above encoding the optimal policy happens to appear for the first time as p_{128}. Thus, it is in the search space from epoch 13 on.

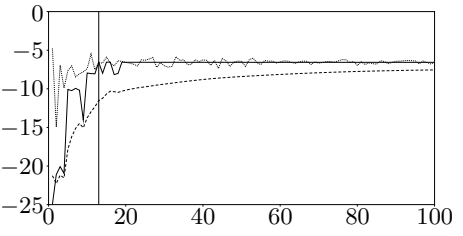

Fig. 2. The graph shows the best exploration performance v_n (dotted curve), the performance during exploitation (solid curve), and the total average performance (dashed curve), over the course of the first 100 episodes. The vertical line at episode 13 marks the point from where on the optimal policy p_{128} is available.

We monitored different quality indicators summarized in Figure 2. We can observe a number of typical behaviors from the graph. In the first iterations the number E_n of episodes per evaluation is too small, resulting in a huge gap between the dotted and the solid curve (overfitting). After about epoch 20 the exploitation performance becomes stable. The same holds for the identification of the best policy. Although this policy is available from episode 13 on, it is not identified reliably until epoch 20, after which the number E_n of averages is large enough. Thus, ODPS in exploration mode is reliably successful roughly from epoch 20 on. The dotted curve indicates the total averaged performance in online mode and converges only slowly towards the optimum. This is because in each epoch a large number of episodes is allocated to systematic exploration, during which the average performance is around -25 (not shown in the graph).

7 Conclusion

We introduced a class of direct policy search methods that in theory can optimally solve any finite, stochastic, episodic POMDP problem. Our results are extremely general in the sense that they make very few assumptions on the underlying MDP. In particular, our approach treats stochasticity naturally and does not need the assumption of a deterministically or probabilistically computable environment. Our only restriction is that the optimal policy must be relevant to machine learning, in the weak sense that it is a *computable* function of the history.

Our ODPS algorithm is not necessarily practical, but most direct policy search schemes are closely related (typically replacing systematic search with heuristics, and the search space of all computable policies with a restricted subspace),

and we outlined how to scale it down to realistic applications, describing an illustrative experiment with a highly stochastic POMDP.

Still, this work remains conceptual in spirit, and, in the authors' point of view, serves the sole purpose of providing a solid theoretical understanding of the power of direct policy search. Our work shows the *principal* ability of direct policy search to solve almost arbitrary problems optimally, and thus provides a solid theoretical justification for its use.

References

1. Hutter, M.: Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability. Springer, Berlin (2004)
2. Levin, L.: Universal sequential search problems. Problems of Information Transmission 9(3), 265–266 (1973)
3. Schaul, T., Schmidhuber, J.: Towards Practical Universal Search. In: Proceedings of the Third Conference on Artificial General Intelligence, Lugano (2010)
4. Schmidhuber, J.: Sequential decision making based on direct search (Lecture Notes on AI 1828). In: Sun, R., Giles, C.L. (eds.) IJCAI-WS 1999. LNCS (LNAI), vol. 1828, p. 213. Springer, Heidelberg (2001)
5. Schmidhuber, J.: Optimal Ordered Problem Solver. Machine Learning 54, 211–254 (2004)
6. Schmidhuber, J.: Gödel machines: Fully Self-Referential Optimal Universal Self-Improvers. In: Goertzel, B., Pennachin, C. (eds.) Artificial General Intelligence, pp. 119–226 (2006)
7. Schmidhuber, J.: Ultimate Cognition *à la* Gödel. Cognitive Computation 1(2), 177–193 (2009)
8. Schultz, W., Dayan, P., Montague, P.R.: A neural substrate of prediction and reward. Science 275(5306), 1593 (1997)
9. Veness, J., Ng, K.S., Hutter, M., Silver, D.: A Monte Carlo AIXI Approximation. Technical Report 0909.0801, arXiv (2009)