Stochastic Simultaneous Optimistic Optimization

Michal Valko, Alexandra Carpentier, Rémi Munos
INRIA Lille - Nord Europe, France & University of Cambridge, UK

SequeL – INRIA Lille
ICML 2013
Atlanta, GA, June 2013

Setting

- **Goal:** Maximize $f : \mathcal{X} \to R$ given a budget of n evaluations.

Setting

- **Goal:** Maximize $f : \mathcal{X} \to R$ given a budget of n evaluations.
- **Challenges:** f is ***stochastic*** and has ***unknown smoothness***

Setting

- **Goal:** Maximize $f : \mathcal{X} \to R$ given a budget of n evaluations.
- **Challenges:** f is ***stochastic*** and has ***unknown smoothness***
- **Protocol:** At round t, select state x_t, observe r_t such that

$$\mathbb{E}[r_t | x_t] = f(x_t).$$

After n rounds, return a state $x(n)$.

Setting

- **Goal:** Maximize $f : \mathcal{X} \to R$ given a budget of n evaluations.
- **Challenges:** f is ***stochastic*** and has ***unknown smoothness***
- **Protocol:** At round t, select state x_t, observe r_t such that

$$\mathbb{E}[r_t|x_t] = f(x_t).$$

After n rounds, return a state $x(n)$.

- **Loss:** $R_n = \sup_{x\in\mathcal{X}} f(x) - f(x(n))$

Setting

- **Goal:** Maximize $f : \mathcal{X} \to R$ given a budget of n evaluations.
- **Challenges:** f is ***stochastic*** and has ***unknown smoothness***
- **Protocol:** At round t, select state x_t, observe r_t such that

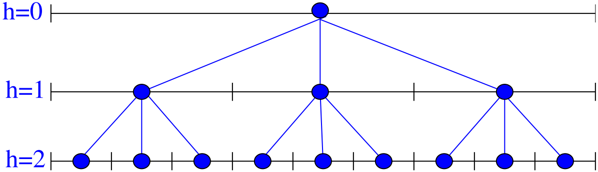
$$\mathbb{E}[r_t | x_t] = f(x_t).$$

After n rounds, return a state $x(n)$.

- **Loss:** $R_n = \sup_{x \in \mathcal{X}} f(x) - f(x(n))$

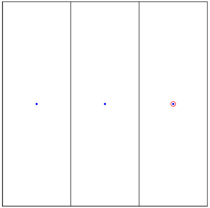
StoSOO operates on a given **hierarchical partitioning**

- For any h, $\mathcal{X}$ is partitioned in K^h cells $(X_{h,i})_{0 \leq i \leq K^h-1}$.
- K-ary tree $\mathcal{T}_\infty$ where depth $h = 0$ is the whole $\mathcal{X}$.

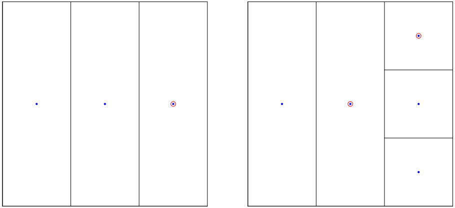


- StoSOO adaptively creates finer and finer partitions of $\mathcal{X}$.
- $x_{h,i} \in X_{h,i}$ is a specific state per cell where f is evaluated

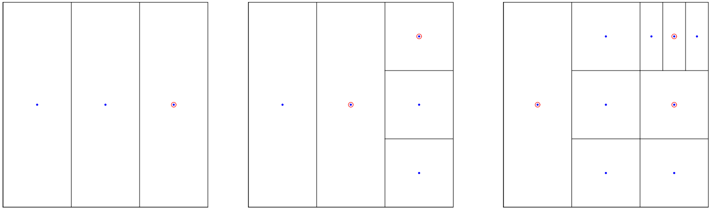
StoSOO adaptively creates finer and finer partitions of $\mathcal{X}$



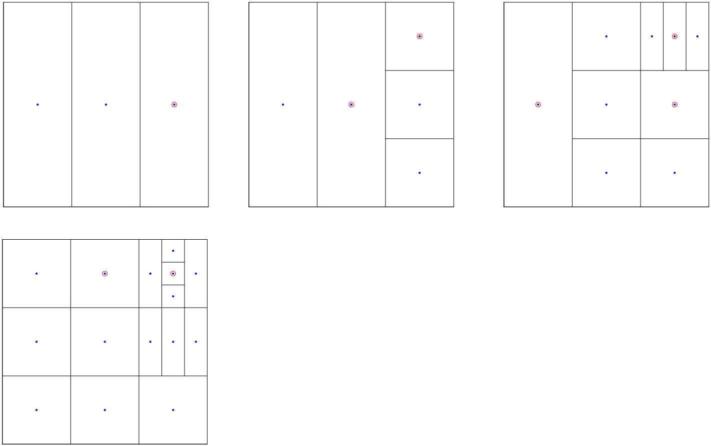
StoSOO adaptively creates finer and finer partitions of $\mathcal{X}$



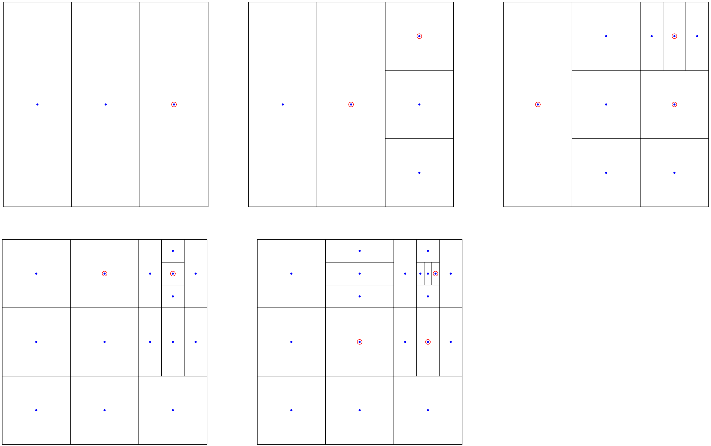
StoSOO adaptively creates finer and finer partitions of $\mathcal{X}$



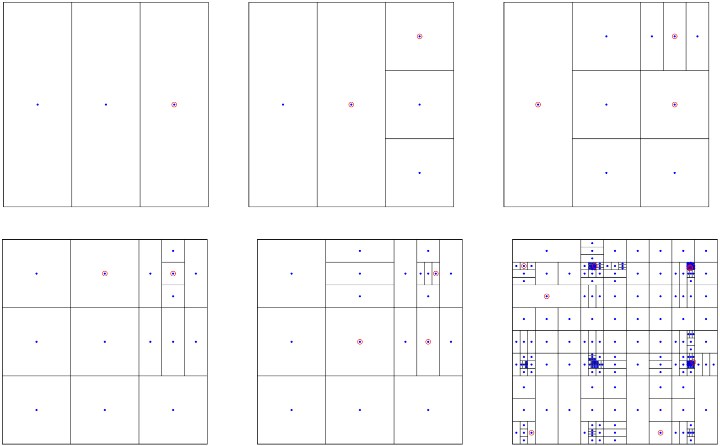
StoSOO adaptively creates finer and finer partitions of $\mathcal{X}$



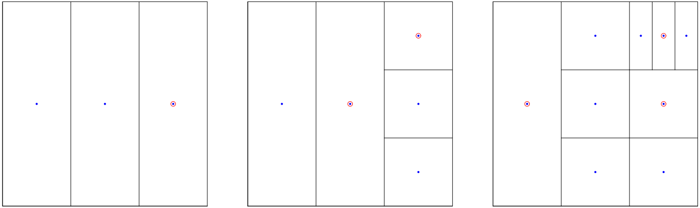
StoSOO adaptively creates finer and finer partitions of $\mathcal{X}$



StoSOO adaptively creates finer and finer partitions of $\mathcal{X}$



Challenge 1: **Stochasticity**

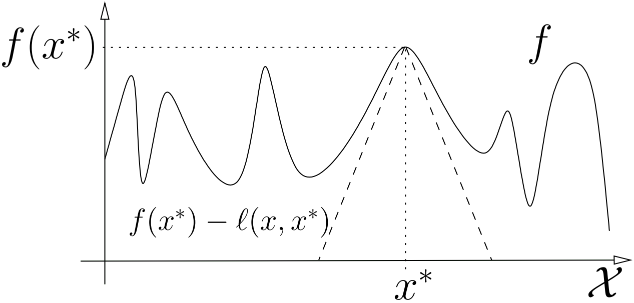


- cannot evaluate the cell only once before splitting
- cannot return the highest x_t encountered as $x(n)$

Challenge 2: **Unknown smoothness**

Assumption about the function: f **is locally smooth** w.r.t. a semi-metric ℓ around one global maximum x^*:

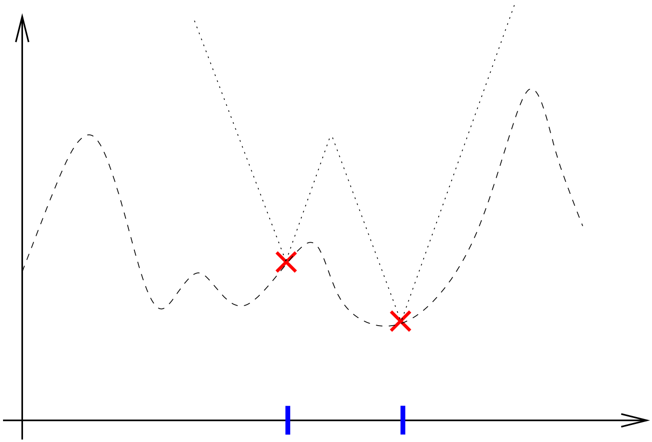
$$\forall x \in \mathcal{X} : f(x^*) - f(x) \leq \ell(x, x^*)$$



"f does not decrease too fast around x^*"

Challenge 2: **Unknown smoothness**

What can we do if the smoothness is known?



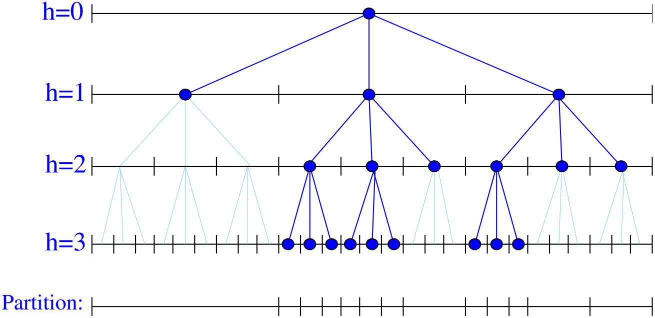
Comparison

	Deterministic function	**Stochastic** function
known smoothness	DOO	Zooming or HOO
unknown smoothness	DIRECT or SOO	

Comparison

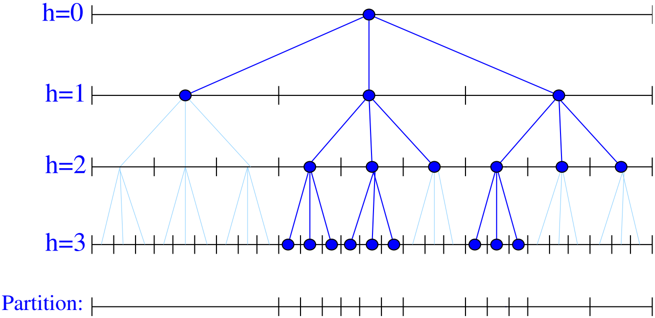
	Deterministic function	**Stochastic** function
known smoothness	DOO	Zooming or HOO
unknown smoothness	DIRECT or SOO	***StoSOO*** this talk

How it works?



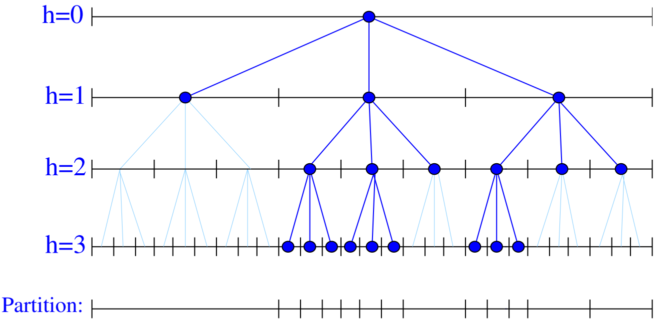
- `StoSOO` iteratively traverses and builds a tree over $\mathcal{X}$

How it works?



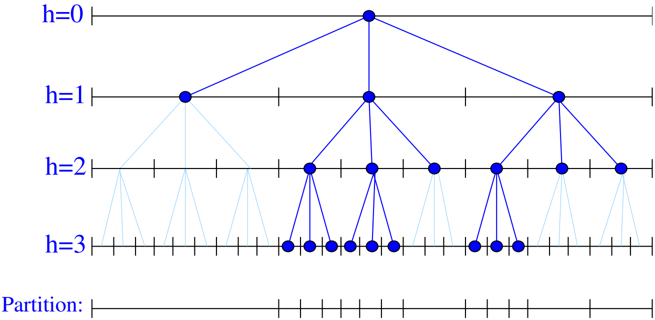
- StoSOO iteratively traverses and builds a tree over $\mathcal{X}$
- at each traversal it selects several nodes **simultaneously**

How it works?



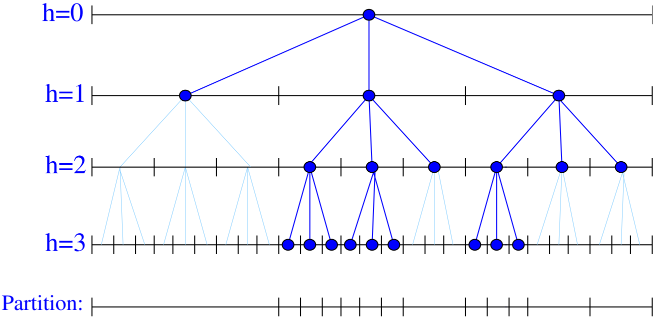
- StoSOO iteratively traverses and builds a tree over $\mathcal{X}$
- at each traversal it selects several nodes **simultaneously**
- **simultaneous** selection to consider all the leaves that can lead to potentially optimal solution

How it works?



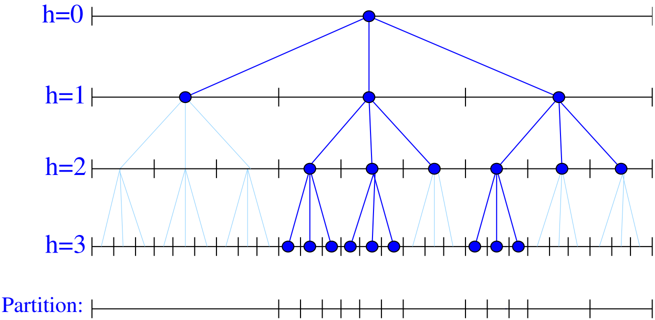
- selected nodes are either **sampled** or **expanded**

How it works?



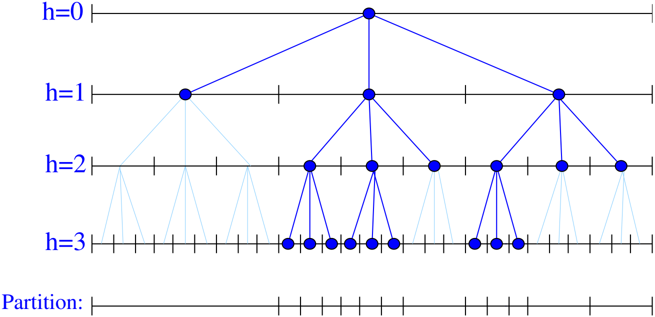
- selected nodes are either **sampled** or **expanded**
- **sample** a leaf k times for a confident estimate of $f(x_{h,i})$

How it works?



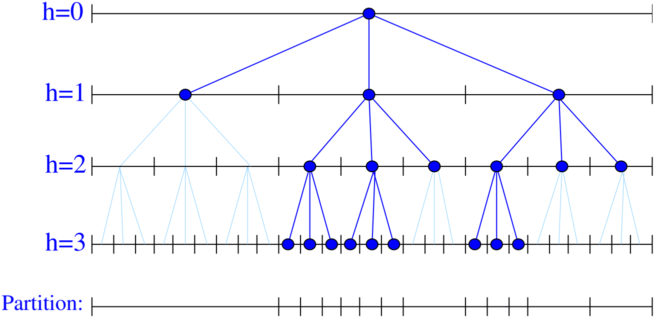
- selected nodes are either **sampled** or **expanded**
- **sample** a leaf k times for a confident estimate of $f(x_{h,i})$
- after sampling a leaf k times, we **expand** it

How it works?



- the selection is **optimistic**, based on confidence bounds

How it works?



- the selection is **optimistic**, based on confidence bounds
- return the deepest **expanded** node

Dealing with stochasticity

- evaluation of f at a point x_t returns a **noisy estimate** r_t,

$$\mathbb{E}[r_t|x_t] = f(x_t)$$

Dealing with stochasticity

- evaluation of f at a point x_t returns a **noisy estimate** r_t,

$$\mathbb{E}[r_t | x_t] = f(x_t)$$

- **approach:** sample each point several (k - parameter) times to obtain an accurate estimate *before* the node is expanded

Dealing with stochasticity

- evaluation of f at a point x_t returns a **noisy estimate** r_t,

$$\mathbb{E}[r_t|x_t] = f(x_t)$$

- **approach:** sample each point several (k - parameter) times to obtain an accurate estimate *before* the node is expanded

$$b_{h,j}(t) \stackrel{\text{def}}{=} \hat{\mu}_{h,j}(t) + \sqrt{\frac{\log(n^2/\delta)}{2T_{h,j}(t)}}$$

where $T_{h,j}(t)$ is the number of times (h,j) has been selected up to time t, and $\hat{\mu}_{h,j}(t)$ is the empirical average of rewards

Dealing with stochasticity

- evaluation of f at a point x_t returns a **noisy estimate** r_t,

$$\mathbb{E}[r_t|x_t] = f(x_t)$$

- **approach:** sample each point several (k - parameter) times to obtain an accurate estimate *before* the node is expanded

$$b_{h,j}(t) \stackrel{\text{def}}{=} \hat{\mu}_{h,j}(t) + \sqrt{\frac{\log(n^2/\delta)}{2T_{h,j}(t)}}$$

where $T_{h,j}(t)$ is the number of times (h,j) has been selected up to time t, and $\hat{\mu}_{h,j}(t)$ is the empirical average of rewards

- **optimistically** select the node with the highest b-value at each depth

Pseudocode of StoSOO

```
while t ≤ n do
  Set b_max = −∞.
  for h = 0 to maximum depth do
    Among all leaves (h, j) ∈ L_t of depth h, select
    (h, i) ∈ arg max_{(h,j)∈L_t} b_{h,j}(t)
    if b_{h,i}(t) ≥ b_max then
      Sample state x_t = x_{h,i} and collect reward r_t
      if T_{h,i}(t) ≥ k then
        Expand this node: add to T_t the K children of (h, i)
        Set b_max = b_{h,i}(t).
        Set t ← t + 1.
      end if
    end if
  end for
end while
```

Return the state corresponding to the deepest expanded node:

$$x(n) = \operatorname*{arg\,max}_{x_{h,j}:(h,j)\in\mathcal{T}_n\setminus\mathcal{L}_n} h.$$

Pseudocode of StoSOO

```
while t ≤ n do
  Set b_max = −∞.
  for h = 0 to maximum depth do
    Among all leaves (h, j) ∈ L_t of depth h, select
    (h, i) ∈ arg max_{(h,j)∈L_t} b_{h,j}(t)
    if b_{h,i}(t) ≥ b_max then
      Sample state x_t = x_{h,i} and collect reward r_t
      if T_{h,i}(t) ≥ k then
        Expand this node: add to T_t the K children of (h, i)
        Set b_max = b_{h,i}(t).
        Set t ← t + 1.
      end if
    end if
  end for
end while
```

Return the state corresponding to the deepest expanded node:

$$x(n) = \arg\max_{x_{h,j}:(h,j)\in\mathcal{T}_n\setminus\mathcal{L}_n} h.$$

Pseudocode of StoSOO

```
while t ≤ n do
   Set b_max = −∞.
   for h = 0 to maximum depth do
      Among all leaves (h, j) ∈ L_t of depth h, select
      (h, i) ∈ arg max_{(h,j)∈L_t} b_{h,j}(t)
      if b_{h,i}(t) ≥ b_max then
         Sample state x_t = x_{h,i} and collect reward r_t
         if T_{h,i}(t) ≥ k then
            Expand this node: add to T_t the K children of (h, i)
            Set b_max = b_{h,i}(t).
            Set t ← t + 1.
         end if
      end if
   end for
end while
```

Return the state corresponding to the deepest expanded node:

$$x(n) = \arg\max_{x_{h,j}:(h,j)\in\mathcal{T}_n\setminus\mathcal{L}_n} h.$$

Pseudocode of StoSOO

```
while t ≤ n do
    Set b_max = −∞.
    for h = 0 to maximum depth do
        Among all leaves (h, j) ∈ L_t of depth h, select
        (h, i) ∈ arg max_{(h,j)∈L_t} b_{h,j}(t)
        if b_{h,i}(t) ≥ b_max then
            Sample state x_t = x_{h,i} and collect reward r_t
            if T_{h,i}(t) ≥ k then
                Expand this node: add to T_t the K children of (h, i)
                Set b_max = b_{h,i}(t).
                Set t ← t + 1.
            end if
        end if
    end for
end while
```

Return the state corresponding to the deepest expanded node:

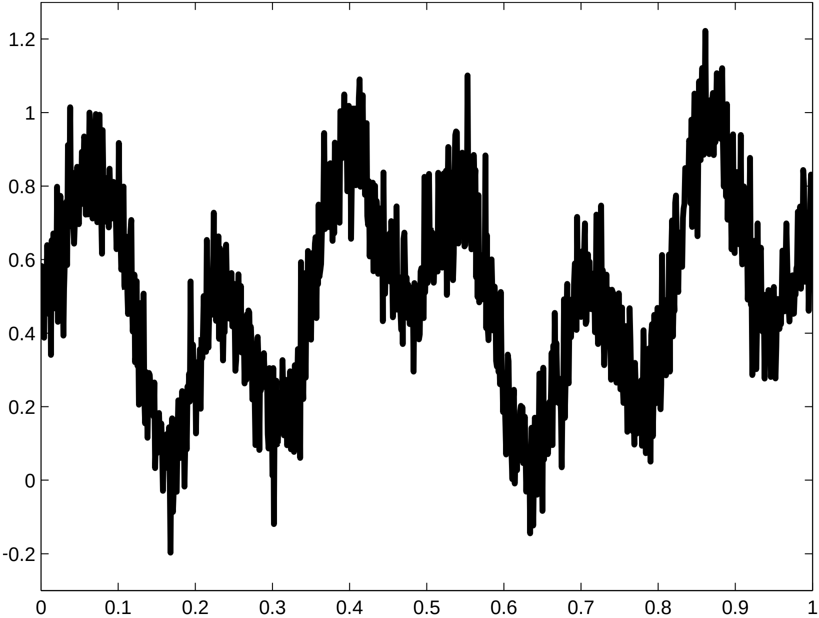
$$x(n) = \arg\max_{x_{h,j}:(h,j)\in\mathcal{T}_n\setminus\mathcal{L}_n} h.$$

Pseudocode of StoSOO

while $t \leq n$ **do**
  Set $b_{\max} = -\infty$.
  for $h = 0$ to maximum depth **do**
    Among all leaves $(h, j) \in \mathcal{L}_t$ of depth h, select
    $(h, i) \in \arg\max_{(h,j)\in\mathcal{L}_t} b_{h,j}(t)$
    if $b_{h,i}(t) \geq b_{\max}$ **then**
      Sample state $x_t = x_{h,i}$ and collect reward r_t
      if $T_{h,i}(t) \geq k$ **then**
        Expand this node: add to $\mathcal{T}_t$ the K children of (h, i)
        Set $b_{\max} = b_{h,i}(t)$.
        Set $t \leftarrow t + 1$.
      end if
    end if
  end for
end while

Return the state corresponding to the deepest expanded node:

$$x(n) = \arg\max_{x_{h,j}:(h,j)\in\mathcal{T}_n\setminus\mathcal{L}_n} h.$$


1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

Measure of complexity

For any $\varepsilon > 0$, write the set of ε-optimal states:

$$\mathcal{X}_\varepsilon \stackrel{\text{def}}{=} \{x \in \mathcal{X}, f(x) \geq f^* - \epsilon\}$$

Definition (near-optimality dimension)

Smallest constant d such that there exists $C > 0$, for all $\varepsilon > 0$, the packing number of $\mathcal{X}_\varepsilon$ with ℓ-balls of radius $\nu\varepsilon$ is less than $C\varepsilon^{-d}$.

Measure of complexity

For any $\varepsilon > 0$, write the set of ε-optimal states:

$$\mathcal{X}_\varepsilon \stackrel{\text{def}}{=} \{x \in \mathcal{X}, f(x) \geq f^* - \epsilon\}$$

Definition (near-optimality dimension)

Smallest constant d such that there exists $C > 0$, for all $\varepsilon > 0$, the packing number of $\mathcal{X}_\varepsilon$ with ℓ-balls of radius $\nu\varepsilon$ is less than $C\varepsilon^{-d}$.

- d depends both on the function and the metric

Measure of complexity

For any $\varepsilon > 0$, write the set of ε-optimal states:

$$\mathcal{X}_\varepsilon \stackrel{\text{def}}{=} \{x \in \mathcal{X}, f(x) \geq f^* - \epsilon\}$$

Definition (near-optimality dimension)

Smallest constant d such that there exists $C > 0$, for all $\varepsilon > 0$, the packing number of $\mathcal{X}_\varepsilon$ with ℓ-balls of radius $\nu\varepsilon$ is less than $C\varepsilon^{-d}$.

- d depends both on the function and the metric
- functions with smaller d are easier to optimize

Measure of complexity

For any $\varepsilon > 0$, write the set of ε-optimal states:

$$\mathcal{X}_\varepsilon \stackrel{\text{def}}{=} \{x \in \mathcal{X}, f(x) \geq f^* - \epsilon\}$$

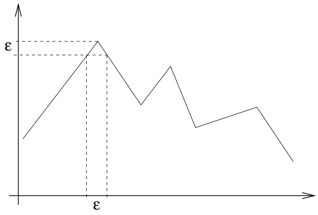
Definition (near-optimality dimension)

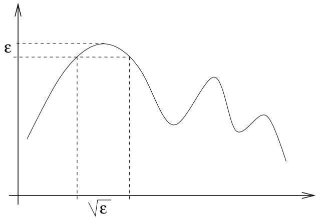
Smallest constant d such that there exists $C > 0$, for all $\varepsilon > 0$, the packing number of $\mathcal{X}_\varepsilon$ with ℓ-balls of radius $\nu\varepsilon$ is less than $C\varepsilon^{-d}$.

- d depends both on the function and the metric
- functions with smaller d are easier to optimize
- $d = 0$ covers a large class of functions already

Measure of complexity: Examples

$f(x^*) - f(x) = \Theta(||x^* - x||)$ $f(x^*) - f(x) = \Theta(||x^* - x||^2)$





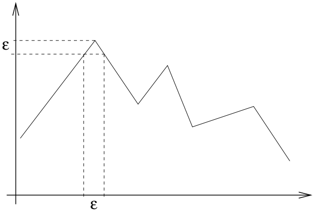
$\ell(x, y) = ||x - y|| \rightarrow d = 0$

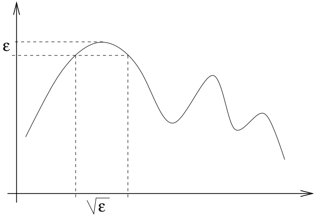
$\ell(x, y) = ||x - y|| \rightarrow d = D/2$
$\ell(x, y) = ||x - y||^2 \rightarrow d = 0$

Measure of complexity: Examples

StoSOO performs as if it knew the best possible semi-metric ℓ

$f(x^*) - f(x) = \Theta(||x^* - x||)$ $f(x^*) - f(x) = \Theta(||x^* - x||^2)$





$\ell(x,y) = ||x-y|| \to d = 0$

$\ell(x,y) = ||x-y|| \to d = D/2$
$\ell(x,y) = ||x-y||^2 \to d = 0$

Main result

Theorem

Let d be the $\nu/3$-near-optimality dimension and C be the corresponding constant. If the assumptions hold, then the loss of `StoSOO` *run with parameters k, $h_{\max}$, and $\delta > 0$, after n iterations is bounded, with probability $1-\delta$, as:*

$$R_n \leq 2\varepsilon + w\left(\min\left(h(n) - 1, h_\varepsilon, h_{\max}\right)\right)$$

where $\varepsilon = \sqrt{\log(nk/\delta)/(2k)}$ and $h(n)$ is the smallest $h \in \mathbb{N}$, such that:

$$C(k+1)h_{\max}\sum_{l=0}^{h}\left(w\left(l\right) + 2\varepsilon\right)^{-d} \geq n,$$

$h_\varepsilon = \arg\min\{h \in \mathbb{N} : w(h+1) < \varepsilon\}$ *and* $\sup_{x \in X_{h,i}} \ell(x_{h,i}, x) \leq w(h)$

Exponential diameters and $d = 0$

Corollary

Assume that the diameters of the cells decrease exponentially fast, i.e., $w(h) = c\gamma^h$ for some $c > 0$ and $\gamma < 1$. Assume that the $\nu/3$-near-optimality dimension is $d = 0$ and let C be the corresponding constant. Then the expected loss of StoSOO *run with parameters k, $h_{\max} = \sqrt{n/k}$, and $\delta > 0$, is bounded as:*

$$\mathbb{E}[R_n] \leq (2 + 1/\gamma)\varepsilon + c\gamma^{\sqrt{n/k}\,\min\{0.5/C, 1\} - 2} + 2\delta.$$

Exponential diameters and $d = 0$

Corollary

For the choice $k = n/\log^3(n)$ and $\delta = 1/\sqrt{n}$, we have:

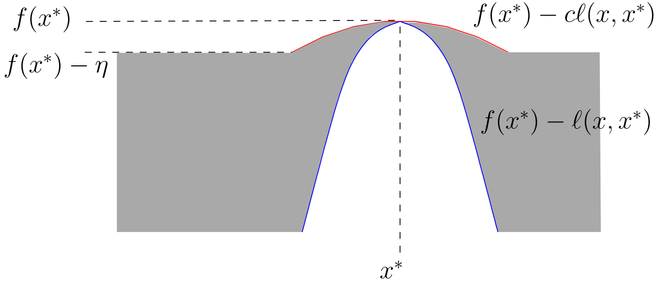
$$\mathbb{E}[R_n] = O\left(\frac{\log^2(n)}{\sqrt{n}}\right).$$

This result shows that, surprisingly, `StoSOO` can achieve the same rate $\tilde{O}(n^{-1/2})$, up to a logarithmic factor, as the HOO or Stochastic DOO algorithms run with the best possible metric, although `StoSOO` does not require the knowledge of it.

The important case $d = 0$

Let a function in such space have upper- and lower envelope around x^* of the same order, i.e., there exists constants $c \in (0, 1)$, and $\eta > 0$, such that for all $x \in \mathcal{X}$:

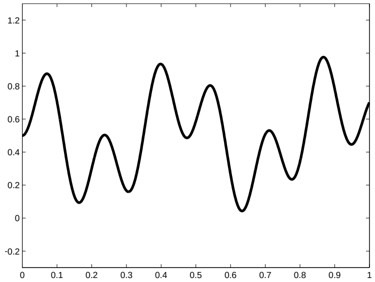
$$\min(\eta, c\ell(x, x^*)) \leq f(x^*) - f(x) \leq \ell(x, x^*). \quad (1)$$



Any function satisfying (1) lies in the gray area and possesses a lower- and upper-envelopes that are of same order around x^*.

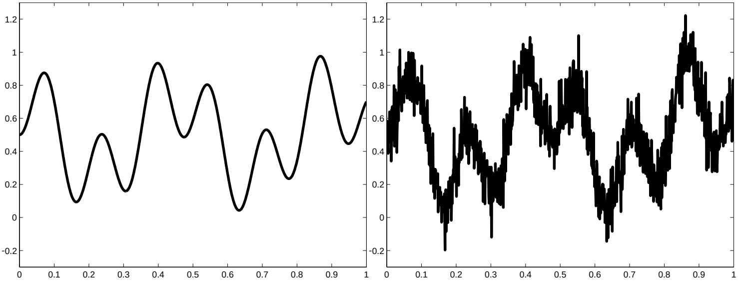
Two-sine product function

$$f_1(x) = \frac{1}{2}\sin(13x)\cdot\sin(27x)$$



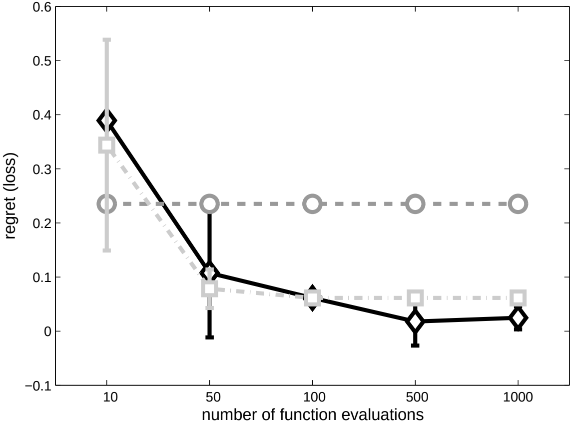
Two-sine product function

$$f_1(x) = \frac{1}{2}\sin(13x)\cdot\sin(27x) + \mathcal{N}_{\mathcal{T}}(0.5, 0.1)$$



Two-sine product function

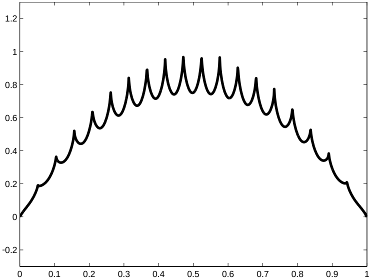
$$f_1(x) = \frac{1}{2}\sin(13x)\cdot\sin(27x) + \mathcal{N}_{\mathcal{T}}(0.5, 0.1)$$



`StoSOO` (diamonds) vs.
Stochastic DOO with ℓ_1 (circles) and ℓ_2 (squares) on f_1

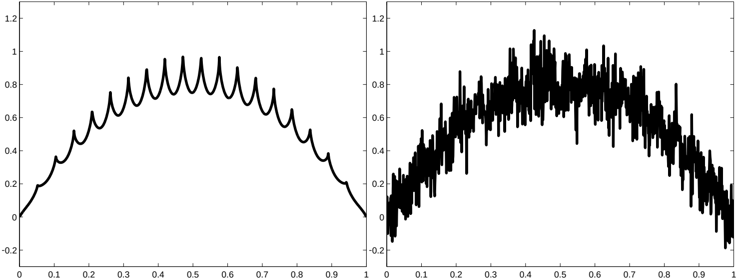
Garland function

$$f_2(x) = 4x(1-x)\cdot\left(\frac{3}{4}+\frac{1}{4}\left(1-\sqrt{|\sin(60x)|}\right)\right).$$



Garland function

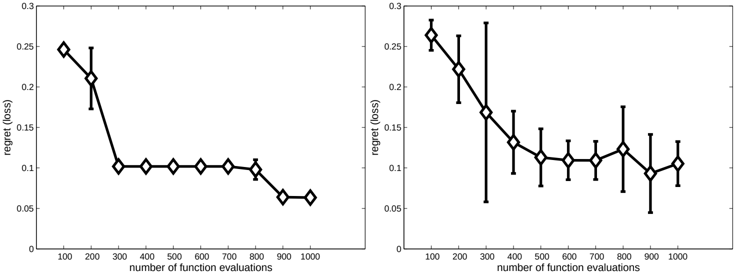
$$f_2(x) = 4x(1-x)\cdot\left(\frac{3}{4}+\frac{1}{4}\left(1-\sqrt{|\sin(60x)|}\right)\right).$$



Not Lipschitz for any L!

Garland function

$$f_2(x) = 4x(1-x) \cdot \left(\frac{3}{4} + \frac{1}{4}\left(1 - \sqrt{|\sin(60x)|}\right)\right).$$



StoSOO's performance for the garland function.
Left noised with $\mathcal{N}_T(0, 0.01)$. **Right**: Noised with $\mathcal{N}_T(0, 0.1)$.

Conclusion

- StoSOO - a black-box stochastic function optimizer
- StoSOO does not need to know the smoothness

Conclusion

- StoSOO - a black-box stochastic function optimizer
- StoSOO does not need to know the smoothness
- Weak assumptions, efficient for low-dimensional problems

Conclusion

- StoSOO - a black-box stochastic function optimizer
- StoSOO does not need to know the smoothness
- Weak assumptions, efficient for low-dimensional problems
- Finite-time performance analysis for $d = 0$

Conclusion

- StoSOO - a black-box stochastic function optimizer
- StoSOO does not need to know the smoothness
- Weak assumptions, efficient for low-dimensional problems
- Finite-time performance analysis for $d = 0$
- Performance as good as as with the best valid semi-metric
- Code: HTTPS://SEQUEL.LILLE.INRIA.FR/SOFTWARE/STOSOO

Conclusion

- StoSOO - a black-box stochastic function optimizer
- StoSOO does not need to know the smoothness
- Weak assumptions, efficient for low-dimensional problems
- Finite-time performance analysis for $d = 0$
- Performance as good as as with the best valid semi-metric
- Code: https://sequel.lille.inria.fr/Software/StoSOO

Thank you!

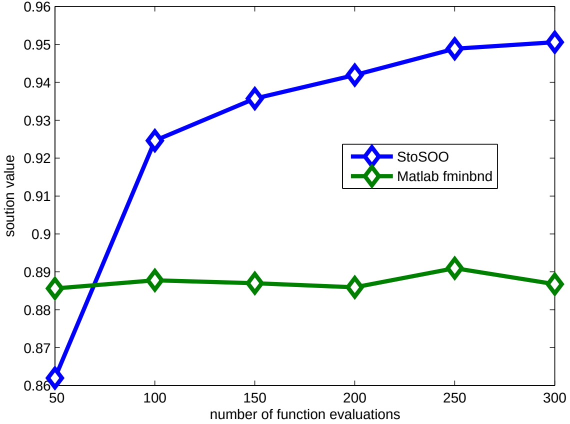
Supported by European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n^{o} 270327 (project CompLACS).

Michal Valko

michal.valko@inria.fr

sequel.lille.inria.fr

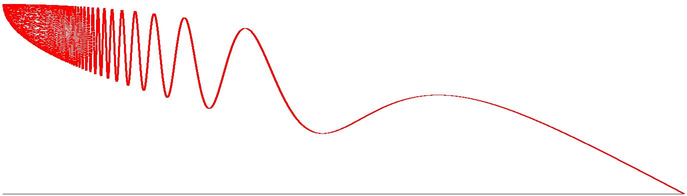
Noised two-sine product function: StoSOO vs. MATLAB



When $d > 0$?

Example of a function with different order in the upper and lower envelopes, when $\ell(x, y) = |x - y|^\alpha$:

$$f(x) = 1 - \sqrt{x} + (-x^2 + \sqrt{x}) \cdot (\sin(1/x^2) + 1)/2$$



The lower-envelope behaves like a square root whereas the upper one is quadratic. There is no semi-metric of the form $|x - y|^\alpha$ for which $d < 3/2$.