Self-Guided Evolution Strategies with Historical Estimated Gradients

Fei-Yu Liu[1,2] , **Zi-Niu Li**[3] and **Chao Qian**[1*]
[1]National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China
[2]University of Science and Technology of China, Hefei 230027, China
[3]Polixir, Nanjing 210038, China
lfy0012@mail.ustc.edu.cn, ziniu.li@polixir.ai, qianc@lamda.nju.edu.cn

Abstract

Evolution Strategies (ES) are a class of black-box optimization algorithms and have been widely applied to solve problems, e.g., in reinforcement learning (RL), where the true gradient is unavailable. ES estimate the gradient of an objective function with respect to the parameters by randomly sampling search directions and evaluating parameter perturbations in these directions. However, the gradient estimator of ES tends to have a high variance for high-dimensional optimization, thus requiring a large number of samples and making ES inefficient. In this paper, we propose a new ES algorithm SGES, which utilizes historical estimated gradients to construct a low-dimensional subspace for sampling search directions, and adjusts the importance of this subspace adaptively. We prove that the variance of the gradient estimator of SGES can be much smaller than that of Vanilla ES; meanwhile, its bias can be well bounded. Empirical results on benchmark black-box functions and a set of popular RL tasks exhibit the superior performance of SGES over state-of-the-art ES algorithms.

1 Introduction

In many real-world optimization tasks, e.g., model optimization with discrete stochastic variables [Oord *et al.*, 2017] and policy optimization in RL [Lillicrap *et al.*, 2016], the gradient information of objective functions can be unavailable. In such cases, first-order optimization methods such as gradient descent may fail, whereas zeroth-order optimization methods such as ES [Schwefel, 1984; Wierstra *et al.*, 2014; Hansen, 2016; Salimans *et al.*, 2017] can perform well.

ES are known as a class of black-box optimization algorithms, which iteratively try to improve the parameters of the optimization problem. In each iteration, ES estimate the gradient of the given objective function by evaluating parameter perturbations in search directions and then update the parameters using the estimated gradient. ES have been successfully applied in a variety of RL tasks [Salimans *et al.*, 2017; Choromanski *et al.*, 2018; Conti *et al.*, 2018; Mania *et al.*, 2018], due to the advantages of invariance to delayed rewards, tolerance to long time horizons, and good parallelization capability. Other applications include hyperparameter tuning [Friedrichs and Igel, 2005], deep neural network model design [Vidnerova and Neruda, 2017], and meta-learning [Song *et al.*, 2019].

Vanilla ES [Wierstra *et al.*, 2014; Salimans *et al.*, 2017] samples the search directions from the isotropic multivariate Gaussian distribution in the entire space (i.e., full parameter space). However, when the dimensionality of the optimization problem becomes high, the gradient estimator tends to have a high variance, which requires a large number of samples to be robust [Nesterov and Spokoiny, 2017]. To reduce the variance of the gradient estimator, Maheswaranathan *et al.* [2019] recently proposed the Guided ES algorithm, which elongates the search distribution along a subspace spanned by the *surrogate gradients*, i.e., directions that are correlated with the true gradient but may be biased. However, it is not yet clear how to obtain the surrogate gradients in general, and the tradeoff between the entire space and the subspace is also hard to be determined. Choromanski *et al.* [2019] proposed the ASEBO algorithm, which seeks an active subspace from accumulated descent directions by applying principal component analysis (PCA). However, obtaining the active subspace requires decomposing an $n \times n$ matrix (where n is the problem dimensionality, i.e., the number of parameters), which is computationally expensive and thus limits its application.

In this paper, inspired by the promising performance of using historical gradients in first-order optimization methods such as momentum SGD [Sutskever *et al.*, 2013] and Adam [Kingma and Ba, 2015], we propose a Self-Guided ES algorithm with historical estimated gradients, briefly called SGES. SGES dynamically maintains a gradient subspace, spanned by the recent k historical estimated gradients. Note that obtaining the subspace requires decomposing an $n \times k$ matrix, which turns out to be very efficient, as $k \ll n$. The search directions are sampled from a hybrid probabilistic distribution characterized by the gradient subspace and its orthogonal complement, corresponding to exploitation and exploration, respectively. The tradeoff between these two subspaces is adaptively adjusted, based on the goodness of the search directions sampled from them in the last iteration. We

[*]This work was supported by the Fundamental Research Funds for the Central Universities (14380004) and the project of HUAWEI-LAMDA Joint Laboratory of Artificial Intelligence. Chao Qian is the corresponding author.

prove that the variance of the gradient estimator of SGES can be bounded and much smaller than that of Vanilla ES; meanwhile, its bias can be well bounded. Empirical results on black-box functions from the open-sourced *Nevergrad* library [Rapin and Teytaud, 2018] as well as a set of popular RL tasks from the *OpenAI Gym* library [Brockman *et al.*, 2016], show that SGES is efficient, and can achieve better optimization performance than state-of-the-art ES algorithms.

2 Related Work

ES generate a descent direction via finite differences over randomly sampled search directions [Schwefel, 1984; Nesterov and Spokoiny, 2017]. They were traditionally used in low-dimensional regimes and considered ill-equipped for high-dimensional problems [Nesterov and Spokoiny, 2017]. However, a flurry of recent research that combines ES with several effective heuristics, i.e., filtering, normalization, and efficient exploration, has shown that ES can scale better than previously believed [Salimans *et al.*, 2017; Mania *et al.*, 2018; Conti *et al.*, 2018]. Particularly, Salimans *et al.* [2017] have applied ES to solve high-dimensional RL problems, achieving empirical performance comparable to state-of-the-art policy gradient algorithms.

However, the intrinsic high variance of the gradient estimator of ES in high-dimensional optimization leads to high sample complexity [Nesterov and Spokoiny, 2017]. By far, how to reduce the high variance is still a challenging problem. A well-known variant of ES for this purpose is Covariance Matrix Adaptation ES (CMA-ES) [Auger and Hansen, 2012; Hansen, 2016], which adapts the search distribution over parameters with historical search directions. However, it needs to maintain a full $n \times n$ matrix, limiting its application to high-dimensional optimization. Other works include combining random orthogonal with Quasi-Monte Carlo finite differences [Choromanski *et al.*, 2018], and combining surrogate gradient information with random search [Maheswaranathan *et al.*, 2019; Choromanski *et al.*, 2019].

Our work is inspired by the recent two algorithms, Guided ES [Maheswaranathan *et al.*, 2019] and ASEBO [Choromanski *et al.*, 2019]. Guided ES reduces the variance of the gradient estimation in ES by defining a search distribution that is elongated along a subspace spanned by the surrogate gradients. However, how to obtain the surrogate gradients in general is unclear, and it is also not easy to determine the trade-off between the entire space and the subspace. ASEBO seeks an active subspace from accumulated descent directions by applying PCA, but with the cost of high computational complexity. Our proposed algorithm SGES utilizes the historical estimated gradients, adjusts the importance of the gradient subspace adaptively, and can perform efficiently.

Note that there are also some works using gradients in concert with traditional evolutionary algorithms (EA), which apply mutation and recombination operators to update parameters. For example, gradients are used to maintain safe mutations to avoid EA falling into local optima [Lehman *et al.*, 2018b], and help EA solve RL problems [Khadka and Tumer, 2018; Colas *et al.*, 2018; Pourchot and Sigaud, 2019].

3 Evolution Strategies

In this section, we briefly introduce the procedure of ES. Consider the problem of minimizing a black-box function $f : \mathbb{R}^n \to \mathbb{R}$, where the gradient ∇f is unavailable. A popular approach for estimating the gradient is to use *Gaussian Smoothing* [Nesterov and Spokoiny, 2017] to make the function smooth. The Gaussian smoothed version of f is

$$\begin{aligned} f_\sigma(\boldsymbol{\theta}) &= \mathbb{E}_{\boldsymbol{\epsilon}\in\mathcal{N}(0,\mathbf{I}_n)}\left[f(\boldsymbol{\theta}+\sigma\boldsymbol{\epsilon})\right] \\ &= (2\pi)^{-n/2}\int_{\mathbb{R}^n} f(\boldsymbol{\theta}+\sigma\boldsymbol{\epsilon})e^{-\|\boldsymbol{\epsilon}\|^2/2}d\boldsymbol{\epsilon}, \end{aligned}$$

where $\sigma > 0$ is the smoothing parameter, and $\mathbf{I}_n$ is the identity matrix of size n. The gradient of f_σ is given by the formula [Wierstra *et al.*, 2014]:

$$\nabla f_\sigma(\boldsymbol{\theta}) = (1/\sigma)\cdot\mathbb{E}_{\boldsymbol{\epsilon}\in\mathcal{N}(0,\mathbf{I}_n)}\left[f(\boldsymbol{\theta}+\sigma\boldsymbol{\epsilon})\boldsymbol{\epsilon}\right].$$

Though intractable in practice, this gradient can be estimated by various Monte Carlo estimators. The most two popular ones are: *vanilla ES gradient estimator* [Wierstra *et al.*, 2014]

$$\widehat{\nabla} f_\sigma^v(\boldsymbol{\theta}) = \frac{1}{\sigma P}\sum_{i=1}^{P} f(\boldsymbol{\theta}+\sigma\boldsymbol{\epsilon}_i)\boldsymbol{\epsilon}_i;$$

antithetic ES gradient estimator [Choromanski *et al.*, 2018]

$$\widehat{\nabla} f_\sigma^a(\boldsymbol{\theta}) = \frac{1}{2\sigma P}\sum_{i=1}^{P}\left(f(\boldsymbol{\theta}+\sigma\boldsymbol{\epsilon}_i) - f(\boldsymbol{\theta}-\sigma\boldsymbol{\epsilon}_i)\right)\boldsymbol{\epsilon}_i, \quad (1)$$

where $\{\boldsymbol{\epsilon}_1, \ldots, \boldsymbol{\epsilon}_P\}$ are called *search directions* and sampled from some distribution, and P is the sample size. It has been shown that the antithetic estimator has a smaller variance than the vanilla estimator [Choromanski *et al.*, 2018].

ES try to minimize the Gaussian smoothed objective function by directly using stochastic gradient descent, with the antithetic ES gradient estimator. In each iteration, ES first sample search directions $\{\boldsymbol{\epsilon}_1, \ldots, \boldsymbol{\epsilon}_P\}$ from some distribution and estimate the gradient by Eq. (1); then update the current parameters using the estimated gradient. For Vanilla ES, the search directions are sampled from the isotropic multivariate Gaussian distribution $\mathcal{N}(0, \mathbf{I}_n)$.

4 The SGES Algorithm

In this section, we introduce the proposed algorithm SGES, as presented in Algorithm 1. The key of SGES is reflected in two aspects. First, SGES dynamically maintains a gradient subspace, which is used to encode a hybrid probabilistic distribution for sampling search directions (see Section 4.1). Second, to harness different optimization stages, SGES employs an adaptive sampling strategy to sample promising search directions from this distribution (see Section 4.2). We summarize the algorithm implementations in Section 4.3. Finally, we provide theoretical results on variance reduction for the gradient estimator of SGES in Section 4.4.

4.1 Gradient Subspace

During the optimization process, though the true gradient may be unavailable, one can estimate the gradient by Eq. (1). SGES collects the recent k estimated gradients, which are usually linearly independent as $k \ll n$, and constructs two subspaces. Let $\mathbf{G}_t \in \mathbb{R}^{n\times k}$ denote the gradient matrix in iteration t, which consists of the k estimated gradients obtained in iterations $t-k$ to $t-1$. SGES generates two subspaces, the gradient subspace (denoted by $\mathcal{L}_{\mathbf{G}}$) and its orthogonal complement (denoted by $\mathcal{L}_{\mathbf{G}}^{\perp}$), by decomposing $\mathbf{G}_t$.

Intuitively, $\mathcal{L}_{\mathbf{G}}$ is an informative subspace capturing the intrinsic structure of the optimization problem. Thus, utilizing $\mathcal{L}_{\mathbf{G}}$ in the process of sampling search directions can improve the quality of the gradient estimator (e.g., reduce the variance as we will show later), especially in high-dimensional optimization where the curse-of-dimensionality occurs.

To sample search directions, SGES leverages these two disentangled subspaces, i.e., $\mathcal{L}_{\mathbf{G}}$ and $\mathcal{L}_{\mathbf{G}}^{\perp}$, to encode a hybrid probabilistic distribution

$$\mathcal{P} = \begin{cases} \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}_{\mathcal{L}_{\mathbf{G}}}) & \text{with probability } \alpha, \\ \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}_{\mathcal{L}_{\mathbf{G}}^{\perp}}) & \text{with probability } 1-\alpha. \end{cases} \quad (2)$$

That is, with probability α, the search directions are sampled from a multivariate Gaussian distribution with the covariance matrix obtained from $\mathcal{L}_{\mathbf{G}}$, corresponding to exploitation; otherwise, they are sampled from a multivariate Gaussian distribution with the covariance matrix obtained from $\mathcal{L}_{\mathbf{G}}^{\perp}$ (which can also be replaced by the entire space), corresponding to exploration. The parameter $\alpha \in (0, 1)$ characterizes the tradeoff of exploitation versus exploration during optimization.

Unlike Guided ES [Maheswaranathan *et al.*, 2019] which mixes a subspace and the entire space linearly, SGES samples from the gradient subspace and its orthogonal complement probabilistically, making it possible to measure the goodness of these two subspaces and adjust their tradeoff adaptively, as we will show in the next subsection.

4.2 Adaptive Sampling

When sampling search directions from the hybrid probabilistic distribution, using a fixed sampling strategy (i.e., a fixed α) may be inefficient since it cannot harness different optimization stages. For example, an ideal algorithm should sample search directions from the gradient subspace greedily when it is far from global optima, while it should sample more uniformly when it gets stuck into local optima. Thus, SGES employs an adaptive sampling strategy, i.e., uses an adaptive α.

Let $\boldsymbol{x}_{\mathcal{L}_{\mathbf{G}}}$ and $\boldsymbol{x}_{\mathcal{L}_{\mathbf{G}}^{\perp}}$ denote the projection of a vector $\boldsymbol{x}$ on the gradient subspace $\mathcal{L}_{\mathbf{G}}$ and its orthogonal complement $\mathcal{L}_{\mathbf{G}}^{\perp}$, respectively. The optimal tradeoff between $\mathcal{L}_{\mathbf{G}}$ and $\mathcal{L}_{\mathbf{G}}^{\perp}$ (i.e., exploitation versus exploration) in iteration t of SGES can be measured by $\alpha_t = \|\nabla f(\boldsymbol{\theta}_t)_{\mathcal{L}_{\mathbf{G}}}\| / \|\nabla f(\boldsymbol{\theta}_t)_{\mathcal{L}_{\mathbf{G}}^{\perp}}\|$. If α_t is close to 1, the gradient subspace is well informative, where concentrating search directions can make the algorithm converge faster. If α_t is close to 0, the representation of the gradient subspace is insufficient, and thus more exploration is needed. However, α_t cannot be computed exactly since the true gradient is typically unknown in practice.

SGES employs a straightforward way to adjust the value of α adaptively. Divide the search directions in the last iteration (i.e., iteration $t-1$) into two parts: $\{\boldsymbol{\epsilon}_1, \ldots, \boldsymbol{\epsilon}_M\}$ which were sampled from the gradient subspace $\mathcal{L}_{\mathbf{G}}$, and $\{\boldsymbol{\epsilon}_{M+1}, \ldots, \boldsymbol{\epsilon}_P\}$ which were sampled from its orthogonal complement $\mathcal{L}_{\mathbf{G}}^{\perp}$. Let $\widehat{r}_{\mathbf{G}}$ and $\widehat{r}_{\mathbf{G}}^{\perp}$ denote the average of the function evaluations on search directions sampled from $\mathcal{L}_{\mathbf{G}}$ and $\mathcal{L}_{\mathbf{G}}^{\perp}$, respectively. That is,

$$\widehat{r}_{\mathbf{G}} = \frac{1}{M} \sum_{i=1}^{M} \min\{f(\boldsymbol{\theta}_{t-1} + \sigma\boldsymbol{\epsilon}_i), f(\boldsymbol{\theta}_{t-1} - \sigma\boldsymbol{\epsilon}_i)\},$$

$$\widehat{r}_{\mathbf{G}}^{\perp} = \frac{1}{P-M} \sum_{i=M+1}^{P} \min\{f(\boldsymbol{\theta}_{t-1} + \sigma\boldsymbol{\epsilon}_i), f(\boldsymbol{\theta}_{t-1} - \sigma\boldsymbol{\epsilon}_i)\}.$$

Note that for each search direction, there are two perturbations, where the minimum function evaluation is taken. SGES uses $\widehat{r}_{\mathbf{G}}$ and $\widehat{r}_{\mathbf{G}}^{\perp}$ to measure the goodness of the two subspaces $\mathcal{L}_{\mathbf{G}}$ and $\mathcal{L}_{\mathbf{G}}^{\perp}$ in the current iteration, respectively. By comparing $\widehat{r}_{\mathbf{G}}$ with $\widehat{r}_{\mathbf{G}}^{\perp}$, SGES adaptively increases or decreases the value of α as follows:

$$\alpha_t = \begin{cases} \min\{\delta\alpha_{t-1}, \kappa_1\} & \text{if } \widehat{r}_{\mathbf{G}} \text{ does not exist or } \widehat{r}_{\mathbf{G}} < \widehat{r}_{\mathbf{G}}^{\perp}, \\ \max\{\frac{1}{\delta}\alpha_{t-1}, \kappa_2\} & \text{if } \widehat{r}_{\mathbf{G}}^{\perp} \text{ does not exist or } \widehat{r}_{\mathbf{G}} \geq \widehat{r}_{\mathbf{G}}^{\perp}, \end{cases} \quad (3)$$

where $\delta > 1$ is a scaling factor, κ_1 and κ_2 are upper and lower bounds of α, respectively. Intuitively, when the gradient subspace $\widehat{r}_{\mathbf{G}}$ is better, i.e., $\widehat{r}_{\mathbf{G}} < \widehat{r}_{\mathbf{G}}^{\perp}$, SGES increases the probability (i.e., α) of sampling from it; otherwise, the probability is decreased. Note that when $\widehat{r}_{\mathbf{G}}$ does not exist, it implies that the probability of sampling from $\widehat{r}_{\mathbf{G}}$ is too small, and thus the probability is increased; when $\widehat{r}_{\mathbf{G}}^{\perp}$ does not exist, the probability is decreased accordingly.

4.3 The Algorithm

A general flow of SGES is shown in Algorithm 1. It first initializes an archive $\mathcal{G}$ with queue properties and a maximum capacity of k, which is used to store the estimated gradients in the last k iterations. In the warmup phase, SGES behaves as same as Vanilla ES. In each subsequent iteration, SGES obtains a gradient matrix $\mathbf{G}$ by stacking together the estimated gradients in the archive $\mathcal{G}$ and computes the eigenvectors of $\mathbf{G}$ by decomposition, e.g., SVD. The orthogonal basis of the gradient subspace $\mathcal{L}_{\mathbf{G}}$ and its orthogonal complement $\mathcal{L}_{\mathbf{G}}^{\perp}$ are generated by stacking together the corresponding eigenvectors. Afterwards, SGES samples search directions $\{\boldsymbol{\epsilon}_1, ..., \boldsymbol{\epsilon}_P\}$ from the hybrid probabilistic distribution, i.e., Eq. (2), and normalizes them such that $\|\boldsymbol{\epsilon}_i\|^2$ satisfies the chi-square distribution $\chi^2(n)$. These search directions are then used to estimate the gradient by Eq. (1), which is applied to update the parameters. Finally, SGES adaptively adjusts the value of α by Eq. (3), according to the results of function evaluations in computing Eq. (1).

4.4 Variance Reduction

In this subsection, we will show that the bias of the gradient estimator of SGES can be well bounded, and the variance can be much smaller than that of Vanilla ES.

For Vanilla ES, the search direction is sampled from the isotropic multivariate Gaussian distribution $\mathcal{N}(0, \mathbf{I}_n)$, while for SGES, it is sampled from the hybrid probabilistic distribution $\mathcal{P}$, i.e., Eq. (2). The covariance matrix of $\mathcal{P}$ is

$$\boldsymbol{\Sigma} = \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{P}}\left[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^{\top}\right] = \alpha\mathbf{U}\mathbf{U}^{\top} + (1-\alpha)\mathbf{U}^{\perp}(\mathbf{U}^{\perp})^{\top},$$

where $\mathbf{U}$ is an $n \times k$ matrix obtained by stacking together some orthogonal basis of $\mathcal{L}_{\mathbf{G}}$, and $\mathbf{U}^{\perp} \in \mathbb{R}^{n\times(n-k)}$ is obtained similarly.

Algorithm 1 SGES Algorithm

Require: Initial parameters $\boldsymbol{\theta}_0$, objective function f, learning rate η, hyper-parameters α, σ, k, total iterations T, warmup iterations $T_w \geq k$

Process:

```
Initialize an archive $\mathcal{G}$ of maximum capacity $k$;
for $t = 0 : T - 1$ do
    if $t < T_w$ then
        Sample search directions $\boldsymbol{\epsilon}_1, ..., \boldsymbol{\epsilon}_P$ from $\mathcal{N}(0, \mathbf{I}_n)$
    else
        Obtain gradient matrix $\mathbf{G} \in \mathbb{R}^{n \times k}$ from $\mathcal{G}$;
        Generate subspaces $\mathcal{L}_{\mathbf{G}}$ and $\mathcal{L}_{\mathbf{G}}^{\perp}$;
        Sample search directions $\boldsymbol{\epsilon}_1, ..., \boldsymbol{\epsilon}_P$ from Eq. (2);
        Normalize these search directions
    end if
    Compute gradient estimate $\widehat{\nabla} f_{\sigma}^{a}(\boldsymbol{\theta}_t)$ by Eq. (1);
    Update parameters via gradient descent:
        $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \widehat{\nabla} f_{\sigma}^{a}(\boldsymbol{\theta}_t)$;
    Add the gradient estimate $\widehat{\nabla} f_{\sigma}^{a}(\boldsymbol{\theta}_t)$ to $\mathcal{G}$;
    if $t \geq T_w$ then
        Adaptively adjust $\alpha$ by Eq. (3)
    end if
end for
```

For convenience of analysis, only one search direction is sampled (i.e., the parameter P in Eq. (1) is 1) in gradient estimation. Let $\hat{g}_{ves}$ and $\hat{g}_{sges}$ denote the gradient estimator of Vanilla ES and SGES, respectively. Thus, we have

$$\hat{g}_{ves} = \frac{f(\boldsymbol{\theta} + \sigma \boldsymbol{\epsilon}) - f(\boldsymbol{\theta} - \sigma \boldsymbol{\epsilon})}{2\sigma} \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \in \mathcal{N}(0, \mathbf{I}_n); \quad (4)$$

$$\hat{g}_{sges} = \boldsymbol{\Sigma}^{-1} \frac{f(\boldsymbol{\theta} + \sigma \boldsymbol{\epsilon}) - f(\boldsymbol{\theta} - \sigma \boldsymbol{\epsilon})}{2\sigma} \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \in \mathcal{P}. \quad (5)$$

By assuming the regularity of f (i.e., Assumptions 1 and 2) and small enough σ (i.e., Assumption 3), Choromanski *et al.* [2019] proved in Theorem 1 that the expectation of the gradient estimator $\hat{g}_{ves}$ of Vanilla ES is close to the true gradient (i.e., the bias is small), and the variance $\mathrm{Var}[\hat{g}_{ves}]$ is close to $(n+1)\|\nabla f(\boldsymbol{\theta})\|^2$. Note that $\mathrm{Var}[\hat{g}_{ves}]$ here denotes the sum of the variance of each dimension of $\hat{g}_{ves}$.

Assumption 1. *f is L-Lipschitz, i.e., for all $\boldsymbol{\theta}, \boldsymbol{\theta}' \in \mathbb{R}^n$, $|f(\boldsymbol{\theta}) - f(\boldsymbol{\theta}')| \leq L \cdot \|\boldsymbol{\theta} - \boldsymbol{\theta}'\|$.*

Assumption 2. *f has a τ-smooth third derivative tensor, i.e., $f(\boldsymbol{\theta} + \sigma\boldsymbol{\epsilon}) = f(\boldsymbol{\theta}) + \sigma\boldsymbol{\epsilon}^\top \nabla f(\boldsymbol{\theta}) + \frac{1}{2}\sigma^2 \boldsymbol{\epsilon}^\top \mathbf{H}(\boldsymbol{\theta})\boldsymbol{\epsilon} + \frac{1}{6}\sigma^3 \boldsymbol{v}\boldsymbol{v}^\top f'''(\boldsymbol{\theta})\boldsymbol{v}$, where $\mathbf{H}(\boldsymbol{\theta})$ denotes the Hessian matrix of f, $f'''(\boldsymbol{\theta})$ denotes the third derivative of f, and $\boldsymbol{v} \in [\mathbf{0}, \boldsymbol{\epsilon}]$ satisfies $\tau\|\boldsymbol{v}\|^3 \geq |\boldsymbol{v}\boldsymbol{v}^\top f'''(\boldsymbol{\theta})\boldsymbol{v}|$.*

Assumption 3. $\sigma < \frac{1}{35}\sqrt{\frac{\varepsilon \min\{\alpha, 1-\alpha\}}{\tau n^3 \max\{L, 1\}}}$ *for some precision parameter* $\varepsilon > 0$.

Theorem 1. *[Choromanski* et al.*, 2019] Under Assumptions 1 to 3, we have*

$$\|\mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}_n)}[\hat{g}_{ves}] - \nabla f(\boldsymbol{\theta})\| \leq \varepsilon,$$
$$|\mathrm{Var}[\hat{g}_{ves}] - (n+1)\|\nabla f(\boldsymbol{\theta})\|^2| \leq \varepsilon.$$

Their proof idea is to apply the τ-smooth third derivative tensor of f in Assumption 2 to Eq. (4), and then derive the expectation and variance by algebraic calculation. By applying the τ-smooth third derivative tensor to Eq. (5) and following their calculation process, we can prove in Theorem 2 that the bias of the gradient estimator $\hat{g}_{sges}$ of SGES is also well bounded and the variance is close to Ω in Eq. (6). The proof is similar to that of Lemma 3.1 and Theorem 3.2 in [Choromanski *et al.*, 2019].

Theorem 2. *Under Assumptions 1 to 3, we have*

$$\|\mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{P}}[\hat{g}_{sges}] - \nabla f(\boldsymbol{\theta})\| \leq \varepsilon,$$
$$|\mathrm{Var}[\hat{g}_{sges}] - \Omega| \leq \varepsilon,$$

$$\text{where } \Omega = ((k+2)/\alpha) \cdot \|\mathbf{U}^\top \nabla f(\boldsymbol{\theta})\|^2 - \|\nabla f(\boldsymbol{\theta})\|^2 + ((n-k+2)/(1-\alpha)) \cdot \|(\mathbf{U}^\perp)^\top \nabla f(\boldsymbol{\theta})\|^2. \quad (6)$$

To compare the variances of $\hat{g}_{ves}$ and $\hat{g}_{sges}$, we can compare their approximation, i.e., $(n+1)\|\nabla f(\boldsymbol{\theta})\|^2$ and Ω, according to the above two theorems. As shown in Eq. (6), Ω depends on the parameter α in Eq. (2). By simple calculation, we can derive that when $\alpha = \frac{\sqrt{\|\mathbf{U}^\top \nabla f(\boldsymbol{\theta})\|^2 (k+2)}}{\sqrt{\|\mathbf{U}^\top \nabla f(\boldsymbol{\theta})\|^2 (k+2)} + \sqrt{\|(\mathbf{U}^\perp)^\top \nabla f(\boldsymbol{\theta})\|^2 (n-k+2)}}$, Ω reaches the minimum $O(pn + k - 2pk) \cdot \|\nabla f(\boldsymbol{\theta})\|^2$, where $p = \|(\mathbf{U}^\perp)^\top \nabla f(\boldsymbol{\theta})\|^2 / \|\nabla f(\boldsymbol{\theta})\|^2$. When $p = o(1)$ and $k = o(n)$, it holds that $\Omega \ll (n+1)\|\nabla f(\boldsymbol{\theta})\|^2$, implying that $\mathrm{Var}[\hat{g}_{sges}] \ll \mathrm{Var}[\hat{g}_{ves}]$. Thus, when the tradeoff α between the two subspaces $\mathcal{L}_{\mathbf{G}}$ and $\mathcal{L}_{\mathbf{G}}^{\perp}$ is well controlled and the representation of the gradient subspace is sufficient (i.e., $p = o(1)$), the variance of the gradient estimator of SGES can be much smaller than that of Vanilla ES.

5 Experiments

To examine the performance of SGES, we conduct experiments on different high-dimensional tasks, including four black-box functions from the recently open-sourced *Nevergrad* library [Rapin and Teytaud, 2018], and the continuous MuJoCo locomotion tasks (which are widely studied in the RL community) from the *OpenAI Gym* library [Brockman *et al.*, 2016]. For these two types of tasks, the warmup iterations T_w of SGES is set to k and $2k$, respectively. For fair comparisons, we use identical random seeds (2016, 2017, 2018, 2019, and 2020) for all tasks and algorithms. Note that it is not yet clear how to obtain the surrogate gradients (e.g., on RL tasks) for Guided ES; thus, we adopt the recent k estimated gradients as the surrogate gradients, to make the comparison between SGES and Guided ES fair. The initial value of α in SGES is set to 0.5.

5.1 Nevergrad Black-box Functions

First, we empirically compare SGES, Vanilla ES [Salimans *et al.*, 2017], CMA-ES[1] [Hansen, 2016], ASEBO[2] [Choromanski *et al.*, 2019], and Guided ES[3] [Maheswaranathan *et*

[1] https://github.com/CMA-ES/pycma

[2] https://github.com/jparkerholder/ASEBO

[3] https://github.com/brain-research/guided-evolutionary-strategies

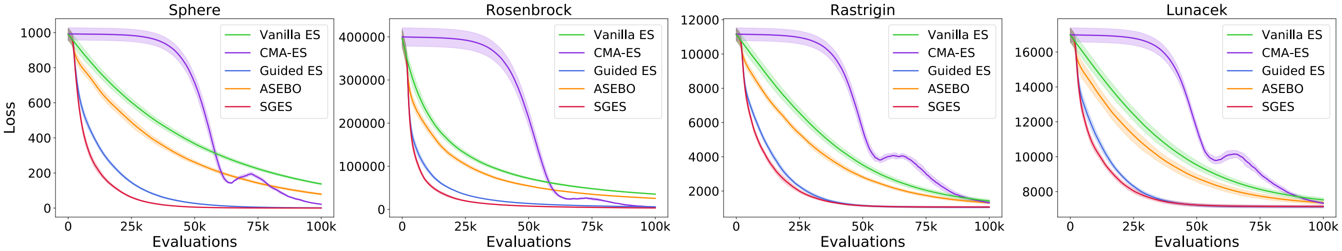
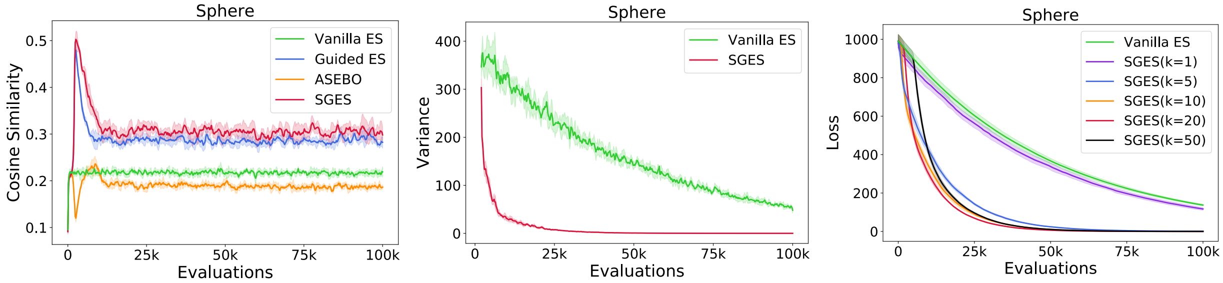


Figure 1: Comparison of the curves averaged over five random seeds for different algorithms on four *Nevergrad* black-box functions.



(a) (b) (c)

Figure 2: (a) Cosine similarity between the estimated gradient and the true gradient for different algorithms (except for CMA-ES which does not estimate the gradient) on the *Sphere* function. (b) Variance of the gradient estimator of Vanilla ES and SGES on the *Sphere* function. (c) Influence of k for the performance of SGES on the *Sphere* function.

al., 2019] on four black-box functions: *Sphere*, *Rosenbrock*, *Rastrigin*, and *Lunacek*, all of which are 1000-dimensional. The smoothing parameter σ is set to a small value 0.01, since there is no noise in function evaluation. For the influence of σ in ES, see [Lehman *et al.*, 2018a]. The learning rate η is chosen properly from $\{0.5, 0.1, 0.01, 0.001\}$. To compare all algorithms as fairly as possible on one function, their common hyper-parameters are set to identical values.

Convergence Speed. Figure 1 shows the mean-curves for different algorithms on black-box functions. We can observe that SGES achieves the best performance across all test functions. The curves of SGES, Guided ES, ASEBO, and Vanilla ES are overlapped at the beginning. This is expected because, in the warmup phase, SGES, Guided ES, and ASEBO behave as same as Vanilla ES. After that, the three algorithms converge faster than Vanilla ES, showing the usefulness of historical gradient information. Note that the superior performance of ASEBO over Vanilla ES and CMA-ES is consistent with that observed in [Choromanski *et al.*, 2019]. ASEBO is worse than SGES and Guided ES, which indicates that the representation of the active subspace may be insufficient. SGES converges faster than Guided ES, disclosing the effectiveness of our adaptive sampling strategy. Note that the loss curve of CMA-ES increases suddenly after about 60k evaluations, which has also been observed in [Choromanski *et al.*, 2019].

Accuracy of Gradient Estimation. To explain why SGES can perform better, we observe the cosine similarity between the estimated gradient and the true gradient on the *Sphere* function. The larger the cosine similarity, the more accurate the estimated gradient. The results are shown in Figure 2(a). It can be observed that the cosine similarity of SGES is the largest and that of Guided ES is the runner-up, which is consistent with their performance rank in Figure 1. Note that the cosine similarity of ASEBO is a little smaller than that of Vanilla ES, but ASEBO achieves better performance, as observed in Figure 1. This is mainly because ASEBO reduces the number of samples in each iteration via its active subspace, thereby able to use more updates than Vanilla ES.

Function	Vanilla ES	CMA-ES	ASEBO	Guided ES	SGES
Sphere	0.48	90.59	136.54	1.51	1.23
Rosenbrock	0.50	92.00	134.34	1.53	1.27
Rastrigin	0.75	90.84	131.40	1.78	1.49
Lunacek	1.33	91.06	132.37	2.44	2.15

Table 1: Average running time (in seconds) over five random seeds for different algorithms on four *Nevergrad* black-box functions.

Variance Reduction. To verify whether SGES can effectively reduce the variance of the gradient estimator, we plot the variance curves of Vanilla ES and SGES on the *Sphere* function, as shown in Figure 2(b). It can be observed that the variance of the gradient estimator of SGES is much smaller than that of Vanilla ES, consistent with our theoretical analysis in Section 4.4.

Influence of k. We also examine the influence of the gradient subspace dimension k, i.e., the number of previously estimated gradients used to construct the subspace. The results of SGES with different values of k on the *Sphere* function are shown in Figure 2(c). We can see that when k is not too small, the convergence speed of SGES is good and not sensitive to the value of k. But if k is set too large, the warmup phase is long and will slow down the entire optimization process. Furthermore, a too large k will make the decomposition of an $n \times k$ matrix in gradient subspace construction computationally expensive. For all test functions, k is set to 20.

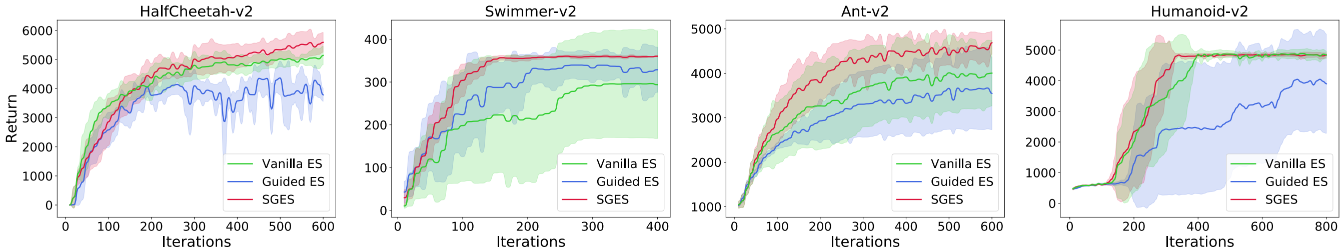


Figure 3: Comparison of the training curves averaged over five random seeds for SGES, Vanilla ES, and Guided ES on four MuJoCo locomotion tasks.

Environment	Timesteps	TRPO	PPO	Vanilla ES	Guided ES	SGES
HalfCheetah-v2	10^7	2385	2876	4729	4032	5038
Swimmer-v2	$5 \cdot 10^5$	72	43	236	318	355
Ant-v2	$2 \cdot 10^7$	2037	2376	3260	2685	3927
Humanoid-v2	$3 \cdot 10^7$	2867	1325	3867	1765	4023

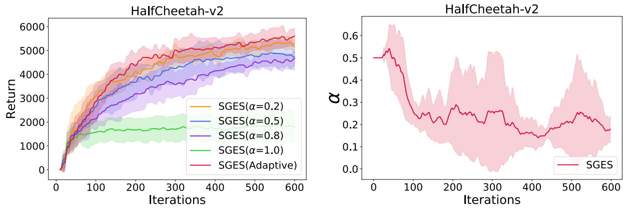
Table 2: Median-returns obtained across five random seeds for different algorithms on four MuJoCo locomotion tasks. The second column indicates the employed timesteps of the algorithms on each task.

Running Time Analysis. Table 1 shows the average running time of each algorithm on four functions. We can see that Vanilla ES, Guided ES, and SGES are the most efficient, taking a similar running time. As expected, CMA-ES and ASEBO are much slower than other algorithms. Their average running time is about two orders of magnitude longer than others in all tasks. This is because both CMA-ES and ASEBO require computing an $n \times n$ matrix in each iteration. Thus, we will only compare Vanilla ES, Guided ES and SGES in RL tasks due to limited computing resources.

5.2 Reinforcement Learning Tasks

Next, we empirically examine the performance of SGES on four MuJoCo locomotion tasks from the *OpenAI Gym* library: *HalfCheetah-v2*, *Swimmer-v2*, *Ant-v2*, and *Humanoid-v2*. We also compare SGES with two popular on-policy deep RL algorithms, TRPO [Schulman *et al.*, 2015] and PPO [Schulman *et al.*, 2017], whose *OpenAI* baseline implementation is used. The linear policy structure is employed, since it is sufficient to capture diverse behaviors on the MuJoCo tasks [Mania *et al.*, 2018]. The techniques of fitness shaping and state normalization are performed, since they are widely used in various RL related works [Salimans *et al.*, 2017; Mania *et al.*, 2018]. For each task, we choose the learning rate η, the smoothing parameter σ, and the sample size P, as recommended by Mania *et al.* [2018]; the gradient subspace dimension k is set to about half of P; the common hyperparameters of all algorithms are set to identical values.

Figure 3 demonstrates the training curves of SGES, Vanilla ES, and Guided ES. We can observe that SGES outperforms Vanilla ES and Guided ES across all the tasks. Furthermore, the shadow (indicating the standard deviation) of SGES is smaller in most tasks, showing that SGES is more robust against random seeds. Particularly, on the *Swimmer-v2* and *Humanoid-v2* tasks, Vanilla ES and Guided ES are very susceptible to random seeds, respectively. Table 2 records the median-returns over five random seeds for all the five algorithms on each task, showing that SGES is still the best.



(a) (b)

Figure 4: For SGES on the *HalfCheetah-v2* task: (a) Adaptive α vs. fixed α; (b) Change curve of adaptive α.

Finally, we examine the effectiveness of adaptive sampling by comparing SGES with fixed $\alpha \in \{0.2, 0.5, 0.8, 1.0\}$ and adaptive α in Eq. (3) on the *HalfCheetah-v2* task. A larger α implies sampling search directions from the gradient subspace with a higher probability. Figure 4(a) shows that a smaller α is better, which may be because there are many suboptimal behaviors (e.g., swaying forward and flipping over) on *HalfCheetah-v2* [Plappert *et al.*, 2018], and thus more exploration is needed. The change curve of adaptive α in Figure 4(b) shows that the adaptive sampling strategy makes α do adjustments automatically and stabilize towards a small value, thus leading to the best performance in Figure 4(a).

6 Conclusion

To mitigate the issue of high variance of the gradient estimator of ES in high-dimensional optimization, we propose a new ES algorithm SGES, which uses the recently estimated gradients to construct a low-dimensional subspace. The search directions for estimating the gradient are sampled from this gradient subspace and its orthogonal complement probabilistically, and the tradeoff between these two subspaces is adjusted adaptively. We prove that compared with Vanilla ES, SGES can reduce the variance effectively. Empirical results on benchmark black-box functions as well as MuJoCo locomotion tasks show the excellent performance of SGES.

References

[Auger and Hansen, 2012] A. Auger and N. Hansen. Tutorial CMA-ES: Evolution strategies and covariance matrix adaptation. In *Proceedings of the 14th ACM Conference on Genetic and Evolutionary Computation (GECCO)*, pages 827–848, Philadelphia, PA, 2012.

[Brockman *et al.*, 2016] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. OpenAI Gym. *CoRR abs/1606.01540*, 2016.

[Choromanski *et al.*, 2018] K. Choromanski, M. Rowland, V. Sindhwani, R. E. Turner, and A. Weller. Structured evolution with compact architectures for scalable policy optimization. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 969–977, Stockholm, Sweden, 2018.

[Choromanski *et al.*, 2019] K. Choromanski, A. Pacchiano, J. Parker-Holder, Y.-H. Tang, and V. Sindhwani. From complexity to simplicity: Adaptive ES-active subspaces for blackbox optimization. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*, pages 10299–10309, Vancouver, Canada, 2019.

[Colas *et al.*, 2018] C. Colas, O. Sigaud, and P. Y. Oudeyer. GEP-PG: Decoupling exploration and exploitation in deep reinforcement learning algorithms. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 1038–1047, Stockholm, Sweden, 2018.

[Conti *et al.*, 2018] E. Conti, V. Madhavan, F. P. Such, J. Lehman, K. O. Stanley, and J. Clune. Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents. In *Advances in Neural Information Processing Systems 31 (NeurIPS)*, pages 5032–5043, Montréal, Canada, 2018.

[Friedrichs and Igel, 2005] F. Friedrichs and C. Igel. Evolutionary tuning of multiple SVM parameters. *Neurocomputing*, 64:107–117, 2005.

[Hansen, 2016] N. Hansen. The CMA evolution strategy: A tutorial. *CoRR abs/1604.00772*, 2016.

[Khadka and Tumer, 2018] S. Khadka and K. Tumer. Evolution-guided policy gradient in reinforcement learning. In *Advances in Neural Information Processing Systems 31 (NeurIPS)*, pages 1196–1208, Montréal, Canada, 2018.

[Kingma and Ba, 2015] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, San Diego, CA, 2015.

[Lehman *et al.*, 2018a] J. Lehman, J. Chen, J. Clune, and K. O. Stanley. ES is more than just a traditional finite-difference approximator. In *Proceedings of the 20th ACM Conference on Genetic and Evolutionary Computation (GECCO)*, pages 450–457, Kyoto, Japan, 2018.

[Lehman *et al.*, 2018b] J. Lehman, J. Chen, J. Clune, and K. O. Stanley. Safe mutations for deep and recurrent neural networks through output gradients. In *Proceedings of the 20th ACM Conference on Genetic and Evolutionary Computation (GECCO)*, pages 117–124, Kyoto, Japan, 2018.

[Lillicrap *et al.*, 2016] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. In *Proceedings of the 4th International Conference on Learning Representations (ICLR)*, San Juan, Puerto Rico, 2016.

[Maheswaranathan *et al.*, 2019] N. Maheswaranathan, L. Metz, G. Tucker, D. Choi, and J. Sohl-Dickstein. Guided evolutionary strategies: Augmenting random search with surrogate gradients. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pages 4264–4273, Long Beach, CA, 2019.

[Mania *et al.*, 2018] H. Mania, A. Guy, and B. Recht. Simple random search of static linear policies is competitive for reinforcement learning. In *Advances in Neural Information Processing Systems 31 (NeurIPS)*, pages 1805–1814, Montréal, Canada, 2018.

[Nesterov and Spokoiny, 2017] Y. E. Nesterov and V. G. Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17(2):527–566, 2017.

[Oord *et al.*, 2017] A. Oord, O. Vinyals, and K. Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems 30 (NIPS)*, pages 6306–6315, Long Beach, CA, 2017.

[Plappert *et al.*, 2018] M. Plappert, R. Houthooft, P. Dhariwal, S. Sidor, R. Y. Chen, X. Chen, T. Asfour, P. Abbeel, and M. Andrychowicz. Parameter space noise for exploration. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, Vancouver, BC, 2018.

[Pourchot and Sigaud, 2019] A. Pourchot and O. Sigaud. CEM-RL: Combining evolutionary and gradient-based methods for policy search. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, New Orleans, LA, 2019.

[Rapin and Teytaud, 2018] J. Rapin and O. Teytaud. Nevergrad - A gradient-free optimization platform. http://GitHub.com/FacebookResearch/Nevergrad, 2018.

[Salimans *et al.*, 2017] T. Salimans, J. Ho, X. Chen, and I. Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *CoRR abs/1703.03864*, 2017.

[Schulman *et al.*, 2015] J. Schulman, S. Levine, P. Abbeel, M. I. Jordan, and P. Moritz. Trust region policy optimization. In *Proceedings of the 32th International Conference on Machine Learning (ICML)*, pages 1889–1897, Lille, France, 2015.

[Schulman *et al.*, 2017] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *CoRR abs/1707.06347*, 2017.

[Schwefel, 1984] H. P. Schwefel. Evolution strategies: A family of non-linear optimization techniques based on imitating some principles of organic evolution. *Annals of Operations Research*, 1(2):165–167, 1984.

[Song *et al.*, 2019] X.-Y. Song, W.-B. Gao, Y.-X. Yang, K. Choromanski, A. Pacchiano, and Y.-H. Tang. ES-MAML: Simple Hessian-free meta learning. *CoRR abs/1910.01215*, 2019.

[Sutskever *et al.*, 2013] I. Sutskever, J. Martens, G. E. Dahl, and G. E. Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pages 1139–1147, Atlanta, GA, 2013.

[Vidnerova and Neruda, 2017] P. Vidnerova and R. Neruda. Evolution strategies for deep neural network models design. In *Proceedings of the 17th Conference on Information Technologies - Applications and Theory (ITAT)*, pages 159–166, Martinské hole, Slovakia, 2017.

[Wierstra *et al.*, 2014] D. Wierstra, T. Schaul, T. Glasmachers, Y. Sun, J. Peters, and J. Schmidhuber. Natural evolution strategies. *Journal of Machine Learning Research*, 15(1):949–980, 2014.