Explicit Gradient Learning for Black-Box Optimization

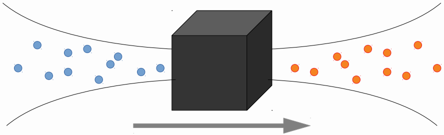
Elad Sarafian*, Mor Sinay*,
Yoram Louzoun, Noa Agmon and Sarit Kraus

Bar-Ilan University

Intelligent Agent, Interaction & Modelling Lab

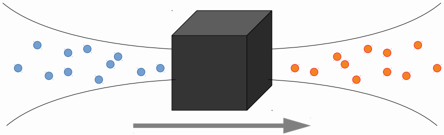
Definition: A Black-Box Function

$f : \Omega \to \mathbb{R}$, $\Omega \subseteq \mathbb{R}^n$ is a Black-Box function if one can sample $y = f(x)$ at $x \in \Omega$, but has no prior knowledge of its analytical form.



Definition: A Black-Box Function

$f : \Omega \to \mathbb{R}$, $\Omega \subseteq \mathbb{R}^n$ is a Black-Box function if one can sample $y = f(x)$ at $x \in \Omega$, but has no prior knowledge of its analytical form.



Black-Box Optimization (BBO)

$$x^* = \arg\min_{x \in \Omega} f(x) \tag{1}$$

For C number of evaluation points, search for x^* and return the best candidate

$$\hat{x} = \arg \min_{x_i,\ i=1,\dots,C} f(x_i) \tag{2}$$

Black-Box Optimization: Applications

Machine Learning:

- **Hyperparameter tuning:** all the non-differential parameters of a learning algorithm.

Machine Learning:

- **Hyperparameter tuning:** all the non-differential parameters of a learning algorithm.
- **Reinforcement Learning:** Find the optimal parameters of a policy $\pi_\theta : s \to a$ s.t. the expected utility function is maximized

$$\theta^* = \arg\max_\theta \mathbb{E}_{\tau \sim \pi_\theta} [R(\tau)]$$

In the real-world everything is a Black-Box...

Black-Box Optimization: Applications

Machine Learning:

- **Hyperparameter tuning:** all the non-differential parameters of a learning algorithm.
- **Reinforcement Learning:** Find the optimal parameters of a policy $\pi_\theta : s \to a$ s.t. the expected utility function is maximized

$$\theta^* = \arg\max_\theta \mathbb{E}_{\tau \sim \pi_\theta} [R(\tau)]$$

In the real-world everything is a Black-Box...

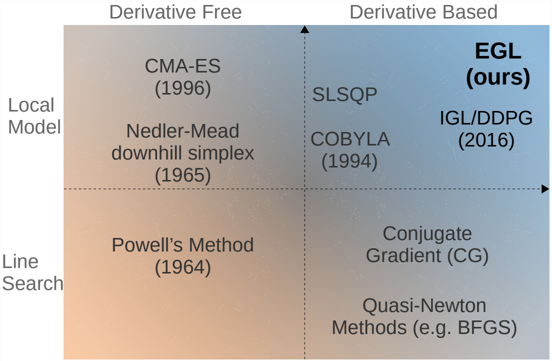
Machine Learning:

- **Hyperparameter tuning:** all the non-differentiable parameters of a learning algorithm.
- **Reinforcement Learning:** Find the optimal parameters of a policy $\pi_\theta : s \to a$ s.t. the expected utility function is maximized

$$\theta^* = \arg\max_{\theta} \mathbb{E}_{\tau \sim \pi_\theta} \left[R(\tau) \right]$$

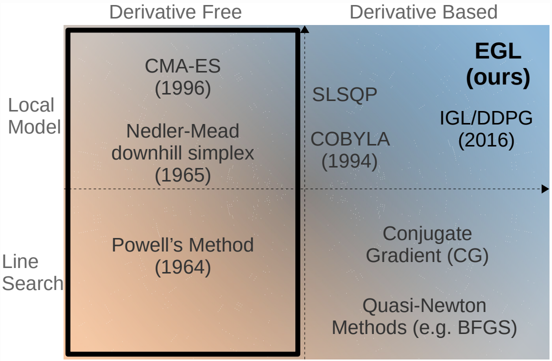
In the real-world, everything is a Black-Box.

BBO methods: Taxonomy



The Black-Box Optimization Taxonomy

BBO methods: Taxonomy



The Black-Box Optimization Taxonomy

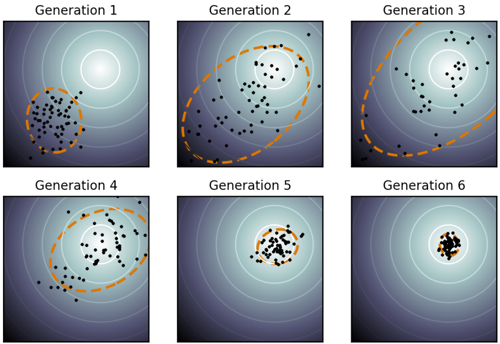
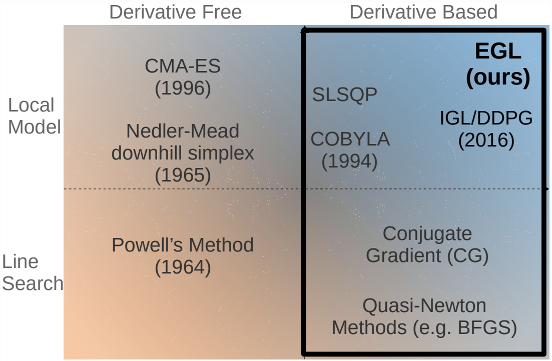


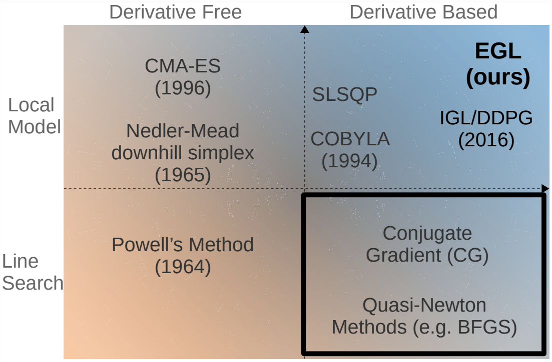
Figure 1: CMA-ES algorithm at work (from Wikipedia)



The Black-Box Optimization Taxonomy

Assume a differentiable function $f \in \mathcal{C}^1$ and use a gradient estimation to direct the search process.

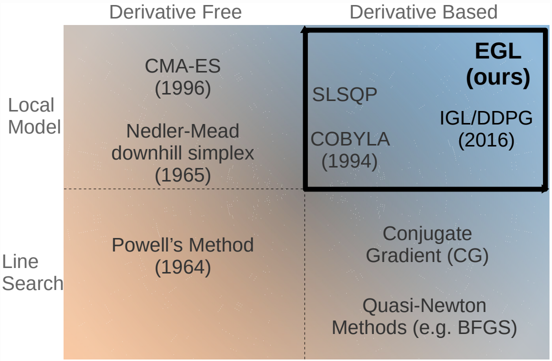
BBO methods: Taxonomy



The Black-Box Optimization Taxonomy

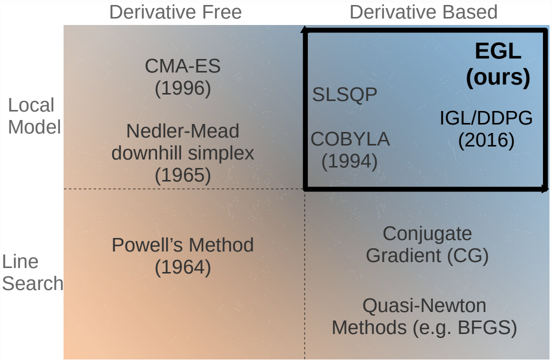
In Line-Search methods, the directional derivative may be estimated by numerical methods, e.g.

$$n \cdot \nabla f(x) \approx \frac{f(x + \Delta n) - f(x - \Delta n)}{2\Delta}$$



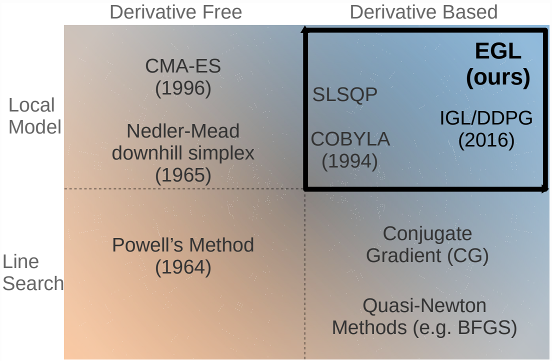
The Black-Box Optimization Taxonomy

In Model-Based methods, the gradient can be estimated by fitting a parametric model $f_\theta \approx f$ and following the parametric gradient ∇f_θ



The Black-Box Optimization Taxonomy

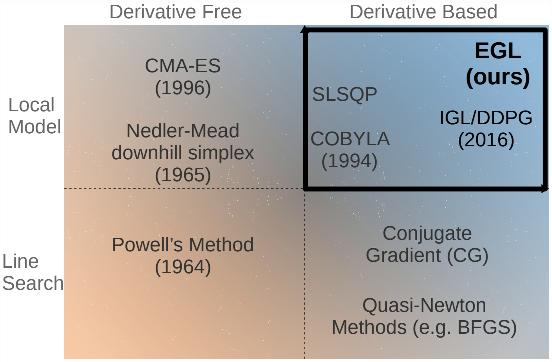
EGL takes Model-Derivative-Based methods forward:



The Black-Box Optimization Taxonomy

EGL takes Model-Derivative-Based methods forward:

1. Instead of learning the function and obtain the parametric gradient. It **directly** fits a global model of the gradient from the data.



The Black-Box Optimization Taxonomy

EGL takes Model-Derivative-Based methods forward:

1. Instead of learning the function and obtain the parametric gradient. It **directly** fits a global model of the gradient from the data.
2. It works with merely locally-integrable functions.

To develop EGL, let's take a closer look at Model-Based methods with Neural-Network parameterization.

Indirect Gradient Learning (IGL)

Neural-Networks excel in fitting models to data $f_\theta \approx f$

Indirect Gradient Learning (IGL)

Neural-Networks excel in fitting models to data $f_\theta \approx f$

1. Given a set of samples around a candidate x_k, minimize the MSE objective

$$\theta^* = \arg\min_\theta \sum_i \|y_i - f_\theta(x_i)\|^2$$

Indirect Gradient Learning (IGL)

Neural-Networks excel in fitting models to data $f_\theta \approx f$

1. Given a set of samples around a candidate x_k, minimize the MSE objective

$$\theta^* = \arg\min_\theta \sum_i \|y_i - f_\theta(x_i)\|^2$$

2. Approximate the gradient with the parametric gradient $\nabla f \approx \nabla f_\theta$

Indirect Gradient Learning (IGL)

Neural-Networks excel in fitting models to data $f_\theta \approx f$

1. Given a set of samples around a candidate x_k, minimize the MSE objective

$$\theta^* = \arg\min_\theta \sum_i \|y_i - f_\theta(x_i)\|^2$$

2. Approximate the gradient with the parametric gradient $\nabla f \approx \nabla f_\theta$
3. Take a gradient-descent step

$$x_{k+1} = x_k - \alpha \nabla f_\theta(x_k)$$

Indirect Gradient Learning (IGL)

Neural-Networks excel in fitting models to data $f_\theta \approx f$

1. Given a set of samples around a candidate x_k, minimize the MSE objective

$$\theta^* = \arg\min_\theta \sum_i \|y_i - f_\theta(x_i)\|^2$$

2. Approximate the gradient with the parametric gradient $\nabla f \approx \nabla f_\theta$
3. Take a gradient-descent step

$$x_{k+1} = x_k - \alpha \nabla f_\theta(x_k)$$

4. Sample points around the new candidate and repeat.

Roots in the Deep Deterministic Policy Gradient (DDPG) seminal paper (2016). Have been applied successfully in robotics domains.

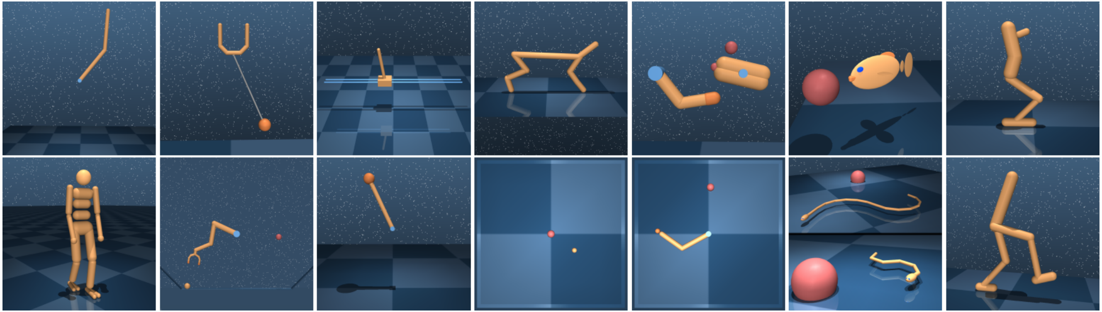


Figure 2: From the DeepMind Control suite Github

Drawbacks of IGL

Pickle I

The gradient is never explicitly learned, nor we obtain any guarantees for the accuracy of its estimation.

Drawbacks of IGL

Pickle I

The gradient is never explicitly learned, nor we obtain any guarantees for the accuracy of its estimation.

Pickle II

For Neural-Networks, the parametric gradient ∇f_θ may be discontinuous even if the objective is continuous.

Pickle I

The gradient is never explicitly learned, nor we obtain any guarantees for the accuracy of its estimation.

Pickle II

For Neural-Networks, the parametric gradient ∇f_θ may be discontinuous even if the objective is continuous.

To overcome these drawbacks of IGL, we wish to learn the gradient explicitly.

Drawbacks of IGL

Pickle I

The gradient is never explicitly learned, nor we obtain any guarantees for the accuracy of its estimation.

Pickle II

For Neural-Networks, the parametric gradient ∇f_θ may be discontinuous even if the objective is continuous.

To overcome these drawbacks of IGL, we wish to learn the gradient explicitly.

Problem

We cannot sample from $\nabla f(x)$ directly.

Drawbacks of IGL

Pickle I

The gradient is never explicitly learned, nor we obtain any guarantees for the accuracy of its estimation.

Pickle II

For Neural-Networks, the parametric gradient ∇f_θ may be discontinuous even if the objective is continuous.

To overcome these drawbacks of IGL, we wish to learn the gradient explicitly.

Problem

We cannot sample from $\nabla f(x)$ directly.

Solution: Learning a surrogate

Instead of learning $\nabla f(x)$ we learn the *mean-gradient* $g_\varepsilon(x)$: averages over the gradient in a volume $V_\varepsilon(x)$ s.t. $\|x' - x\| \leq \varepsilon$ for all $x' \in V_\varepsilon(x)$.

Explicit Gradient Learning (EGL)

Recall the first order Taylor expression for differentiable functions

$$f(x+\tau) = f(x) + \nabla f(x) \cdot \tau + O(\|\tau\|^2).$$

Recall the first order Taylor expression for differentiable functions

$$f(x+\tau) = f(x) + \nabla f(x) \cdot \tau + O(\|\tau\|^2).$$

Definition: The Mean-Gradient

The mean-gradient at x with $\varepsilon > 0$ averaging radius is

$$g_\varepsilon(x) = \arg\min_{g \in \mathbb{R}^n} \int_{V_\varepsilon(x)} |g \cdot \tau - f(x+\tau) + f(x)|^2 d\tau$$

where $V_\varepsilon(x) \subset \mathbb{R}^n$ is a convex subset s.t. $\|x' - x\| \leq \varepsilon$ for all $x' \in V_\varepsilon(x)$ and the integral domain is over τ s.t. $x + \tau \in V_\varepsilon(x)$.

Characteristics of the Mean-Gradient

$$g_\varepsilon(x) = \arg\min_{g\in\mathbb{R}^n} \int\limits_{V_\varepsilon(x)} |g\cdot\tau - f(x+\tau) + f(x)|^2 d\tau$$

Benefit I: Continuity

If $f(x)$ is continuous in V s.t. $V_\varepsilon(x) \subset V$ then the mean-gradient is a continuous function at x.

Characteristics of the Mean-Gradient

$$g_\varepsilon(x) = \arg\min_{g \in \mathbb{R}^n} \int\limits_{V_\varepsilon(x)} |g \cdot \tau - f(x+\tau) + f(x)|^2 d\tau$$

Benefit I: Continuity

If $f(x)$ is continuous in V s.t. $V_\varepsilon(x) \subset V$ then the mean-gradient is a continuous function at x.

Benefit II: Controllable Accuracy

For any differentiable function f with a continuous gradient, there is $\kappa_g > 0$, so that for any $\varepsilon > 0$ the mean-gradient satisfies $\|g_\varepsilon(x) - \nabla f(x)\| \leq \kappa_g \varepsilon$ for all $x \in \Omega$.

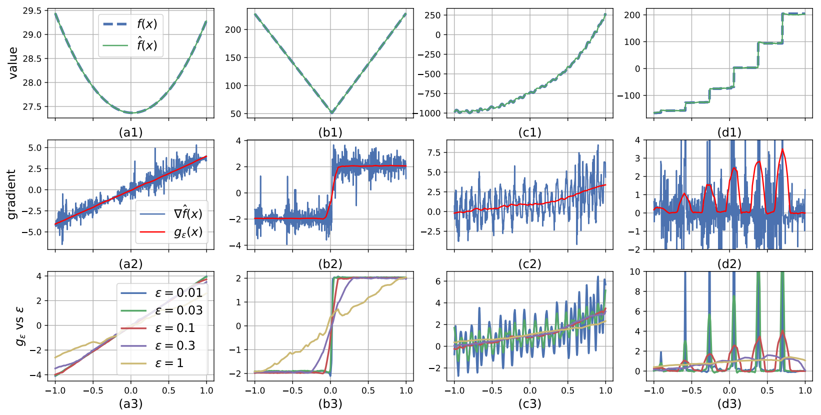


Figure 3: Comparing indirect gradient learning and explicit gradient learning for 4 typical functions: (a) parabolic; (b) piecewise linear; (c) multiple local minima; (d) step function.

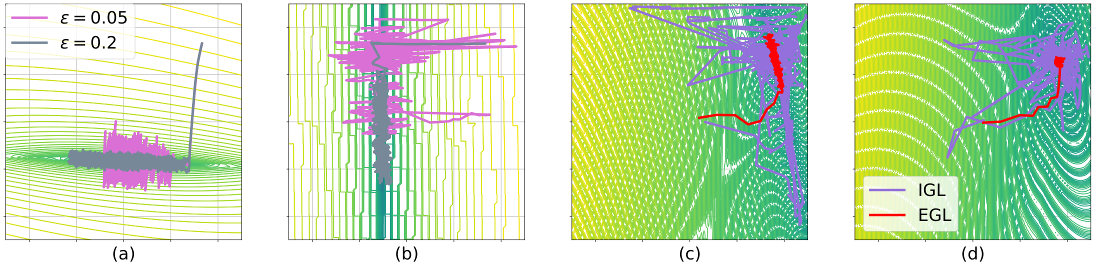


Figure 4: Visualizing EGL and IGL with different ε for various 2D problems from COCO test suite.

$$g_\varepsilon(x) = \arg\min_{g \in \mathbb{R}^n} \int_{V_\varepsilon(x)} |g \cdot \tau - f(x+\tau) + f(x)|^2 d\tau$$

Learning the mean-gradient is done by minimizing the Monte-Carlo approximation of the Mean-Gradient:

Explicit Gradient Learning: Design

Learning the mean-gradient is done by minimizing the Monte-Carlo approximation of the Mean-Gradient:

1. Sample set of pairs of observations $\mathcal{D}_k = \{(x_i, y_i)\}_{i=1}^{m}$ s.t. $x_i \in V_\varepsilon(x_k)$ where x_k is a candidate solution.

Learning the mean-gradient is done by minimizing the Monte-Carlo approximation of the Mean-Gradient:

1. Sample set of pairs of observations $\mathcal{D}_k = \{(x_i, y_i)\}_{i=1}^{m}$ s.t. $x_i \in V_\varepsilon(x_k)$ where x_k is a candidate solution.
2. Minimize the loss function

$$\mathcal{L}_{k,\varepsilon}(\theta) = \sum_{i=1}^{m} \sum_{x_j \in V_\varepsilon(x_i)} |(x_j - x_i) \cdot g_\theta(x_i) - y_j + y_i|^2 \tag{3}$$

$$g_\varepsilon(x) = \arg\min_{g \in \mathbb{R}^n} \int_{V_\varepsilon(x)} |g \cdot \tau - f(x+\tau) + f(x)|^2 d\tau$$

Explicit Gradient Learning: Design

Learning the mean-gradient is done by minimizing the Monte-Carlo approximation of the Mean-Gradient:

1. Sample set of pairs of observations $\mathcal{D}_k = \{(x_i, y_i)\}_{i=1}^m$ s.t. $x_i \in V_\varepsilon(x_k)$ where x_k is a candidate solution.
2. Minimize the loss function

$$\mathcal{L}_{k,\varepsilon}(\theta) = \sum_{i=1}^{m} \sum_{x_j \in V_\varepsilon(x_i)} |(x_j - x_i) \cdot g_\theta(x_i) - y_j + y_i|^2 \tag{4}$$

Theorem + Corollary

Given a proper set of samples (denoted as a poised set), any Lipschitz continuous Neural Network that optimizes Eq. (4) is a a controllably accurate model.

$$\|\nabla f(x) - g_\theta(x)\| \leq \kappa_g \varepsilon$$

Proposition (Bertsekas (1999), Proposition 1.2.3)

Suppose $f : \mathbb{R}^n \to \mathbb{R}$ is κ_f-smooth and bounded below. Let $x_{k+1} = x_k - \alpha \nabla f(x_k)$ and $\alpha \leq \frac{1}{\kappa_f}$. Then $\|\nabla f(x_k)\| \xrightarrow{k \to \infty} 0$.

Explicit Gradient Learning: Convergence

Proposition (Bertsekas (1999), Proposition 1.2.3)

Suppose $f : \mathbb{R}^n \to \mathbb{R}$ is κ_f-smooth and bounded below. Let $x_{k+1} = x_k - \alpha \nabla f(x_k)$ and $\alpha \leq \frac{1}{\kappa_f}$. Then $\|\nabla f(x_k)\| \xrightarrow{k \to \infty} 0$.

We prove that EGL converges to a stationary point and to the global optimum if the problem is convex.

Explicit Gradient Learning: Convergence

Proposition (Bertsekas (1999), Proposition 1.2.3)

Suppose $f : \mathbb{R}^n \to \mathbb{R}$ is κ_f-smooth and bounded below. Let $x_{k+1} = x_k - \alpha \nabla f(x_k)$ and $\alpha \leq \frac{1}{\kappa_f}$. Then $\|\nabla f(x_k)\| \xrightarrow{k\to\infty} 0$.

We prove that EGL converges to a stationary point and to the global optimum if the problem is convex.

Convergence of EGL

Suppose a controllable mean-gradient model g_ε with error constant κ_g, the gradient descent iteration $x_{k+1} = x_k - \alpha_k g_{\varepsilon_k}(x_k)$ with α_k s.t. $\frac{5\varepsilon_k}{\|\nabla f(x_k)\|} \leq \alpha_k \leq \min\left(\frac{1}{\kappa_g}, \frac{1}{\kappa_f}\right)$ guarantees:

1. Monotonically decreasing steps s.t. $f(x_{k+1}) \leq f(x_k) - 2.25\frac{\varepsilon^2}{\alpha}$.
2. $\|\nabla f(x_k)\| \xrightarrow{k\to\infty} 0$ for a proper choice of ε_k.

Explicit Gradient Learning: Convergent Algorithm

Algorithm 1: Convergent EGL

Input: $x_0, \alpha, \varepsilon, \gamma_\alpha < 1, \gamma_\varepsilon < 1, \bar{\varepsilon}$

$k = 0$

while $\varepsilon \leq \bar{\varepsilon}$ **do**

  Build Model:

    Collect data $\{(x_i, y_i)\}_1^m, x_i \in V_\varepsilon(x_k)$

    Learn a local model $g_\varepsilon(x_k)$

  Gradient Descent:

    $x_{k+1} \leftarrow x_k - \alpha g_\varepsilon(x_k)$

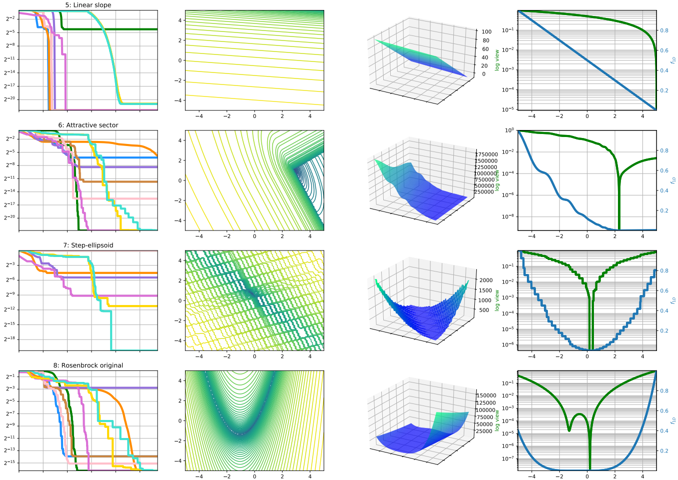
    if $f(x_{k+1}) > f(x_k) - 2.25\frac{\varepsilon^2}{\alpha}$ **then**

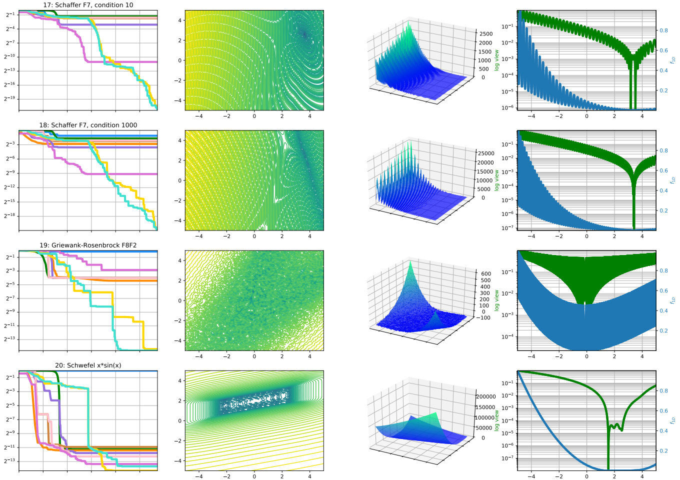
      $\alpha \leftarrow \gamma_\alpha \alpha$

      $\varepsilon \leftarrow \gamma_\alpha \gamma_\varepsilon \varepsilon$

  $k \leftarrow k + 1$

return x_k


5: Linear slope
6: Attractive sector
7: Step-ellipsoid
8: Rosenbrock original
log view
f_{10}



17: Schaffer F7, condition 10
18: Schaffer F7, condition 1000
19: Griewank-Rosenbrock F8F2
20: Schwefel x*sin(x)
log view
f_{1D}

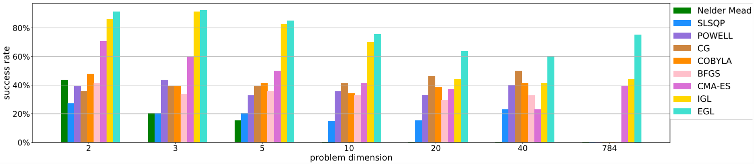


Figure 5: Comparing the success rate for a budget $C = 150 \cdot 10^3$.

Evaluation In the COCO Test Suite

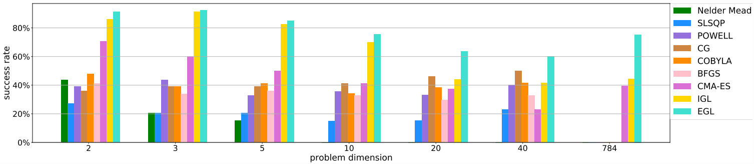


Figure 5: Comparing the success rate for a budget $C = 150 \cdot 10^3$.

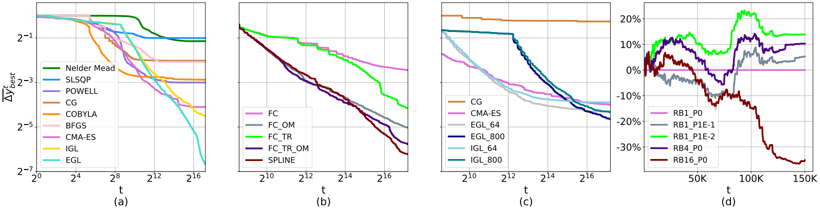
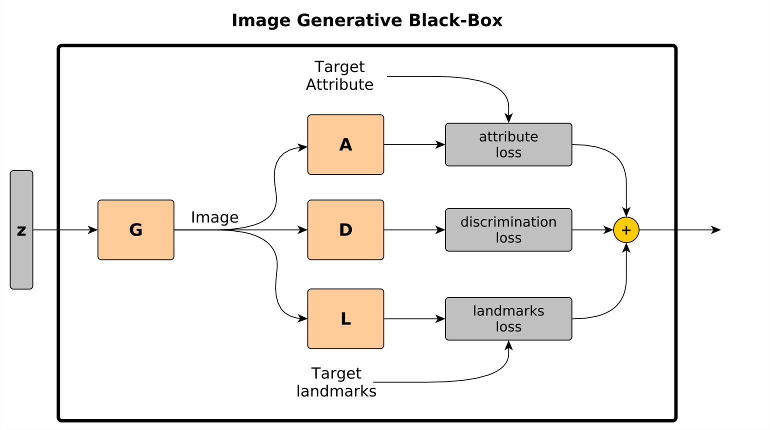
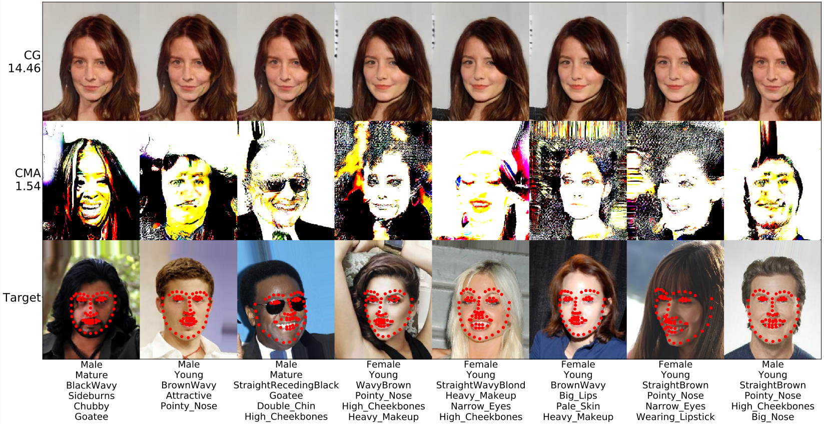


Figure 6: The scaled distance Δy^t_{best} as a function of $t \in [1, .., C]$ for: (a) EGL and baselines on 40D, (b) dynamic mapping ablation test, (c) Different m samples on the 784D set.



$$f_{al}(z) = \lambda_a \mathcal{L}_a(G(z)) + \lambda_l \mathcal{L}_l(G(z)) + \lambda_g \tanh(D(G(z)))$$

Searching the Latent Space of Generative Models



Searching the Latent Space of Generative Models

Searching the Latent Space of Generative Models

Summary

- EGL is a model-based and derivative based BBO method.

Summary

- EGL is a model-based and derivative based BBO method.
- EGL can optimize non-convex and noisy functions.

Summary

- EGL is a model-based and derivative based BBO method.
- EGL can optimize non-convex and noisy functions.
- EGL converges to a local minimum.

Summary

- EGL is a model-based and derivative based BBO method.
- EGL can optimize non-convex and noisy functions.
- EGL converges to a local minimum.
- EGL outperforms existing methods both in a synthetic test-suite and real-world optimization application.

Thank you for your attention