On Connections between Constrained Optimization and Reinforcement Learning

Nino Vieillard[1,2], Olivier Pietquin[1], and Matthieu Geist[1]

[1]Google Research, Brain Team.
[2]Université de Lorraine, CNRS, Inria, IECL, F-54000 Nancy, France.

Abstract

Dynamic Programming (DP) provides standard algorithms to solve Markov Decision Processes. However, these algorithms generally do not optimize a scalar objective function. In this paper, we draw connections between DP and (constrained) convex optimization. Specifically, we show clear links in the algorithmic structure between three DP schemes and optimization algorithms. We link conservative Policy iteration to Frank-Wolfe, Mirror-Descent Modified Policy Iteration to Mirror Descent, and Politex (Policy Iteration Using Expert Prediction) to Dual Averaging. These abstract DP schemes are representative of a number of (deep) reinforcement learning (RL) algorithms. By highlighting these connections (most of which have been noticed earlier, but in a scattered way), we would like to encourage further studies linking RL and convex optimization, that could lead to the design of new, more efficient, and better understood RL algorithms.

1 Introduction

A standard way to model sequential decision making is through the frame of Markov Decision problems (MDPs). These MDPs, when known and small enough, can be solved with Dynamic Programming (DP), of which two classical algorithms are policy iteration (PI) and value iteration (VI). Solving an MDP means finding a policy π_* such that its value v_{π_*} dominates the value of any other policy, state-wise. As such, it cannot be easily framed as a (convex) optimization problem, even if connections can be made: VI can be (indirectly) seen as minimizing the residual $\|T_* v - v\|$ through a fixed-point approach, and PI can be obtained as Newton descent on the same residual (*e.g.*, [Pérolat et al., 2016] and references therein).

Reinforcement learning aims at solving sequential decision making problems through interactions. Many RL algorithms can be seen as approximate and possibly asynchronous and stochastic variations of DP algorithms, mainly PI and VI. As such, connections of these algorithms to convex optimization are quite loose. A notable exception is algorithms based on policy gradient, that aim at maximizing the expected value function, $J(\pi) = \mathbb{E}_{s\sim\mu}[v_\pi(s)]$, or a proxy of this objective function [Deisenroth et al., 2013]. However, this function is not concave, which is often a problem from an optimization perspective. Another exception is the set of algorithms that aim at minimizing the residual $J(v) = \|T_* v - v\|$. Again, this is not convex in general and this approach comes with drawbacks [Geist et al., 2017].

In this work, we describe three connections between DP schemes and convex optimization algorithms, from an algorithmic perspective, and without considering estimation or approximation errors. We relate the Frank-Wolfe algorithm [Frank and Wolfe, 1956] to Conservative Policy Iteration [Kakade and Langford, 2002] in Section 3. Then, we establish a connection

Optimization Foundations for Reinforcement Learning Workshop at NeurIPS 2019, Vancouver, Canada.

between Mirror Descent [Beck and Teboulle, 2003] and Mirror Descent Modified Policy Iteration [Geist et al., 2019] in Section 4, and finally between Dual Averaging [Nesterov, 2009] and Politex [Lazic et al., 2019] in Section 5. Most of these connections have been noticed earlier, but in a scattered way, and sometimes not clearly. We also discuss briefly more connections between RL and optimization in Section 6.

2 Background

2.1 Markov Decision Processes

We consider the problem of solving infinite horizon discounted Markovian Decision Process (MDP). An MDP is a tuple $\{\mathcal{S}, \mathcal{A}, P, r, \gamma\}$, where $\mathcal{S}$ and $\mathcal{A}$ are finite spaces of respectively states and actions, $P \in \Delta_{\mathcal{S}}^{\mathcal{S}\times\mathcal{A}}$ is a Markovian transition kernel (writing Δ_X the simplex over the set X), $r \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ a reward function and $\gamma \in (0,1)$ the discount factor. The objective of dynamic programming is to find a policy π_* that dominates every policy π in the space of policies that we denote Π. A policy $\pi \in \Pi$ is a function that associate to each state $s \in \mathcal{S}$ a distribution over $\mathcal{A}$, $\pi(\cdot|s)$. It defines an associated stochastic kernel, $P_\pi(s'|s) = \mathbb{E}_{a\sim\pi(\cdot|s)}[P(s'|s,a)]$, and an associated expected reward function $r_\pi(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}[r(s,a)]$. The value $v_\pi \in \mathbb{R}^{\mathcal{S}}$ of a policy associates to each state the expected discounted cumulative reward,

$$v_\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \,\middle|\, s_0 = s, s_{t+1} \sim P(\cdot|s_t, a_t), a_t \sim \pi(\cdot|s_t)\right].$$

To every policy is associated a Bellman evaluation operator, defined for each $v \in \mathbb{R}^{\mathcal{S}}$ as $T_\pi v = r_\pi + \gamma P_\pi v$. The value function of a policy is the unique fixed point of this operator. The Bellman optimality operator is defined as $T_* v = \max_\pi T_\pi v$, and the optimal value function v_* verifies $v_* = T_* v_*$. We define the set of greedy policies w.r.t. to any value v as $\mathcal{G}v = \{\pi | T_\pi v = T_* v\}$. An optimal policy π_* is such that $\pi_* \in \mathcal{G}v_*$.

The state-action value function $q_\pi \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ associated to a policy π is defined as

$$q_\pi(s,a) = r(s,a) + \gamma \mathbb{E}_{s'\sim P(\cdot|s,a)}[v_\pi(s')].$$

This state-action value function possesses similar properties to the value function, In particular, we can easily re-write the Bellman equations using state-action values instead of values, and we have that $T_\pi q_\pi = q_\pi$ and $T_* q_{\pi_*} = q_{\pi_*} = q_*$. The greediness can be defined w.r.t to a state-action value by $\mathcal{G}q = \text{argmax}_a\, q(\cdot, a)$.

Two classic DP algorithms are PI and VI. PI iterates as follows, $\pi_{k+1} \in \mathcal{G}(v_{\pi_k})$, its convergence being guaranteed by the fact that for any policy π, $v_{\mathcal{G}v_\pi} \geq v_\pi$, the equality being obtained only at optimality. VI iterates as follows, $v_{k+1} = T_* v_k$, its convergence being guaranteed by the fact that the Bellman operator is a γ-contraction. In both cases, the argument for convergences are loosely related to convex optimization. Both schemes can be generalized to modified policy iteration (MPI), that iterates as

$$\begin{cases} v_k = T_{\pi_k}^m v_{k-1} \\ \pi_{k+1} \in \mathcal{G}v_k. \end{cases}$$

We retrieve PI with $m = \infty$ (in this case, we compute the fixed point of the contractive operator T_{π_k}) and VI with $m = 1$ (noticing that $v_k = T_{\pi_k} v_{k-1} = T_{\mathcal{G}v_{k-1}} v_{k-1} = T_* v_{k-1}$). This can be seen as a PI scheme where the full evaluation of the policy is relaxed to a partial, less costly, evaluation.

We define the objective function $J \in \mathbb{R}^{\Delta_{\mathcal{A}}^{\mathcal{S}}}$ as

$$J(\pi) = \mathbb{E}_{s\sim\mu}\left[v_\pi(s)\right],$$

with μ a given distribution of states. We aim at solving $\max_{\pi\in\Delta_{\mathcal{A}}^{\mathcal{S}}} J(\pi)$. This non-concave optimization problem can be addressed for example with a gradient ascent, and it is the basis of a number of RL algorithms [Deisenroth et al., 2013], either directly or indirectly.

For example, Conservative Policy Iteration [Kakade and Langford, 2002] or Trust Region Policy Optimization (TRPO) [Schulman et al., 2015] were designed for optimizing a proxy to this objective function. An interesting observation is that the natural gradient of $J(\pi)$, $\tilde{\nabla} J(\pi)$, in the tabular case (and with a softmax representation for the policy), is indeed the state-action value function, that is $\tilde{\nabla} J(\pi) = q_\pi$. This is a direct corollary of the results of Kakade [2002]. We'll use this observation later.

2.2 Convex Optimization

We study the problem of finding the maximum of a concave function over a convex set, instead of the standard setting where one aims to find the minimum of a convex function. This allows us to draw more natural connections to RL, without loss of generality. More precisely, given a concave function f taking its values in $\mathbb{R}$ and a closed convex space $\mathcal{X}$, we define a maximization problem as finding $x_* \in \mathcal{X}$ such that

$$f(x_*) = \max_{x \in \mathcal{X}} f(x). \tag{1}$$

We consider methods derived from the gradient ascent scheme. This classic algorithm assumes that f is (sub)differentiable, and produces a sequence (x_k), with η a learning rate,

$$x_{k+1} = x_k + \eta \nabla f(x_k).$$

There is no reason for this sequence to stay in $\mathcal{X}$, and a classical solution is projected gradient ascent, that additionally project back the iterate onto the convex set,

$$\begin{cases} y_{k+1} = x_k + \eta \nabla f(x_k) \\ x_{k+1} = \Phi_{\mathcal{X}}(y_{k+1}), \end{cases}$$

with $\Phi_{\mathcal{X}}$ the projection onto $\mathcal{X}$. We'll discuss other algorithms while drawing connections to RL in the following sections.

3 Conservative Policy Iteration and Frank Wolfe

We start by linking the classic Frank Wolfe algorithm [Frank and Wolfe, 1956], also known as Conditional Gradient Descent, to the DP scheme Conservative Policy Iteration (CPI, Kakade and Langford [2002]). This link was first made by Scherrer and Geist [2014], who studied a boosting generalization of CPI based on Frank-Wolfe, but without mentioning it, and was explicitly given by Shani et al. [2019].

Frank–Wolfe Frank-Wolfe (FW) addresses problem (1). Given the current iterate x_k and its gradient $g_k = \nabla f(x_k)$, it searches for the element of $\mathcal{X}$ being the most colinear to the gradient, and update the iterate by doing a convex mixture of these elements, which ensure that the iterates stay in the convex set. Formally, it iterates as follows, with $x_0 \in \mathcal{X}$ for initialization:

$$\begin{cases} g_k = \nabla f(x_k) \\ s_k = \operatorname{argmax}_{x \in \mathcal{X}} \langle x, g_k \rangle \\ x_{k+1} = (1-\alpha) x_k + \alpha s_k. \end{cases} \tag{2}$$

Compared to projected gradient ascent, it might be easier in some problems to find the most colinear element than to project the iterate onto the convex set.

Conservative Policy Iteration CPI is a classic RL algorithms that soften the PI scheme by replacing the greedy policy by a stochastic mixture of the greedy policy and the previous one. It iterates as follows:

$$\pi_{k+1} = (1-\alpha)\pi_k + \alpha \mathcal{G} q_{\pi_k},$$

with $\mathcal{G} q_{\pi_k}$ any greedy policy respectively to q_{π_k}. Without approximation, the optimal rate is $\alpha = 1$, and the algorithm is of practical interest when there is approximation, but we stick to the exact case, the aim being to draw connections to optimization.

Let $\mu \in \Delta_{\mathcal{S}} > 0$ be a distribution over states[1], and define the scalar product on $\mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ as $\langle q_1, q_2\rangle = \sum_s \mu(s) \sum_a q_1(s,a) q_2(s,a)$. The greedy policy $\mathcal{G}q_{\pi_k}$ can be equivalently seen as the policy maximizing $\langle \pi, q_{\pi_k}\rangle$. Recall also that the state-action value function is indeed the natural gradient of J, $\tilde{\nabla} J(\pi) = q_\pi$. Given this, we can rewrite the update of CPI as follows:

$$\begin{cases} q_k = q_{\pi_k} = \tilde{\nabla} J(\pi_k) \\ \pi'_k = \text{argmax}_{\pi \in \Delta_{\mathcal{A}}^{\mathcal{S}}} \langle \pi, q_k \rangle \\ \pi_{k+1} = (1-\alpha)\pi_k + \alpha \pi'_k. \end{cases} \tag{3}$$

Comparing Eqs. (2) and (3), it is clear that CPI is indeed the Frank-Wolfe algorithm applied to the objective function $J(\pi)$, up to the fact that the vanilla gradient is replaced by the natural gradient[2]. We are not aware of a FW algorithm considering natural gradients in the optimization literature.

CPI being a variation of the PI scheme, we can also write an MPI-like variation of CPI, as considered by Vieillard et al. [2019]. This gives the following iterate (with π_0 and q_0 for initialization), that generalizes scheme (3):

$$\begin{cases} q_k = T_{\pi_k}^m q_{k-1} \\ \pi'_k = \text{argmax}_{\pi \in \Delta_{\mathcal{A}}^{\mathcal{S}}} \langle \pi, q_k \rangle \\ \pi_{k+1} = (1-\alpha)\pi_k + \alpha \pi'_k. \end{cases}$$

With $m < \infty$, we no longer have a FW scheme, but the algorithmic approach is very similar. In addition to the process updating the policies, we have an additional process that replaces the gradient computation, this process iterating value functions based on the Bellman operator of the current policy. With this point of view, the function q_k plays a role similar to the gradient, giving the direction of improvement. Notice that this scheme converges, despite the lack of concavity [Vieillard et al., 2019] (but the proof does not rely on optimization-like arguments).

4 MD-MPI and Mirror Descent

Here, we show a connection between the Mirror Descent (MD, Beck and Teboulle [2003]) optimization method and the DP scheme Mirror Descent Modified Policy iteration (MD-MPI, Geist et al. [2019]).

Mirror Descent Let D_Ω be the Bregman divergence generated by a convex function $\Omega \in \mathbb{R}^{\mathcal{X}}$, $D_\Omega(x||x') = \Omega(x) - \Omega(x') - \langle \nabla\Omega(x'), x - x'\rangle$. Classic Bregman divergences are the Euclidean distance, generated by the ℓ_2 norm, or the Kullback-Leibler (KL) divergence, generated by the negative entropy. Mirror Descent solves a maximization problem by iterating as follows:

$$\begin{cases} g_k = \nabla f(x_k) \\ x_{k+1} = \text{argmax}_{x \in \mathcal{X}} \, \eta \langle x, g_k \rangle - D_\Omega(x||x_k). \end{cases} \tag{4}$$

In words, MD consists at each step in maximizing a linearization of the function of interest, with the constraint of not moving to far from the previous iterate, this constraint being quantified by the Bregman divergence. It generalizes classic gradient ascent, that can be retrieved in the unconstrained case by considering the Euclidean distance.

MD-MPI Geist et al. [2019] introduced a set of abstract DP algorithms to study the propagation of errors of regularized RL algorithms, where the greediness is softened by a convex regularizer (typically, a scaled entropy or KL divergence, see the paper for details). MD-MPI considers the general case where the greediness is regularized by a Bregman

[1] Notice that the original CPI considers the distribution $d_{\mu,\pi_k} = (1-\gamma)\mu(I - \gamma P_{\pi_k})^{-1}$ (the γ-weighted occupancy measure induced by π_k for the initial state distribution μ) instead of μ, which is of importance for the analysis in the approximate setting, but does not change the algorithm in the exact setting considered here.

[2] With the vanilla gradient, the derivation would be a bit more invovled, but we would obtain the original CPI, that is with μ being replaced by d_{μ,π_k}, see the derivations of Scherrer and Geist [2014].

divergence, within an MPI-like scheme. We consider a slight variation here, where greediness is computed in expectation with respect to μ (instead of state-wise in the original work).

Let $\Omega : \Delta_{\mathcal{A}} \to \mathbb{R}$ be a convex function, and define (with a slight abuse of notation) the convex function $\Omega : \Delta_{\mathcal{A}}^{\mathcal{S}} \to \mathbb{R}$ as $\Omega(\pi) = \sum_s \mu(s)\Omega(\pi(.|s))$. We first consider a PI-like update of MD-MPI (that is, setting $m = \infty$, or MD-PI), and recall that $q_{\pi_k} = \tilde{\nabla} J(\pi_k)$. The corresponding update rule can be written as

$$\begin{cases} q_k = q_{\pi_k} = \tilde{\nabla} J(\pi_k) \\ \pi_{k+1} = \text{argmax}_{\pi \in \Delta_{\mathcal{S}}^{\mathcal{A}}} \eta \langle \pi, q_k \rangle - D_\Omega(\pi || \pi_k). \end{cases} \tag{5}$$

As before, comparing Eqs. (4) and (5), MD-PI is exactly mirror descent applied to $J(\pi)$, up to the fact that the vanilla gradient is replaced by the natural one. We're not aware of any MD algorithm doing so in the optimization literature.

With a KL divergence, this scheme is very close to TRPO, the main differences being that the distribution d_{μ,π_k} (defined in footnote 1) is used instead of μ, at each iteration. Contrary to CPI, that can be seen exactly as Frank-Wolfe with vanilla gradient (see footnote 2), Eq. (5) with the vanilla gradient would not give TRPO. The reader can refer to Shani et al. [2019], who draw connections between MD and TRPO, and show that TRPO is a kind of modified MD.

We can also write the original MD-MPI (up to the slight variation mentioned before):

$$\begin{cases} q_k = T_{\pi_k}^m q_{k-1} \\ \pi_{k+1} = \text{argmax}_{\pi \in \Delta_{\mathcal{S}}^{\mathcal{A}}} \eta \langle \pi, q_k \rangle - D_\Omega(\pi || \pi_k). \end{cases}$$

As it was the case for CPI, we obtain a mirror descent scheme, up to the fact that the gradient is replaced by a second iterative process on the q-values. In both domains, the idea is similar: the goal is to move in the direction of improvement – either a new point or a new policy –, this direction being provided by the gradient or the q-value, but staying not too far from the previous point or policy.

5 Politex and Dual Averaging

Finally, we show a new connection between the recent Politex (Policy Iteration Using Expert Prediction) algorithm [Lazic et al., 2019], and the Dual Averaging (DA, Nesterov [2009]) optimization scheme.

Dual Averaging Dual Averaging can be considered as a lazy version of MD (see Bubeck et al. [2015, Section 4.4]), and iterates as follows

$$\begin{cases} g_k = \nabla f(x_k) \\ x_{k+1} = \text{argmax}_{x \in \mathcal{X}} \eta \langle x, \sum_{j=0}^k g_j \rangle - \Omega(x). \end{cases} \tag{6}$$

In some cases, it can be shown to be equivalent to MD. This algorithmic scheme can also be linked to Follow the Regularized Leader (FTRL), that is somehow a building block of Politex (the related paper focusing indeed on prediction with expert advice, which is related).

Politex Politex is a PI scheme, in the average reward case, that, at each iteration, estimates the value of its current policy, and computes a new policy that is the softmax of the sum of all q-values of the preceding policies. Notice that a softmax policy is a greedy policy regularized by an entropy (*e.g.*, see Geist et al. [2019]). Here, we first consider this algorithm in the cumulative discounted reward case, generalized to any regularizer. In this case, we can write a generalization of Politex as

$$\begin{cases} q_k = q_{\pi_k} = \tilde{\nabla} J(\pi_k) \\ \pi_{k+1} = \text{argmax}_{\pi \in \Delta_{\mathcal{S}}^{\mathcal{A}}} \eta \langle \pi, \sum_{j=0}^k q_j \rangle - \Omega(\pi). \end{cases} \tag{7}$$

Comparing Eqs. (6) and (7), we see that this variation of Politex is exactly DA with a natural gradient in lieu of the vanilla one.

Table 1: Summary of connections and analogies.

Convex Optimization	Reinforcement Learning
x	π
$f(x)$	? ($J(\omega)$ in tabular case)
$\nabla f(x)$	q_π
Frank–Wolfe	Conservative Policy Iteration
Mirror Descent	MD-MPI (type 2)
Dual Averaging	Politex

This can be easily generalized to an MPI-like scheme:

$$\begin{cases} q_k = T_{\pi_k}^m q_{k-1} \\ \pi_{k+1} = \operatorname{argmax}_{\pi \in \Delta_{\mathcal{S}}^{\mathcal{A}}} \eta \langle \pi, \sum_{j=0}^k q_j \rangle - \Omega(\pi). \end{cases}$$

Again, this is DA, up to the fact that the gradient is replaced by an iterative process for computing q-values.

6 Other connections

There are other works, not discussed above, that draw connections between optimization and RL. We mention briefly some of them here.

Bertsekas and Tsitsiklis [1996] see $v - T_* v$ as the gradient of an unknown function, and this has served as a building block to Goyal and Grand-Clement [2019] to introduce a Nesterov-like VI algorithm, with improved convergence rate.

Another classic DP approach, less used in RL, compute the optimal value function as the solution of a linear program. Neu et al. [2017] build on this to study regularized MDPs, where optimal policies are computed as the solution of real convex problems. The practical issue is that it involves working in the dual space (on state-action distributions), which might be not practical (as required quantities might not be easy to estimate).

Agarwal et al. [2019] study the convergence of various policy gradient algorithms, and despite the lack of concavity, their proof techniques borrow to optimization (and allow showing even global convergence in some cases).

7 Discussion

Using an analogy between the gradient and the state-action value function, we have shown a strong connection between some constrained convex optimization and RL algorithms. We have made this connection clear on three cases: Frank Wolfe and Conservative policy iteration, Mirror Descent and MD Modified Policy Iteration, and Dual Averaging and Politex. A summary of these connections is presented in Table 1. All these algorithms are improvements of canonical algorithms in their fields, namely Projected Gradient Descent in optimization and policy iteration in dynamic programming. They all share a same idea of regularization, either by a stochastic mixture of consecutive policies (like in CPI), or by explicitly regularizing the improvement step (like in mirror descent, that regularizes its improvement with a divergence).

From these three examples, we see how some methods have been identified in Optimization and in RL, and this sheds light and links that could be exploited in RL. Indeed, such a connection could be useful for either designing new algorithms in RL, or giving new techniques of analysis to better understand these algorithms. Drawing more formal links between optimization and RL (for example, linking the contraction in RL to strong convexity or Lipschitzness in optimization) could also be useful to strengthen the general connection between these fields.

References

Alekh Agarwal, Sham M Kakade, Jason D Lee, and Gaurav Mahajan. Optimality and approximation with policy gradient methods in markov decision processes. *arXiv preprint arXiv:1908.00261*, 2019.

Amir Beck and Marc Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003.

Dimitri P Bertsekas and John N Tsitsiklis. *Neuro-dynamic programming*, volume 5. Athena Scientific Belmont, MA, 1996.

Sébastien Bubeck et al. Convex optimization: Algorithms and complexity. *Foundations and Trends in Machine Learning*, 8(3-4):231–357, 2015.

Marc Peter Deisenroth, Gerhard Neumann, Jan Peters, et al. A survey on policy search for robotics. *Foundations and Trends® in Robotics*, 2(1–2):1–142, 2013.

Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956.

Matthieu Geist, Bilal Piot, and Olivier Pietquin. Is the bellman residual a bad proxy? In *Advances in Neural Information Processing Systems*, pages 3205–3214, 2017.

Matthieu Geist, Bruno Scherrer, and Olivier Pietquin. A theory of regularized markov decision processes. In *Proceedings of the International Conference on Machine Learning*, 2019.

Vineet Goyal and Julien Grand-Clement. A first-order approach to accelerated value iteration. *arXiv preprint arXiv:1905.09963*, 2019.

Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, volume 2, pages 267–274, 2002.

Sham M Kakade. A natural policy gradient. In *Advances in neural information processing systems*, pages 1531–1538, 2002.

Nevena Lazic, Yasin Abbasi-Yadkori, Kush Bhatia, Gellert Weisz, Peter Bartlett, and Csaba Szepesvari. Politex: Regret bounds for policy iteration using expert prediction. In *Proceedings of the International Conference on Machine Learning*, pages 3692–3702, 2019.

Yurii Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical programming*, 120(1):221–259, 2009.

Gergely Neu, Anders Jonsson, and Vicenç Gómez. A unified view of entropy-regularized markov decision processes. *arXiv preprint arXiv:1705.07798*, 2017.

Julien Pérolat, Bilal Piot, Matthieu Geist, Bruno Scherrer, and Olivier Pietquin. Softened approximate policy iteration for markov games. In *Proceedings of The International Conference on Machine Learning*, pages 1860–1868, 2016.

Bruno Scherrer and Matthieu Geist. Local policy search in a convex space and conservative policy iteration as boosted policy search. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 35–50. Springer, 2014.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the International Conference on Machine Learning*, pages 1889–1897, 2015.

Lior Shani, Yonathan Efroni, and Shie Mannor. Adaptive trust region policy optimization: Global convergence and faster rates for regularized mdps. *arXiv preprint arXiv:1909.02769*, 2019.

Nino Vieillard, Olivier Pietquin, and Matthieu Geist. Deep conservative policy iteration. *arXiv preprint arXiv:1906.09784*, 2019.