Policy Optimization with Advantage Decomposition

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Policy gradient (PG) on high-dimensional control tasks exhibit its difficulty because of the large variance of the PG estimators. Recent studies show that the Rao-Blackwell theorem (RB) can efficiently reduce the variance of PG by decomposing the action space. However, to invoke RB, previous studies either make strong assumptions or heuristically use the Hessian matrix of the advantage function which is not theoretically sound and also requires specific network architectures to obtain. Inspired by those works, we present algorithmic approaches via pairwise testing and submodular optimization to decompose the action space using the information from the advantage function. Our methods impose weaker assumptions than both previous proposals and are the first ones to provide rigorous performance guarantees. Empirical studies demonstrate the performance improvements on high-dimensional synthetic settings and OpenAI Gym's MuJoCo continuous control tasks.

1 Introduction

Policy gradient (PG) methods [19] have been widely applied to various challenging problems including video games [12], robotics [8], and continuous control tasks [16, 10]. A major challenge of PG is the high variance of the gradient estimator. Since its inception, the variance of the PG estimator has been continuously improved [19, 17, 12, 6, 11, 4, 18]. Especially, when dealing with high-dimensional action spaces, Rao-Blackwell theorem (RB) [2] based methods [20, 9] demonstrate significant efficiency. To use RB on the PG estimator, the algorithm is required to find the underlying dependency structure of the action space. The structure information is important to keep the unbiasedness of the estimator while finding the estimator with the optimal variance.

Several studies have been proposed. [20] assumes that the policy distribution is fully factorized and decomposes the PG estimator into as many components as the dimension of the action. The action-dependent estimator (ADFB) proposed in [20] also uses averaged action value as the baseline function. However, the fully factorized assumption is relatively strong in practice. Also, there's an inconsistency: the demonstrated algorithm uses the action value of averaged action as the baseline instead of the averaged action value proposed in its theory. [9] proposes policy optimization with second-order advantage information (POSA) which leverages the block diagonal structure of the advantage Hessian matrix. It successfully finds the non-trivial action space structure, but may suffer from the following issues. First, POSA focuses on the elements of Hessian. Though the Hessian elements corresponding to two independent action dimensions are supposed to be zero, the converse does not hold. Hence, finding the block diagonal structure then does not guarantee the correct partition of the action space. Second, POSA finds the independent partition within the action space using Hessian which indicates pairwise dependency. That is also a necessary condition but not a sufficient one. A simple counter-example would be having three boolean variables that sum up to one: the variables are pairwise independent, but mutually dependent. These inherent weaknesses motivate the rigorous study of the application of RB on PG.

We present policy optimization with advantage decomposition (POAD). Our proposed algorithms make the assumption that the advantage function can be decomposed into multiple components which is weaker than both the assumptions made in POSA and in ADFB. The assumption yields exactly the same estimator which was demonstrated but not explained in ADFB but not explained. Our assumption is also purely based on value learning which allows the algorithm to make the best approximation and leverages the value function information for the policy learning. Finding the decomposition of the action-value function also addresses the flaws in POSA. As the algorithm no longer relies on the pairwise dependencies of the dimensions of the action space, the results are theoretically sound.

We propose two algorithms for computing a decomposition. Both are based on a novel "dependence score" that can be applied to any two disjoint subsets of variables. This score is described by a simple formula that can be estimated to within arbitrary accuracy by random sampling. Both the approaches are efficient (in sublinear time) and have theoretically sound performance guarantees. Our Pairwise Estimates (PE) algorithm computes dependence score estimates for every pair among n variables that correspond to n dimensions of the action space. The estimates are used to build a weighted graph where each node is a variable and the weight of each edge is the corresponding score. The edges are keeping removed in decreasing weight order until only the desired number k of connected components is left. We prove that the output partition is a factor-$O(kn^2)$ approximation of the best possible one. Our Submodular Minimization (SM) algorithm achieves a significantly better almost factor-4 approximation guarantee. The main insight behind this algorithm is that the dependence score is a submodular function of the bipartition. We then reduce the problem to computing the optimal partition, which could be solved efficiently by existing studies of submodular minimization algorithms [5, 14, 7].

We evaluate both our algorithms on a variety of reinforcement learning tasks, including the high-dimension environments from OpenAI Gym's MuJoCo continuous control tasks. POAD outperforms the existing approaches with both the PE algorithm and the SM algorithm. Especially, POAD with SM has the best overall performance and the best time complexity. Also, POAD with our decomposition algorithms is more stable than POSA's [9] which uses the Hessian matrix to partition the action space.

2 Preliminaries

2.1 Policy Optimization

We consider policy optimization in the discrete-time Markov decision process (MDP) setting, denoted as the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, r, \rho_0, \gamma)$. That includes $\mathcal{S} \in \mathbb{R}^m$ is the m dimensional state space, $\mathcal{A} \in \mathbb{R}^n$ is the n dimensional action space, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}^+$ is the environment transition probability function, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, ρ_0 is the initial state distribution and $\gamma \in (0, 1]$ is the unnormalized discount factor. Policy optimization learns a stochastic policy $\pi_\theta : \mathcal{S} \times \mathcal{A} \to \mathbb{R}_+$, which is parameterized by θ, to maximize the expected cumulative reward

$$J(\theta) = \mathbb{E}_{s\sim\rho_\pi, a\sim\pi}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)],$$

where $\rho_\pi(s) = \sum_{t=1}^{\infty} \gamma^{t-1}\mathbb{P}(s_t = s)$ is the discounted state visitation distribution. Define the value function

$$V^\pi(s_t) = \mathbb{E}_\pi[\sum_{t' \geq t}^{\infty} \gamma^{t'-t} r(s_{t'}, a_{t'})|s_t, \pi]$$

to be the expected return of policy π at state s_t. Notices that the state-action function as

$$Q^\pi(s_t, a_t) = \mathbb{E}_\pi[\sum_{t' \geq t}^{\infty} \gamma^{t'-t} r(s_{t'}, a_{t'})|s_t, a_t, \pi]$$

to be the expected return by policy π after taking the action a_t at the state s_t. We make use of $\hat{Q}^\pi(s_t, a_t)$ and $\hat{V}^\pi(s_t)$ to denote the empirical function approximator of $Q^\pi(s_t, a_t)$ and $V^\pi(s_t)$, respectively. Define the advantage function to be the gap between the value function and the action-value, as $A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t)$. For simplicity, we rewrite it as the time-independent formulation $J(\theta) = \mathbb{E}_{\pi,\rho_\pi}[r(s, a)]$.

Subsequently, we give the formulation of Action Dependent Factorization Baselines (ADFB) estimator [20],

$$\nabla_\theta J(\theta)_{ADFG} = \sum_{i=1}^{n} \mathbb{E}_{\pi(a_i)}[\nabla_\theta \log \pi(a_i|s)\{Q^\pi(s,a) - Q(s,\bar{a_i},a_{-i})\}],$$

where a_i is i-th action dimension and denote $a_{-i} = a_{[n]/\{i\}}$, and Action Subspace Dependent Gradient (ASDG) estimator [9],

$$\begin{aligned}\nabla_\theta J(\theta)_{ASDG} = & \sum_{j=1}^{k} \mathbb{E}_{\pi(a_{(j)}|s)}[\nabla_\theta \log \pi(a_{(j)}|s)(A^\pi(s,a_{(j)}) - c(s,(a_{(j)},\tilde{a}_{(-j)}))) \\ & - \nabla_\theta f_j(\theta,s,\xi)\nabla_{a_{(j)}} c_j(s,a_{(j)})],\end{aligned}$$

where the reparametrization term $\nabla_\theta f(\theta,s,\xi) \in \mathbb{R}^{N_\theta \times n}$ is divided into k parts as $\nabla_\theta f = [\nabla_\theta f_1, ..., \nabla_\theta f_k]$ and N_θ is the dimension of θ. Note that k is the number of partitions of the action space. Similarly, $a_{(j)}$ is j-th action dimension subset and $a_{(-j)}$ is the complementary action dimension set. Specifically, $\tilde{a}_{(-j)}$ denotes that $a_{(-j)}$ is regarded as a constant in the function. Also note that ADFG is the special case of ASDG.

An important observation is that ASDG is unbiased and with lower variance under the assumption that the advantage is decomposed by the action spaces. We formulate the assumption as below with an error term, and thereafter focus on minimizing the error term in our algorithm.

Assumption 1 (Advantage Decomposition Assumption)**.** *Assume that the advantage function $A^\pi(s,a)$ can be locally approximated by decomposition function with respect to a, that is,*

$$A^\pi(s,a) = A_1^\pi(s,a_{(1)}) + \cdots + A_k^\pi(s,a_{(k)}) + D(s,a),$$

where $D(s,a)$ is the approximation error.

2.2 Direct Sum Decomposition

As pointed out in the Assumption 1, we consider the general family $F(\boldsymbol{V})$ as a function of a product set. A direct sum decomposition of F is a partition $(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_k)$ of the set of variables $\boldsymbol{V}$ such that

$$F(\boldsymbol{V}) = F_1(\boldsymbol{X}_1) + \cdots + F_k(\boldsymbol{X}_k) \tag{1}$$

for some functions $F_1, \ldots, F_k$. We are interested in approximate sublinear-time algorithms for computing direct sum decompositions given oracle access to F.

Given a partition $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_k$ of the variables, the quality of the decomposition with respect to this partition is measured by

$$\delta^p(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_k) = \min_{F_1,\ldots,F_k} \|F(X_1, \ldots, X_k) - F_1(X_1) - \cdots - F_k(X_k)\|_p$$

where $\|\cdot\|_p$ is the ℓ^p-norm over the product measure $\|F\| = \mathbb{E}\big[|F(X_1, \ldots, X_k)|^p\big]^{1/p}$. In Sections 3 and 4 the norm index p can be any integer at least 1. In Section 5 $|\cdot|$ is absolute value, and p equals 2. When p is clear from context we omit it from the notation.

The objective is to find an approximation of the best-possible partition, which minimizes the objective

$$\delta_2^p(F) = \min \delta^p(\boldsymbol{X}, \overline{\boldsymbol{X}})$$

where the minimum is taken over all bipartitions $(\boldsymbol{X}, \overline{\boldsymbol{X}})$ of $\boldsymbol{V}$ and $\overline{\boldsymbol{X}}$ denotes the complement of $\boldsymbol{X}$. More generally, we consider partitions into (at least) k nonempty sets, giving rise to the objective value $\delta_k^p(F) = \min \delta^p(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_k)$.

Our algorithms are based on the following dependence estimator inspired by the rank-1 test of [3]. Let $\boldsymbol{X}$ and $\boldsymbol{Y}$ be two disjoint sets of variables. The dependence estimator $D_F(\boldsymbol{X}, \boldsymbol{Y})$ is the random variable

$$D_F = F(X,Y,Z) + F(X',Y',Z) - F(X',Y,Z) - F(X,Y',Z)$$

where X, X' are independent samples of the $\boldsymbol{X}$ variable, Y, Y' are independent samples of the $\boldsymbol{Y}$ variable, and Z is a random sample of the remaining variables.

If F decomposes into a direct sum that partitions the $\boldsymbol{X}$ and $\boldsymbol{Y}$ variables then D_F equals zero. Conversely, $\|D_F\|_p$ measures the quality of the approximation in the ℓ^p-norm.

In the analysis it will be convenient to use the notation $F \approx_\delta G$ for $\|F - G\|_p \leq \delta$. The following two facts are immediate

Fixing: If $F(X, Z) \approx_\delta G(X, Z)$ then $F(\underline{X}(Z), Z) \approx_\delta G(\underline{X}(Z), Z)$ for some function $\underline{X}(Z)$.

Triangle inequality: If $F \approx_\delta G$ and $G \approx_{\delta'} H$ then $F \approx_{\delta+\delta'} H$.

2.3 Submodularity Optimization

In preparation towards our algorithms in Section 5, we recall the definition of a submodular function and its Lovász extension.

Definition 2 (Submodular set function)**.** *If $\boldsymbol{V}$ is a finite set, a submodular function is a set function $D : 2^{\boldsymbol{V}} \to \mathbb{R}$, where $2^{\boldsymbol{V}}$ denotes the power set of $\boldsymbol{V}$. For every $S, T \subseteq \boldsymbol{V}$, we have*

$$D(S) + D(T) \geq D(S \cup T) + D(S \cap T).$$

Definition 3 (Lovász extension)**.** *Consider any vector $x = \{x_1, \cdots, x_n\}$ such that each $0 \leq x_i \leq 1$. Then the Lovász extension is defined as $D^L = \mathbb{E}(D(\{i | x_i \geq \lambda\}))$ where the expectation is over λ chosen from the uniform distribution on the interval* $[0, 1]$.

The Lovász extension is a convex function if and only if the function D is submodular. Therefore, it reduces submodular minimization of a discrete function to convex minimization of a continuous one. Several algorithms that rely on this connection have been proposed [5, 14, 7]; see [1] for a survey.

3 Estimating the Quality of a Bipartition

In this Section we show that $\|D_F(\boldsymbol{X}, \boldsymbol{Y})\|_p$ is an approximate estimator for the quality $\delta^p(\boldsymbol{X}, \boldsymbol{Y})$ of a decomposition, namely

$$\delta^p(\boldsymbol{X}, \boldsymbol{Y}) \leq \|D_F(\boldsymbol{X}, \boldsymbol{Y})\|_p \leq 4 \cdot \delta^p(\boldsymbol{X}, \boldsymbol{Y}). \tag{2}$$

We then state standard bounds on the quality of the sampling approximators for this value. Taken together, we obtain efficient factor $(4 + \varepsilon)$-approximation algorithms for $\delta^p(\boldsymbol{X}, \boldsymbol{Y})$ of a given partition:

Theorem 4. *There is an algorithm that given a bipartition $\boldsymbol{X}, \boldsymbol{Y}$ of the variables and a parameter $\varepsilon > 0$, outputs a value between $\delta^p(\boldsymbol{X}, \boldsymbol{Y})$ and $4\delta^p(\boldsymbol{X}, \boldsymbol{Y}) + \varepsilon$ with probability at least $1 - \gamma$ from*

$$\min\big\{K^p \|F\|_{2p}^{2p} / \gamma\varepsilon^{2p}, K^p \|F\|_\infty^{2p} \log(1/\gamma)/\varepsilon^{2p}\big\} \tag{3}$$

queries to F in time linear in the number of queries, where $K > 0$ is a constant.

The first bound has a linear dependence on $1/\gamma$ assuming merely that F is bounded in the $2p$-norm. The second bound has a much better, logarithmic dependence on $1/\gamma$ but requires F to be uniformly bounded. Intermediate bounds should also be possible but for simplicity we only provide the two extremes.

The analysis in fact applies to any pair of disjoint subsets $\boldsymbol{X}, \boldsymbol{Y}$ that do not necessarily partition all the variables. In this more general setting (which will be useful in Section 4) distance is measured by the formula

$$\delta^p(\boldsymbol{X}, \boldsymbol{Y}) = \min_{A,B} \|F(X, Y, Z) - A(X, Z) - B(Y, Z)\|_p.$$

Claim 5 (Completeness of D_F)**.** *For all disjoint $\boldsymbol{X}$, $\boldsymbol{Y}$, $\|D_F(\boldsymbol{X}, \boldsymbol{Y})\|_p \leq 4 \cdot \delta^p(\boldsymbol{X}, \boldsymbol{Y})$.*

Proof. By definition of $\delta(\boldsymbol{X}, \boldsymbol{Y})$ there exists a decomposition of the form

$$F(V) = A(X, Z) + B(Y, Z) + D(X, Y, Z),$$

where $\|D(X, Y, Z)\| = \delta(\boldsymbol{X}, \boldsymbol{Y})$. In the expansion of D_F all the A and B terms cancel out, leaving

$$\begin{aligned} \|D_F(\boldsymbol{X}, \boldsymbol{Y})\| &= \|D(X, Y, Z) + D(X', Y', Z) - D(X, Y', Z) - D(X', Y, Z)\| \\ &\leq \|D(X, Y, Z)\| + \|D(X', Y', Z)\| + \|D(X, Y', Z)\| + \|D(X', Y, Z)\| \\ &= 4\delta(\boldsymbol{X}, \boldsymbol{Y}). \end{aligned}$$

□

Claim 6 (Soundness of D_F). *For all disjoint $\boldsymbol{X}, \boldsymbol{Y}$, $\delta^p(\boldsymbol{X}, \boldsymbol{Y}) \leq \|D_F(\boldsymbol{X}, \boldsymbol{Y})\|_p$* [1]

Proof. Let $\varepsilon = \|D_F(\boldsymbol{X}, \boldsymbol{Y})\|$. Then

$$F(X, Y, Z) \approx_\varepsilon F(X, Y', Z) - F(X', Y, Z) - F(X', Y', Z).$$

By fixing, there exist functions $x' = \underline{X}'(Z)$ and $y' = \underline{Y}'(Z)$ for which

$$F(X, Y, Z) \approx_\varepsilon F(\underline{X}'(Z), \underline{Y}'(Z), Z) - F(X, \underline{Y}'(Z), Z) - F(\underline{X}'(Z), Y, Z).$$

Therefore $F(X, Y, Z) \approx_e A(X, Z) + B(Y, Z)$, where $A(X, Z) = F(X, \underline{Y}'(Z), Z)$ and $B(Y, Z) = F(\underline{X}'(Z), y, Z) - F(\underline{X}'(Z), \underline{Y}'(Z), Z)$. □

To finish the proof of Theorem 4 we need the following (standard) sampling bounds derived from Chebyshev's and Chernoff's inequalities.

Claim 7. *The empirical average of $|f(X)|^p$ over N random samples X is within ε^p of $\|f\|_p^p$ except with probability $\min\{\|f\|_{2p}^{2p}/N\varepsilon^{2p}, 2\exp(-N\varepsilon^{2p}/2\|f\|_\infty^{2p})\}$.*

Claim 8. *Assume $t, \hat{t} \geq 0$. If $t^p \leq \hat{t}^p \leq t^p + (\varepsilon/2)^p$ then $t \leq \hat{t} \leq t + \varepsilon$.*

Proof. The left-hand inequalities are immediate. For the right-hand ones we start we consider two cases. If $t \leq \varepsilon/2$, then $\hat{t}^p \leq 2(\varepsilon/2)^p \leq \varepsilon \leq t + \varepsilon$. If $t > \varepsilon/2$ then

$$\hat{t} - t \leq \frac{\hat{t}^p - t^p}{t^{p-1}} \leq \frac{(\varepsilon/2)^p}{(\varepsilon/2)^{p-1}} \leq \varepsilon.$$ □

Claim 9. *The value $\|D_F(\boldsymbol{X}, \boldsymbol{Y})\|_p$ can be estimated within ε from* (3) *queries to F in linear time with probability $1 - \gamma$.*

Proof. By the triangle inequality, $\|D_F(\boldsymbol{X}, \boldsymbol{Y})\|_p \leq 4\|F\|_p$. By convexity $\|D_F(\boldsymbol{X}, \boldsymbol{Y})\|_p^p \leq 4^{p-1}\|F\|_p^p$. By Claim 7, an $(1 - \gamma)$-probability estimate that lies between $\|D_F(\boldsymbol{X}, \boldsymbol{Y})\|_p^p$ and $\|D_F(\boldsymbol{X}, \boldsymbol{Y})\|_p^p + (\varepsilon/2)^p$ can be computed by taking the empirical average of (3) samples of D_F. Each sample of D_F can be obtained using four queries to F. The quality of the estimate then follows from Claim 8. □

Theorem 4 now follows from Claim 9 and inequality (2).

4 Approximate Partitioning via Pairwise Estimates

We analyze the following algorithm for computing an approximate decomposition.

Algorithm 1: Direct sum decomposition algorithm

```
Input: number of partitions $k$.
Output: partition $\mathcal{P}$.
for each iteration do
    (Step 1) For every pair of distinct variables $\boldsymbol{x}, \boldsymbol{y}$, calculate an estimate $\hat{e}(\boldsymbol{x}, \boldsymbol{y})$ for
      $e(\boldsymbol{x}, \boldsymbol{y}) = \|D_F(\{\boldsymbol{x}\}, \{\boldsymbol{y}\})\|_p$;
    (Step 2) Create a weighted graph with vertices $\boldsymbol{V}$ and weights $\hat{e}(\boldsymbol{x}, \boldsymbol{y})$. Order the edges in
      increasing weight and keep removing them until the graph partitions into $k$ components.[2];
end
```

Theorem 10. *Assuming $e(\boldsymbol{x}, \boldsymbol{y}) \leq \hat{e}(\boldsymbol{x}, \boldsymbol{y}) \leq e(\boldsymbol{x}, \boldsymbol{y}) + \varepsilon$ for all $\boldsymbol{x}$ and $\boldsymbol{y}$,*

$$\delta^p(\mathcal{P}) \leq (8k - 10)n^2(4\delta_k^p(F) + \varepsilon). \tag{4}$$

Corollary 11. *If Step 1 of the algorithm is implemented by empirically averaging over*

$$\min\{K^p\|F\|_{2p}^{2p}n^2/\gamma\varepsilon^{2p}, K^p\|F\|_\infty^{2p}\log(n/\gamma)/\varepsilon^{2p}\}$$

random samples then approximation guarantee (4) *holds with probability at least $1 - \gamma$. Note that K is an absolute constant.*

[1]In the case of Boolean functions under uniform measure this was proved by David et al. [3].

5 Approximate Partitioning via Submodular Function Minimization

Let $f(\boldsymbol{X}) = \|D_F(\boldsymbol{X}, \overline{\boldsymbol{X}})\|_2^2 = \mathbb{E}[D_F(\boldsymbol{X}, \overline{\boldsymbol{X}})^2]$. The main result of this section is

Theorem 12. *f is a symmetric submodular function.*

By (2), $\delta_2^2(F) \le \min \|D_F(\boldsymbol{X}, \overline{\boldsymbol{X}})\|_2 \le 4\delta_2^2(F)$, where the minimum is taken over all nontrivial bipartitions. Since submodular functions can be minimized efficiently [5, 13, 14, 7], by Theorem 12 it is possible to efficiently compute a factor-4 approximation to $\delta_2^2(F)$ *given oracle access to f.*

Corollary 13. *There is an algorithm that, given oracle access to F, runs in time polynomial in n, $1/\varepsilon$, and $\|F\|_4/\gamma$ or $\|F\|_\infty \log(1/\gamma)$, and outputs a set $\boldsymbol{X}$ such that $\delta_2(F) \le f(\boldsymbol{X}) \le 4\delta_2(F) + \varepsilon$ with probability at least $1 - \gamma$.*

In the remaining part of this section we prove Theorem 12.

Claim 14. *For every partition $(\boldsymbol{X}, \boldsymbol{Y})$, $\mathbb{E}[D_F(\boldsymbol{X}, \boldsymbol{Y})^2] = 4 \cdot \mathbb{E}[F(X, Y) \cdot D_F(X, Y, X', Y')]$.*

Proof. Let X_0, X_1 be independent copies of $\boldsymbol{X}$ and Y_0, Y_1 be independent copies of $\boldsymbol{Y}$. We can express D_F in the form

$$D_F = \sum_{i,j \in \{0,1\}} (-1)^{i+j} F(X_i, Y_j).$$

Then

$$\begin{aligned}
\mathbb{E}[D_F^2] &= \mathbb{E} \sum_{i,j \in \{0,1\}} (-1)^{i+j} F(X_i, Y_j) \cdot \sum_{i',j' \in \{0,1\}} (-1)^{i'+j'} F(X_{i'}, Y_{j'}) \\
&= \mathbb{E} \sum_{i,j,i',j' \in \{0,1\}} (-1)^{i+j+i'+j'} F(X_i, Y_j) \cdot F(X_{i'}, Y_{j'}) \\
&= \sum_{i,i',j,j' \in \{0,1\}} (-1)^{(i \oplus i') + (j \oplus j')}\, \mathbb{E}\big[F(X_i, Y_j) \cdot F(X_{i'}, Y_{j'})\big],
\end{aligned}$$

where $\oplus$ is addition modulo 2. By symmetry of the variables

$$\mathbb{E}\big[F(X_i, Y_j) \cdot F(X_{i'}, Y_{j'})\big] = \mathbb{E}\big[F(X_0, Y_0) \cdot F(X_{i \oplus i'}, Y_{j \oplus j'})\big]$$

substituting in the previous expression and changing the order of the expectation and the summation we obtain

$$\mathbb{E}[D_F^2] = \mathbb{E}\, F(X_0, Y_0) \sum_{i,i',j,j' \in \{0,1\}} (-1)^{(i \oplus i') + (j \oplus j')} F(X_{i \oplus i'}, Y_{j \oplus j'}).$$

The summation is exactly $4D_F(X_0, Y_0, X_1, Y_1)$ as desired. □

Let $D_F(\boldsymbol{X}, \boldsymbol{Y} \mid Z)$ denote $D_{F_Z}(\boldsymbol{X}, \boldsymbol{Y})$, where F_Z is obtained from F by fixing $\boldsymbol{Z}$ to Z.

Claim 15. *$D_F(\boldsymbol{XY}, \boldsymbol{ZW}) + D_F(\boldsymbol{XZ}, \boldsymbol{YW}) - D_F(\boldsymbol{XYZ}, \boldsymbol{W}) - D_F(\boldsymbol{X}, \boldsymbol{YZW})$ is equal to $D_F(\boldsymbol{Y}, \boldsymbol{Z} \mid XW') + D_F(\boldsymbol{Y}, \boldsymbol{Z} \mid X'W)$.*

Proof. Expanding the left-hand side yields sixteen terms of the form $F(X^? Y^? Z^? W^?)$ with a leading plus or minus sign, where the question mark indicates a primed or unprimed superscript. Each of the terms $F(XYZW)$ and $F(X'Y'Z'W')$ occurs exactly twice with a plus sign and twice with a minus sign, so those eight terms cancel out. There remain four terms of the type $F(XY^?Z^?W')$ and four of the type $F(X'Y^?Z^?W)$. Among those, the terms where exactly one of Y, Z is primed are negative and the others are positive. Therefore the type $F(XY^?Z^?W')$ terms add up to $D_F(\boldsymbol{Y}, \boldsymbol{Z} \mid XW')$ and the type $F(X'Y^?Z^?W)$ terms add up to $D_F(\boldsymbol{Y}, \boldsymbol{Z} \mid X'W)$. □

Claim 16. *For independent Y, Z, $\mathbb{E}_Y \operatorname{Var}_Z F(Y, Z) \ge \operatorname{Var}_Z \mathbb{E}_Y F(Y, Z)$.*

Proof. Set $F' = F - \mathbb{E}_Z[F]$. Then

$$\mathbb{E}_Y \operatorname{Var}_Z F = \mathbb{E}_Y \mathbb{E}_Z[F'^2] = \mathbb{E}_Z[\mathbb{E}_Y[F'^2]],$$

while

$$\operatorname{Var}_Z \mathbb{E}_Y F = \mathbb{E}_Z[(\mathbb{E}_Y F - \mathbb{E}_Z \mathbb{E}_Y F)^2] = \mathbb{E}_Z[\mathbb{E}_Y[F']^2].$$

The fact follows from the Cauchy-Schwarz inequality. □

Claim 17. $\mathbb{E}[F(XYZW)D_F(YZY'Z' \mid X'W)]$ *is nonnegative.*

Proof. Without loss of generality we may assume $\boldsymbol{W}$ is empty. Then the expression is the difference of the terms

$$ev = \mathbb{E}[F(XYZ)F(X'YZ)] - \mathbb{E}[F(XYZ)F(X'YZ')] \quad \text{and}$$
$$ve = \mathbb{E}[F(XYZ)F(X'Y'Z)] - \mathbb{E}[F(XYZ)F(X'Y'Z')].$$

Let $\hat{F}(\boldsymbol{YZ}) = \mathbb{E}_X[F(X\boldsymbol{YZ})]$. Then

$$ev = \mathbb{E}_{YZ}[\hat{F}^2] - \mathbb{E}_Y[\mathbb{E}_Z[\hat{F}]^2] = \mathbb{E}_Y \operatorname{Var}_Z[\hat{F}].$$

and

$$ve = \mathbb{E}_Z[\mathbb{E}_Y[\hat{F}]^2] - \mathbb{E}_{ZY}[\hat{F}]^2 = \operatorname{Var}_Z \mathbb{E}_Y[\hat{F}].$$

The claim now follows from Fact 16. □

Proof of Lemma 12. Proving submodularity of f amounts to verifying the inequality

$$\mathbb{E}[D_F(\boldsymbol{XY}, \boldsymbol{ZW})^2] + \mathbb{E}[D_F(\boldsymbol{XZ}, \boldsymbol{YW})^2] - \mathbb{E}[D_F(\boldsymbol{XYZ}, \boldsymbol{W})^2] - \mathbb{E}[D_F(\boldsymbol{X}, \boldsymbol{YZW})^2] \geq 0.$$

for every partition $(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}, \boldsymbol{W})$ of the variables. This follows by applying Claims 14, 15, and 17 to each of the two expressions in order. □

6 Experiments

6.1 Implementation Details

Our setting is similar to the one used in [9][3]. Our algorithm uses PPO [16] together with GAE [15]. We use three neural networks including a policy network for PPO and a value network for GAE. Both of the networks have the same architecture as is in the original A2C and PPO work [12, 16]. We utilize a third network which estimates the advantage function $A^\mu(s, a)$. Unlike [9] where $\nabla_{aa} A^\mu(s, a)$ has to be computed efficiently, our advantage network does not require any special neural network architecture to achieve that. In our implementation, $A^\mu(s, a)$ first encodes the state s with a hidden layer and concatenates it with the action a. It then feeds the output into a two-layer MLPs. Other parameters are consistent with those in [9].

Pairwise Estimation Algorithm Implementation-wise, we use an alternative version of the algorithm which shares the same theoretical guarantee but empirically runs faster. Instead of adding all the edges at the beginning and remove one at a step, we keep adding edges in decreasing weight until just before the connected component count drops below k. During the pairwise estimation, we collect the batch data and sample a second action for each of the state (apart from the action that is interacting with the environment and generating the trajectories). The second action is used as the testing term X' and Y'. We have the pairwise dependence scores (weights of the edge) averaged within each of the batches.

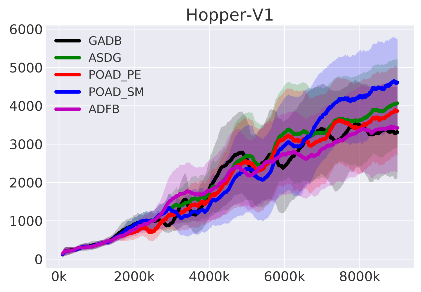
Submodular Minimization Algorithm While the [5] algorithm is considered impractical, we implement the submodular minimization with enumerating the possible partitions. We do not know if any other efficient submodular minimization algorithm (see [1, Chapter 10] for a survey) provides similar optimality guarantees when provided with an approximate oracle as is in our settings and we plan to investigate this in the future.

Synthetic High-Dimensional Action Spaces Similar to previous works, we design a synthetic environment with a one-step MDP where the reward is quadratic with respect to a and the quadratic term is block-diagonal. As the environment setting satisfies Assumption 1, it achieves almost-optimal performance which is similar to POSA.

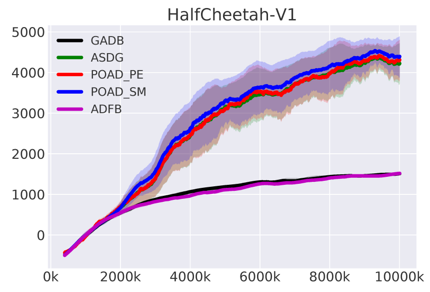
6.2 OpenAI Gym's MuJoCo Environments

Fig. 1 shows the results on several of the high-dimensional continuous control tasks from OpenAI Gym's MuJoCo Environments. The legend POAD_PE and POAD_SM stand for our proposed

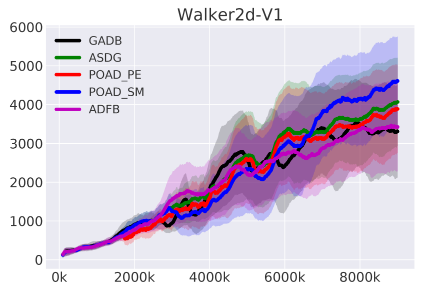
[3]We will publish our code upon the acceptance of this paper.



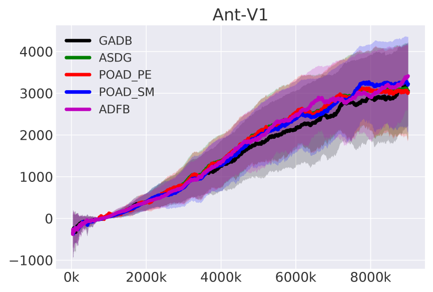
(a) Hopper



(b) HalfCheetah



(c) Walker



(d) Ant

Figure 1: Comparison between three baselines (ADFB, GADB, ASDG) and our two direct sum testing algorithms (POAD_PE and POAD_SM) on various OpenAI Gym Mujoco continuous control tasks, including Hopper-V1, HalfCheetah-V1 Walker2d-V1 and Ant-V1 . Notice that POAD_PE is the pairwise estimation algorithm and POAD_SM is the submodular Minimization.

algorithms. POAD_SM has the best overall performance which is consistent across various of the settings. Our algorithm based on pairwise estimates has similar performance compared with previous Hessian-based algorithm POSA.

We observe that RB works exceptionally well on HalfCheetah, which coincides with the fact that the approximation error is relatively low in such environment. As an intuition, the policy optimization algorithm has to balance accuracy (i.e., the satisfaction of its model assumption) and efficiency (i.e., the variance reduction performance) in practice. The rigorous analysis of the approximation error in our study helps the understanding of the policy optimization algorithms and their assumptions.

References

[1] Francis Bach. Learning with submodular functions: A convex optimization perspective. *Foundations and Trends in Machine Learning*, 6(2-3):145–373, 2013.

[2] George Casella and Christian P Robert. Rao-blackwellisation of sampling schemes. *Biometrika*, 83(1):81–94, 1996.

[3] Roee David, Irit Dinur, Elazar Goldenberg, Guy Kindler, and Igor Shinkar. Direct sum testing. *SIAM Journal on Computing*, 46(4):1336–1369, 2017.

[4] Will Grathwohl, Dami Choi, Yuhuai Wu, Geoff Roeder, and David Duvenaud. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. In *International Conference on Learning Representations*, 2018.

[5] Martin Grötschel, László Lovász, and Alexander Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.

[6] Shixiang Gu, Timothy Lillicrap, Zoubin Ghahramani, Richard E Turner, and Sergey Levine. Q-prop: Sample-efficient policy gradient with an off-policy critic. *arXiv preprint arXiv:1611.02247*, 2016.

[7] Satoru Iwata, Lisa Fleischer, and Satoru Fujishige. A combinatorial strongly polynomial algorithm for minimizing submodular functions. *J. ACM*, 48(4):761–777, July 2001.

[8] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

[9] Jiajin Li and Baoxiang Wang. Policy optimization with second-order advantage information. *arXiv preprint arXiv:1805.03586*, 2018.

[10] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[11] Hao Liu, Yihao Feng, Yi Mao, Dengyong Zhou, Jian Peng, and Qiang Liu. Action-dependent control variates for policy optimization via stein identity. In *International Conference on Learning Representations*, 2018.

[12] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy P Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, 2016.

[13] Maurice Queyranne. Minimizing symmetric submodular functions. *Mathematical Programming*, 82(1-2):3–12, 1998.

[14] Alexander Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *J. Comb. Theory, Ser. B*, 80(2):346–355, 2000.

[15] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[16] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[17] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

[18] George Tucker, Surya Bhupatiraju, Shixiang Gu, Richard E Turner, Zoubin Ghahramani, and Sergey Levine. The mirage of action-dependent baselines in reinforcement learning. *arXiv preprint arXiv:1802.10031*, 2018.

[19] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[20] Cathy Wu, Aravind Rajeswaran, Yan Duan, Vikash Kumar, Alexandre M.Bayen, Sham Kakade, Igor Mordatch, and Pieter Abbeel. Variance reduction for policy gradient with action-dependent factorized baselines. In *International Conference on Learning Representations*, 2018.

A Proofs of the Pairwise Estimates Algorithm

For a partition $\mathcal{P}$ of the variables, let $\Delta(\mathcal{P}) = \sum \delta(\{\boldsymbol{x}\}, \{\boldsymbol{y}\})$, where the sum is taken over all pairs that cross the partition. We will deduce Theorem 10 from the following bound on $\delta(\mathcal{P})$

Claim 18. *For every k-partition $\mathcal{P}$, $\delta(\mathcal{P}) \le (16k-20)\Delta(\mathcal{P})$.*

The following fact is immediate from the definitions of δ.

Fact 19. *For any partition $(\boldsymbol{U}, \overline{\boldsymbol{U}})$ such that $\boldsymbol{X} \subseteq \boldsymbol{U}$ and $\boldsymbol{Y} \subseteq \overline{\boldsymbol{U}}$, $\delta(\boldsymbol{X}, \boldsymbol{Y}) \le \delta(\boldsymbol{U}, \overline{\boldsymbol{U}})$.*

Proof of Theorem 10. By Claim 5 and Fact 19, all edges $(\boldsymbol{x}, \boldsymbol{y})$ in the optimal partition must satisfy $e(\boldsymbol{x}, \boldsymbol{y}) \le 4\delta_2(F)$. By our assumption on the quality of the approximations,

$$\hat{e}(\boldsymbol{x}, \boldsymbol{y}) \le 4\delta_2(F) + \varepsilon. \tag{5}$$

Since the algorithm removes edges in increasing order of weight, all the edges that cross the output partition $\mathcal{P}$ must also satisfy this inequality. Then

$$\begin{aligned}
\delta(\mathcal{P}) &\le (16k-20)\Delta(\mathcal{P}) && \text{by Claim 18,}\\
&\le (16k-20)\sum\nolimits_{\boldsymbol{x},\,\boldsymbol{y}\text{ cross }\mathcal{P}} e(\boldsymbol{x}, \boldsymbol{y}) && \text{by Claim 6,}\\
&\le (16k-20)\sum\nolimits_{\boldsymbol{x},\,\boldsymbol{y}\text{ cross }\mathcal{P}} \hat{e}(\boldsymbol{x}, \boldsymbol{y}) && \text{by quality of approximation,}\\
&\le (16k-20)\sum\nolimits_{\boldsymbol{x},\,\boldsymbol{y}\text{ cross }\mathcal{P}} 4\delta_2(F) + \varepsilon && \text{by (5),}\\
&\le (8k-10)n^2 \cdot (4\delta_2(F) + \varepsilon).
\end{aligned}$$

The last inequality holds because there are at most $\binom{n}{2} \le n^2/2$ pairs of variables crossing the partition. □

In the case $k = 2$, the leading constant $8k - 10 = 6$ can be improved to 1 by using Claim 21 instead of Claim 18 and the fact that at most $n^2/4$ pairs cross the partition.

It remains to prove Claim 18. We use $\boldsymbol{X}\boldsymbol{X}'$ to denote the union of the variable sets $\boldsymbol{X}$ and $\boldsymbol{X}'$.

Claim 20. *For disjoint sets of variables $\boldsymbol{X}, \boldsymbol{X}', \boldsymbol{Y}$, $\delta(\boldsymbol{X}\boldsymbol{X}', \boldsymbol{Y}) \le \delta(\boldsymbol{X}, \boldsymbol{Y}) + 2\delta(\boldsymbol{X}', \boldsymbol{Y})$.*

Proof of Claim 20. For simplicity of notation we omit the dependences on Z. Assume that

$$\begin{aligned}
F(X, X', Y) &\approx_\delta A(X, X') + B(X', Y) \qquad \text{and}\\
F(X, X', Y) &\approx_{\delta'} A'(X, X') + B'(X, Y).
\end{aligned}$$

By the triangle inequality,

$$A(X, X') + B(X', Y) \approx_{\delta+\delta'} A'(X, X') + B'(X, Y).$$

Fix $X'(Z) = \underline{X}'(Z)$. Writing $C(X) = A(X, \underline{X}') - A'(X, \underline{X}')$ and $D(Y') = B(\underline{X}', Y')$ we get that

$$B'(X, Y) \approx_{\delta+\delta'} C(X) + D(Y).$$

By the triangle inequality (with the second equation), we get that

$$F(X, X', Y) \approx_{\delta+2\delta'} A'(X, X') + C(X) + D(Y). \qquad □$$

Claim 21. *For every bipartition $\boldsymbol{X}, \overline{\boldsymbol{X}}$ of the variables, $\delta(\boldsymbol{X}, \overline{\boldsymbol{X}}) \le 4 \cdot \Delta(\boldsymbol{X}, \overline{\boldsymbol{X}})$.*

Proof. By Claim 20,

$$\delta(\boldsymbol{X}'\{\boldsymbol{x}\}, \{\boldsymbol{y}\}) \le \delta(\boldsymbol{X}', \{\boldsymbol{y}\}) + 2\delta(\{\boldsymbol{x}\}, \{\boldsymbol{y}\})$$

for all $\boldsymbol{X}' \subseteq \boldsymbol{X} \setminus \{x\}$ and $\boldsymbol{y}$. Applying this inequality iteratively we conclude that $\delta(\boldsymbol{X}, \{\boldsymbol{y}\}) \le 2\sum_{\boldsymbol{x}\in\boldsymbol{X}} \delta(\{\boldsymbol{x}\}, \{\boldsymbol{y}\})$. Also by Claim 20

$$\delta(\boldsymbol{X}, \boldsymbol{Y}'\{\boldsymbol{y}\}) \le \delta(\boldsymbol{X}, \boldsymbol{Y}') + 2\delta(\boldsymbol{X}, \boldsymbol{Y}'\{\boldsymbol{y}\}),$$

so $\delta(\boldsymbol{X}, \boldsymbol{Y}) \le 2\sum_{\boldsymbol{y}\in\boldsymbol{Y}} \delta(\boldsymbol{X}, \{\boldsymbol{y}\})$. Combining the two inequalities we obtain the desired conclusion. □

To extend the proof to larger k, we generalize the first inequality in this sequence to k-partitions. The same sequence of inequalities then yields the conclusion of Theorem 10. It remains to prove Claim 18.

Claim 22. *For every $2k$-partition $(\boldsymbol{Y}_1, \dots, \boldsymbol{Y}_k, \boldsymbol{Z}_1, \dots, \boldsymbol{Z}_k)$,*

$$\delta(\boldsymbol{Y}_1, \dots, \boldsymbol{Y}_k, \boldsymbol{Z}_1, \dots, \boldsymbol{Z}_k) \leq 2\delta(\boldsymbol{Y}_1\boldsymbol{Z}_1, \dots, \boldsymbol{Y}_k\boldsymbol{Z}_k) + 3\delta(\boldsymbol{Y}_1 \dots \boldsymbol{Y}_k, \boldsymbol{Z}_1 \dots \boldsymbol{Z}_k).$$

Proof. Assume that

$$F(\mathcal{V}) \approx_\delta F_1(Y_1, Z_1) + \cdots + F_t(Y_t, Z_t)$$
$$F(\mathcal{V}) \approx_{\delta'} A(Y_1, \dots, Y_t) + B(Z_1, \dots, Z_t).$$

By the triangle inequality

$$A(Y_1, \dots, Y_t) + B(Z_1, \dots, Z_t) \approx_{\delta+\delta'} F_1(Y_1, Z_1) + \cdots + F_t(Y_t, Z_t).$$

Fixing $Z_1, \dots, Z_t$ to values $\underline{Z}_1, \dots, \underline{Z}_t$ we get the decomposition

$$A(Y_1, \dots, Y_t) \approx_{\delta+\delta'} F_1(Y_1, \underline{Z}_1) + \cdots + F_t(Y_t, \underline{Z}_t) - B(\underline{Z}_1, \dots, \underline{Z}_t).$$

and similarly

$$B(Z_1, \dots, Z_t) \approx_{\delta+\delta'} F_1(\underline{Y}_1, Z_1) + \cdots + F_t(\underline{Y}_t, Z_t) - A(\underline{Y}_1, \dots, \underline{Y}_t).$$

Plugging these into the second equation gives the desired decomposition. □

Proof of Claim 18. We assume that k is a power of two and prove by induction that $\delta(\mathcal{P}) \leq c_k\Delta(\mathcal{P})$, where c_k is the sequence $c_{2k} = 2c_k + 12$, $c_2 = 4$. The base case $k = 2$ follows from Claim 21. Assume the claim holds for k and apply Claim 22 to $\mathcal{P}$. By inductive assumption and Claim 21,

$$\delta(\mathcal{P}) \leq 2 \cdot c_k\Delta(\boldsymbol{Y}_1\boldsymbol{Z}_1, \dots, \boldsymbol{Y}_k\boldsymbol{Z}_k) + 3 \cdot 4\Delta(\boldsymbol{Y}_1 \dots \boldsymbol{Y}_k, \boldsymbol{Z}_1 \dots \boldsymbol{Z}_k).$$

Since $\mathcal{P}$ is a refinement of both these partitions, it follows that $\delta(\mathcal{P}) \leq (2c_k + 12)\Delta(\mathcal{P}) = c_{2k}\Delta(\mathcal{P})$, concluding the induction.

The recurrence solves to $c_k = 8k - 12$, proving the claim when k is a power of two. When it is not, the same reasoning applies to the closest power of two exceeding k (by taking some of the sets in the partition to be empty), which is at most $2k - 1$, proving the desired bound. □

B Submodular Minimization with Inexact Oracle

For a general function F, it is impossible to obtain the exact oracle access to $f = \|D_F\|_2^2$, since it requires exact evaluation of the expectation $\mathbb{E}[D_F(\boldsymbol{X}, \overline{\boldsymbol{X}})^2]$. However, we can obtain arbitrarily good approximations by sampling. That is, given an input $\boldsymbol{X}$ for f, accuracy parameters ε and γ, it can output a value between $f(\boldsymbol{X})$ and $f(\boldsymbol{X}) + \varepsilon$ with high probability at least $1 - \gamma$. If the approximation can be computed in time polynomial in $1/\gamma$ and $1/\varepsilon$, we can say that the approximate oracle is *efficient*. By Fact 7, f can be approximated from $O(\|F\|_4^4/\gamma\varepsilon^2)$ or $O(\|F\|_\infty^4 \log(1/\gamma)/\varepsilon^2)$ queries to F in linear time. Thus, it admits an efficient approximation oracle (i.e., $\|F\|_4^4$ is bounded).

[5] reduces the problem of minimizing a submodular f to minimizing its Lovász extension $\mathbb{E}_{\lambda\sim[0,1]}[f(\{i\colon x_i \geq \lambda\})]$, which is a convex function over $[0, 1]^n$. Applying the ellipsoid method, they obtain an efficient algorithm for minimizing f up to arbitrary accuracy. Their best algorithm has complexity that is inverse logarithmic in the accuracy parameter but assumes the availability of an exact oracle. An approximate oracle for f can still emulate a *weak* separation oracle for the level sets of the Lovász extension, implying the our main POAD_SM algorithmic consequence.