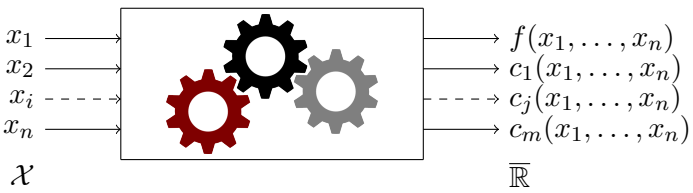
Sondes locales intensives lors de l'exécution de MADS dans un environnement parallèle

Guillaume LAMEYNARDIE
Directeur : Charles AUDET, Co-directeur : Sébastien LE DIGABEL
Supervisé à l'IREQ par : Stéphane ALARIE

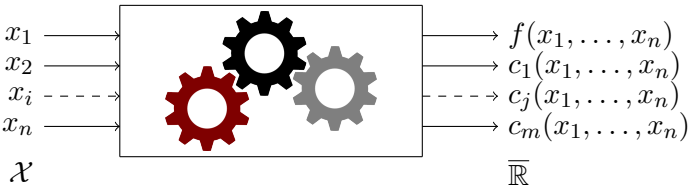
Plan

Introduction

L'optimisation de boites noires

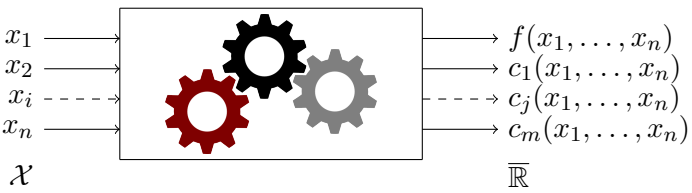


L'optimisation de boites noires



- $x \in \mathcal{X} \subset \mathbb{R}^n$.
- f et $c_j : \mathcal{X} \to \mathbb{R} \cup \{+\infty\} = \overline{\mathbb{R}}$.
- On cherche $x^* \in \mathcal{X}$ qui retourne la plus petite valeur de f tel que $c_j(x^*) \leq 0$ pour tout j.

L'optimisation de boites noires

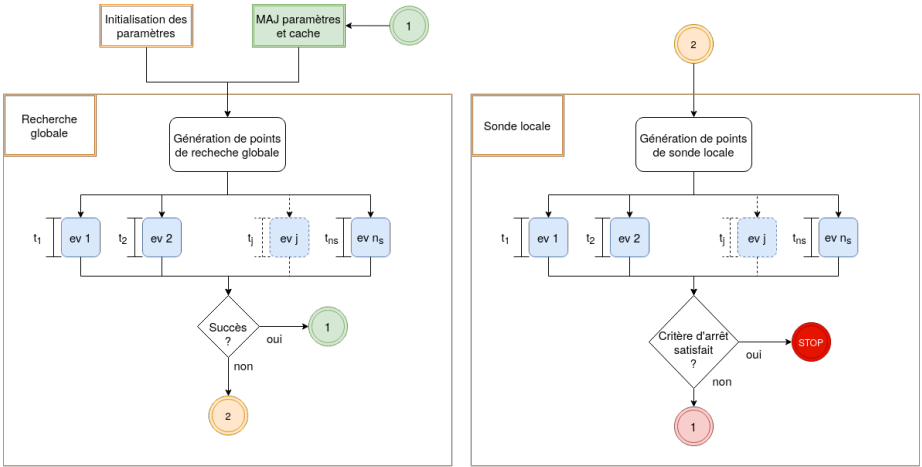


- Calculer une fois f et c_j est coûteux en temps et en ressources.
- Les dérivées ne sont pas accessibles, et même si elles existent (ce qui n'est pas garanti) elles ne seraient d'aucune utilité.
- f et c_j sont déterministes.

L'optimisation de boite noire

$$\mathcal{P} : \begin{cases} \min\limits_{x \in \mathcal{X}} & f(x) \\ \text{s.c} & c_j(x) \leq 0, \quad j \in [\![1, m]\!]. \end{cases}$$

L'algorithme MADS [Audet and Dennis, Jr. (2006)]



L'algorithme MADS [Audet and Dennis, Jr. (2006)]

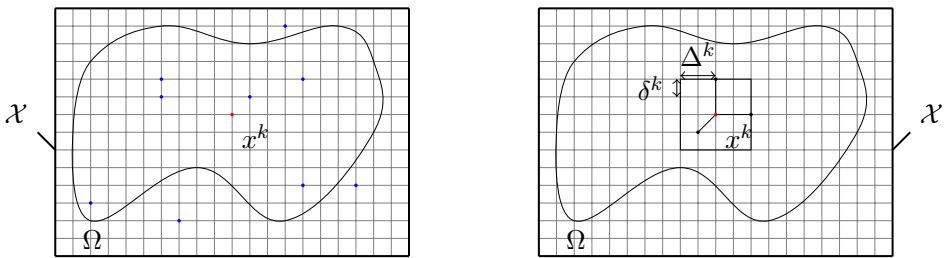


FIGURE – Si l'étape de SEARCH (a) ne parvient pas à améliorer f, on effectue l'étape de POLL (b).

L'algorithme MADS [Audet and Dennis, Jr. (2006)]

(a) (b)

FIGURE – Si l'étape de POLL ne parvient pas à améliorer la valeur de f (a) alors δ^k et Δ^k sont diminués (b).

Les règles de génération de points dans MADS

Dans la SEARCH :

- N'importe où sur le maillage.
- En un nombre fini.

Les règles de génération de points dans MADS

Dans la POLL :

- Sur le maillage, à une distance maximale de Δ^k de la meilleure solution connue x^k.
- les directions entre les points de sonde et x^k doivent former un ensemble générateur positif.

Exploitation de ressources parallèles

Définition (Processeur)

Pour une boite noire et un utilisateur donné, un *processeur* est un ensemble de ressources que l'utilisateur juge *a priori* capable de mener à terme n'importe laquelle des évaluations de cette boite noire en une durée inférieure à une borne fixée par ce même utilisateur.

Exploitation de ressources parallèles

Définition (Processeur)

Pour une boite noire et un utilisateur donné, un *processeur* est un ensemble de ressources que l'utilisateur juge *a priori* capable de mener à terme n'importe laquelle des évaluations de cette boite noire en une durée inférieure à une borne fixée par ce même utilisateur.

- processeur $\neq$ CPU : puce électronique dans un ordinateur.

Exploitation de ressources parallèles

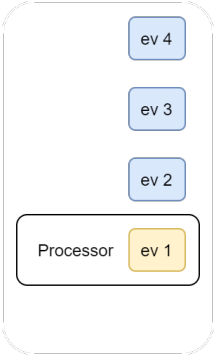


FIGURE – Pile d'évaluation séquentielle.

Exploitation de ressources parallèles

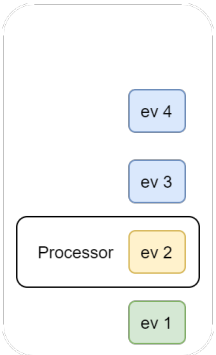


FIGURE – Pile d'évaluation séquentielle.

Exploitation de ressources parallèles

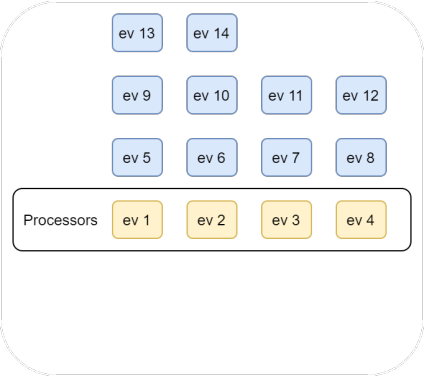


FIGURE – Pile d'évaluation parallèle avec plus d'évaluations que de processeurs.

Exploitation de ressources parallèles

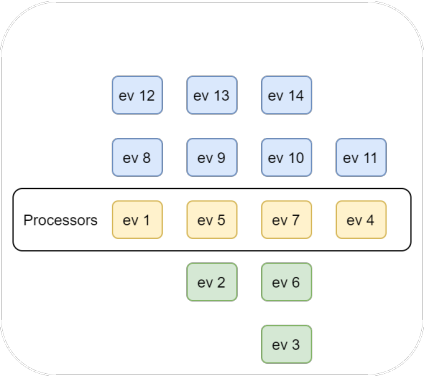


FIGURE – Pile d'évaluation parallèle avec plus d'évaluations que de processeurs.

Exploitation de ressources parallèles

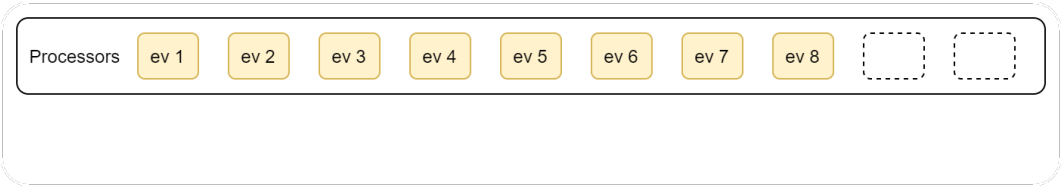


FIGURE – Pile d'évaluation parallèle avec plus de processeurs que d'évaluations.

Exploitation de ressources parallèles

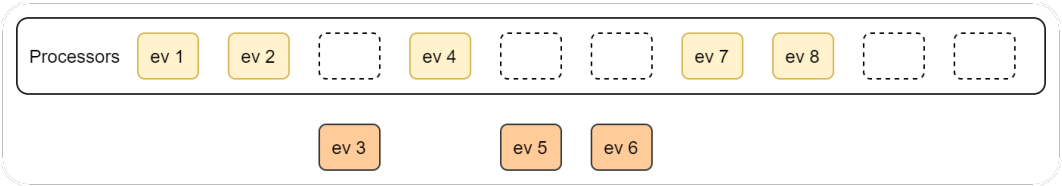


FIGURE – Pile d'évaluation parallèle avec plus de processeurs que d'évaluations.

Motivations

Motivations

- Les environnements à fort degré de parallélisme (clusters, cloud computing) sont de plus en plus accessibles.
- Des problèmes avec un faible nombre de variables existent toujours.
- On espère qu'avec plus de ressources on puisse résoudre des problèmes plus efficacement.

Génération d'un grand nombre de points : À l'étape de SEARCH

Dans la SEARCH

- Talgorn et al. (2020).
- Effectue une optimisation sur un modèle dynamique du problème.
- Filtre les points générés à l'aide de six métriques différentes.
- Évalue les points filtrés sur la boite noire.

Dans la SEARCH

$$(\mathcal{S}) : \begin{cases} \min\limits_{x \in \mathcal{X}} & \tilde{f}(x) \\ \text{s.c} & \tilde{c_j}(x) \leq 0, \quad j \in [\![1, m]\!]. \end{cases}$$

est un modèle de $(\mathcal{P})$ si il partage certaines similarités avec $(\mathcal{P})$.

- Plus facile/rapide à résoudre que $(\mathcal{P})$

Dans la SEARCH

Algorithme 1 Génération de points pour la SEARCH

[0]. Initialisation :
- M^k : Maillage à l'itération k de MADS
- $\mathbb{S}^k$: Ensemble de points de SEARCH
- $T \leftarrow \varnothing$.
- $\tilde{V} \leftarrow \varnothing$.
- $\mathcal{F}_i : i \in [\![1, 6]\!]$: stratégies de filtrage
- $i \leftarrow 1$.
- Choisir $t_{max} \in \mathbb{N}^*$: nb. max. de points à générer.
- Choisir V un ensemble de points déjà évalués.

[1]. Génération de $\tilde{V}$:
- Construire $(\mathcal{S})$ avec l'information donnée par V.
- Résoudre $(\mathcal{S})$ et stocker les points évalués dans $\tilde{V}$.
- Aller à [2].

[2]. Filtrage des points de $\tilde{V}$
- **Tant que** $\mathcal{F}_i(\tilde{V})$ retourne un point $\tilde{s}$ **faire** :
  - **Si** $|T| < t_{max}$ **faire** :
    - $T \leftarrow T \cup \{\tilde{s}\}$.
  - **Sinon** :
    - Aller à [3].
- **Sinon** :
  - $i \leftarrow i + 1$.
  - Aller à [2].

[3]. Projection sur le maillage
- **Pour** $\tilde{s} \in T$ **faire** :
  - Projeter $\tilde{s}$ sur M^k pour obtenir s.
  - $\mathbb{S}^k \leftarrow \mathbb{S}^k \cup \{s\}$

On utilise V pour construire

$$(\mathcal{S}) : \begin{cases} \min\limits_{x \in \mathcal{X}} & \tilde{f}(x), \\ \text{s.c} & \tilde{c}_j(x) \leq 0, \quad j \in [\![1, m]\!]. \end{cases}$$

Dans la SEARCH

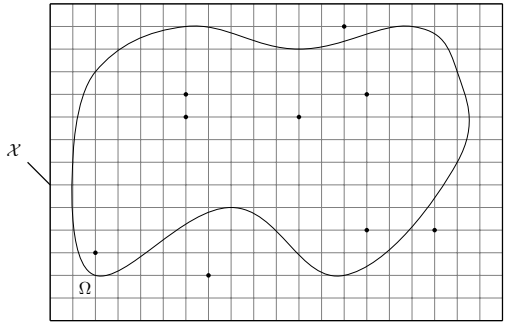
Algorithme 2 Génération de points pour la SEARCH

[0]. Initialisation :
- M^k : Maillage à l'itération k de MADS
- $\mathbb{S}^k$: Ensemble de points de SEARCH
- $T \leftarrow \varnothing$.
- $\tilde{V} \leftarrow \varnothing$.
- $\mathcal{F}_i : i \in [\![1, 6]\!]$: stratégies de filtrage
- $i \leftarrow 1$.
- Choisir $t_{max} \in \mathbb{N}^*$: nb. max. de points à générer.
- Choisir V un ensemble de points déjà évalués.

[1]. Génération de $\tilde{V}$:
- Construire $(\mathcal{S})$ avec l'information donnée par V.
- Résoudre $(\mathcal{S})$ et stocker les points évalués dans $\tilde{V}$.
- Aller à [2].

[2]. Filtrage des points de $\tilde{V}$
- **Tant que** $\mathcal{F}_i(\tilde{V})$ retourne un point $\tilde{s}$ **faire** :
  - **Si** $|T| < t_{max}$ **faire** :
    - $T \leftarrow T \cup \{\tilde{s}\}$.
  - **Sinon** :
    - Aller à [3].
- **Sinon** :
  - $i \leftarrow i + 1$.
  - Aller à [2].

[3]. Projection sur le maillage
- **Pour** $\tilde{s} \in T$ **faire** :
  - Projeter $\tilde{s}$ sur M^k pour obtenir s.
  - $\mathbb{S}^k \leftarrow \mathbb{S}^k \cup \{s\}$



Dans la SEARCH

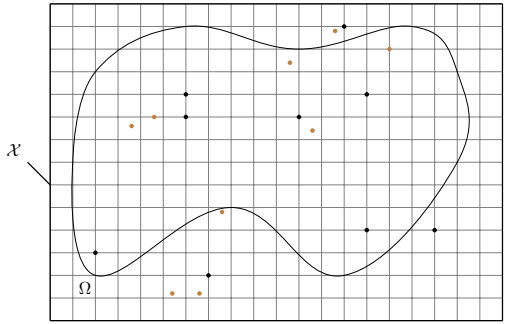
Algorithme 3 Génération de points pour la SEARCH

[0]. Initialisation :
- M^k : Maillage à l'itération k de MADS
- $\mathbb{S}^k$: Ensemble de points de SEARCH
- $T \leftarrow \varnothing$.
- $\tilde{V} \leftarrow \varnothing$.
- $\mathcal{F}_i : i \in [\![1, 6]\!]$: stratégies de filtrage
- $i \leftarrow 1$.
- Choisir $t_{max} \in \mathbb{N}^*$: nb. max. de points à générer.
- Choisir V un ensemble de points déjà évalués.

[1]. Génération de $\tilde{V}$:
- Construire $(\mathcal{S})$ avec l'information donnée par V.
- Résoudre $(\mathcal{S})$ et stocker les points évalués dans $\tilde{V}$.
- Aller à [2].

[2]. Filtrage des points de $\tilde{V}$
- **Tant que** $\mathcal{F}_i(\tilde{V})$ retourne un point $\tilde{s}$ **faire** :
  - **Si** $|T| < t_{max}$ **faire** :
    - $T \leftarrow T \cup \{\tilde{s}\}$.
  - **Sinon** :
    - Aller à [3].
- **Sinon** :
  - $i \leftarrow i + 1$.
  - Aller à [2].

[3]. Projection sur le maillage
- **Pour** $\tilde{s} \in T$ **faire** :
  - Projeter $\tilde{s}$ sur M^k pour obtenir s.
  - $\mathbb{S}^k \leftarrow \mathbb{S}^k \cup \{s\}$



Dans la SEARCH

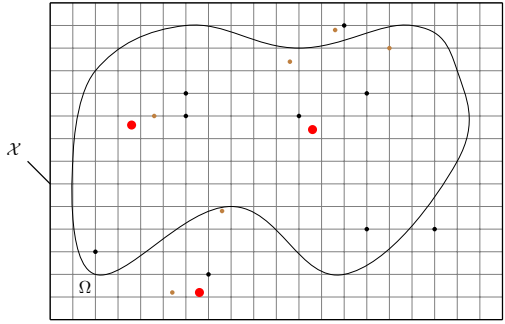
Algorithme 4 Génération de points pour la SEARCH

[0]. Initialisation :
- M^k : Maillage à l'itération k de MADS
- $\mathbb{S}^k$: Ensemble de points de SEARCH
- $T \leftarrow \varnothing$.
- $\tilde{V} \leftarrow \varnothing$.
- $\mathcal{F}_i : i \in [\![1, 6]\!]$: stratégies de filtrage
- $i \leftarrow 1$.
- Choisir $t_{max} \in \mathbb{N}^*$: nb. max. de points à générer.
- Choisir V un ensemble de points déjà évalués.

[1]. Génération de $\tilde{V}$:
- Construire $(\mathcal{S})$ avec l'information donnée par V.
- Résoudre $(\mathcal{S})$ et stocker les points évalués dans $\tilde{V}$.
- Aller à [2].

[2]. Filtrage des points de $\tilde{V}$
- **Tant que** $\mathcal{F}_i(\tilde{V})$ retourne un point $\tilde{s}$ **faire** :
  - **Si** $|T| < t_{max}$ **faire** :
    - $T \leftarrow T \cup \{\tilde{s}\}$.
  - **Sinon** :
    - Aller à [3].
- **Sinon** :
  - $i \leftarrow i + 1$.
  - Aller à [2].

[3]. Projection sur le maillage
- **Pour** $\tilde{s} \in T$ **faire** :
  - Projeter $\tilde{s}$ sur M^k pour obtenir s.
  - $\mathbb{S}^k \leftarrow \mathbb{S}^k \cup \{s\}$



Dans la SEARCH

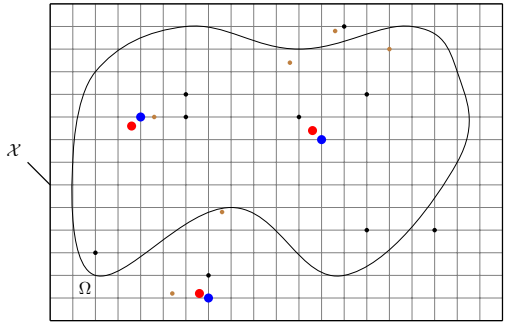
Algorithme 5 Génération de points pour la SEARCH

[0]. Initialisation :
- M^k : Maillage à l'itération k de MADS
- $\mathbb{S}^k$: Ensemble de points de SEARCH
- $T \leftarrow \varnothing$.
- $\tilde{V} \leftarrow \varnothing$.
- $\mathcal{F}_i : i \in [\![1, 6]\!]$: stratégies de filtrage
- $i \leftarrow 1$.
- Choisir $t_{max} \in \mathbb{N}^*$: nb. max. de points à générer.
- Choisir V un ensemble de points déjà évalués.

[1]. Génération de $\tilde{V}$:
- Construire $(\mathcal{S})$ avec l'information donnée par V.
- Résoudre $(\mathcal{S})$ et stocker les points évalués dans $\tilde{V}$.
- Aller à [2].

[2]. Filtrage des points de $\tilde{V}$
- **Tant que** $\mathcal{F}_i(\tilde{V})$ retourne un point $\tilde{s}$ **faire** :
  - **Si** $|T| < t_{max}$ **faire** :
    - $T \leftarrow T \cup \{\tilde{s}\}$.
  - **Sinon** :
    - Aller à [3].
- **Sinon** :
  - $i \leftarrow i + 1$.
  - Aller à [2].

[3]. Projection sur le maillage
- **Pour** $\tilde{s} \in T$ **faire** :
  - Projeter $\tilde{s}$ sur M^k pour obtenir s.
  - $\mathbb{S}^k \leftarrow \mathbb{S}^k \cup \{s\}$



Génération d'un grand nombre de points : À l'étape de POLL

Dans la POLL

- Trois stratégies : Multi POLL, POLL en Oignon, POLL Enrichie.
- Permet le dimensionnement en générant plus de $2n$ points à chaque étape de POLL.
- Diffèrent dans la manière dont sont générés les points.

Rappel : POLL classique

Algorithme 6 POLL classique - Génération de points

[0]. Initialisation :

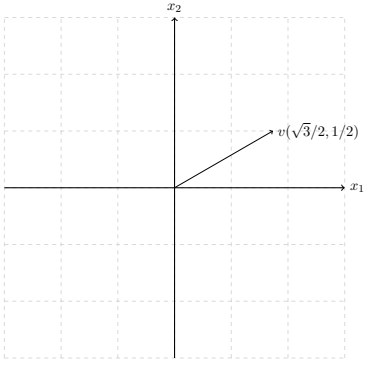
- Δ^k : Pas de cadre
- δ^k : Pas de maillage
- x^k : Point donnant lieu au succès courant
- $\mathbb{D} \leftarrow \varnothing$: Ensemble de direction de POLL
- $\mathbb{P} \leftarrow \varnothing$: Ensemble de points de POLL

[1]. Génération de directions de sonde :

- Sélectionner $u \in \mathbb{R}^n$ non nulle.
- Poser $v = \frac{u}{\|u\|}$.
- Poser $H = I - 2vv^\top$.
- $\mathbb{D} \leftarrow \left\{ \pm \delta^k \text{round}\left(\frac{\Delta^k}{\delta^k} \frac{h_j}{\|h_j\|_\infty} \right) \ : \ j \in [\![1, n]\!] \right\} \subset \delta^k \mathbb{Z}^n$.

[2]. Génération de points de sonde :

- $\mathbb{P} \leftarrow \{x^k + d \ : \ d \in \mathbb{D}\}$.
- **RETOURNER** $\mathbb{P}$.



Rappel : POLL classique

Algorithme 7 POLL classique - Génération de points

[0]. Initialisation :

- Δ^k : Pas de cadre
- δ^k : Pas de maillage
- x^k : Point donnant lieu au succès courant
- $\mathbb{D} \leftarrow \varnothing$: Ensemble de direction de POLL
- $\mathbb{P} \leftarrow \varnothing$: Ensemble de points de POLL

[1]. Génération de directions de sonde :

Sélectionner $u \in \mathbb{R}^n$ non nulle.

Poser $v = \frac{u}{\|u\|}$.

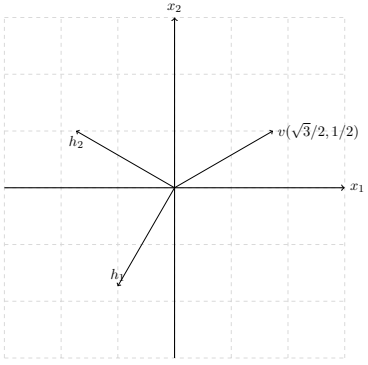
Poser $H = I - 2vv^\top$.

$$\mathbb{D} \leftarrow \left\{ \pm \delta^k \text{round}\left(\frac{\Delta^k}{\delta^k} \frac{h_j}{\|h_j\|_\infty} \right) \ : \ j \in [\![1, n]\!] \right\} \subset \delta^k \mathbb{Z}^n.$$

[2]. Génération de points de sonde :

$\mathbb{P} \leftarrow \{x^k + d \ : \ d \in \mathbb{D}\}$.

RETOURNER $\mathbb{P}$.



Rappel : POLL classique

Algorithme 8 POLL classique - Génération de points

[0]. Initialisation :

- Δ^k : Pas de cadre
- δ^k : Pas de maillage
- x^k : Point donnant lieu au succès courant
- $\mathbb{D} \leftarrow \varnothing$: Ensemble de direction de POLL
- $\mathbb{P} \leftarrow \varnothing$: Ensemble de points de POLL

[1]. Génération de directions de sonde :

Sélectionner $u \in \mathbb{R}^n$ non nulle.

Poser $v = \frac{u}{\|u\|}$.

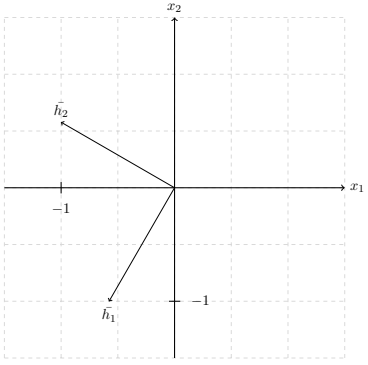
Poser $H = I - 2vv^\top$.

$$\mathbb{D} \leftarrow \left\{ \pm \delta^k \text{round}\left(\frac{\Delta^k}{\delta^k} \frac{h_j}{\|h_j\|_\infty} \right) \ : \ j \in [\![1, n]\!] \right\} \subset \delta^k \mathbb{Z}^n.$$

[2]. Génération de points de sonde :

$\mathbb{P} \leftarrow \{x^k + d \ : \ d \in \mathbb{D}\}$.

RETOURNER $\mathbb{P}$.



Rappel : POLL classique

Algorithme 9 POLL classique - Génération de points

[0]. Initialisation :

- Δ^k : Pas de cadre
- δ^k : Pas de maillage
- x^k : Point donnant lieu au succès courant
- $\mathbb{D} \leftarrow \varnothing$: Ensemble de direction de POLL
- $\mathbb{P} \leftarrow \varnothing$: Ensemble de points de POLL

[1]. Génération de directions de sonde :

Sélectionner $u \in \mathbb{R}^n$ non nulle.

Poser $v = \frac{u}{\|u\|}$.

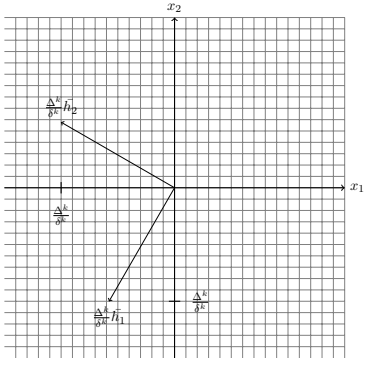
Poser $H = I - 2vv^\top$.

$$\mathbb{D} \leftarrow \left\{ \pm \delta^k \text{round}\left(\frac{\Delta^k}{\delta^k} \frac{h_j}{\|h_j\|_\infty} \right) \ : \ j \in [\![1, n]\!] \right\} \subset \delta^k \mathbb{Z}^n.$$

[2]. Génération de points de sonde :

$\mathbb{P} \leftarrow \{x^k + d \ : \ d \in \mathbb{D}\}$.

RETOURNER $\mathbb{P}$.



Rappel : POLL classique

Algorithme 10 POLL classique - Génération de points

[0]. Initialisation :

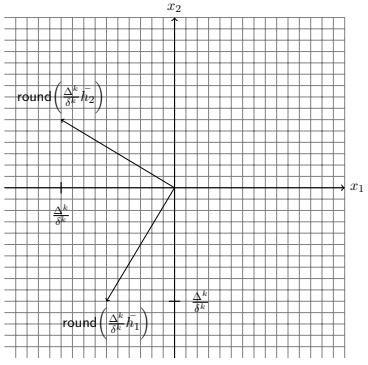
- Δ^k : Pas de cadre
- δ^k : Pas de maillage
- x^k : Point donnant lieu au succès courant
- $\mathbb{D} \leftarrow \varnothing$: Ensemble de direction de POLL
- $\mathbb{P} \leftarrow \varnothing$: Ensemble de points de POLL

[1]. Génération de directions de sonde :

- Sélectionner $u \in \mathbb{R}^n$ non nulle.
- Poser $v = \frac{u}{\|u\|}$.
- Poser $H = I - 2vv^\top$.
- $\mathbb{D} \leftarrow \left\{ \pm \delta^k \text{round}\left(\frac{\Delta^k}{\delta^k} \frac{h_j}{\|h_j\|_\infty} \right) \ : \ j \in [\![1, n]\!] \right\} \subset \delta^k \mathbb{Z}^n$.

[2]. Génération de points de sonde :

- $\mathbb{P} \leftarrow \{x^k + d \ : \ d \in \mathbb{D}\}$.
- **RETOURNER** $\mathbb{P}$.



Rappel : POLL classique

Algorithme 11 POLL classique - Génération de points

[0]. Initialisation :

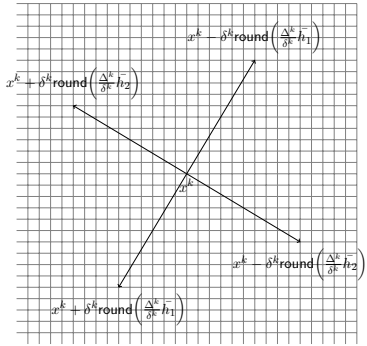
- Δ^k : Pas de cadre
- δ^k : Pas de maillage
- x^k : Point donnant lieu au succès courant
- $\mathbb{D} \leftarrow \varnothing$: Ensemble de direction de POLL
- $\mathbb{P} \leftarrow \varnothing$: Ensemble de points de POLL

[1]. Génération de directions de sonde :

- Sélectionner $u \in \mathbb{R}^n$ non nulle.
- Poser $v = \frac{u}{\|u\|}$.
- Poser $H = I - 2vv^\top$.
- $\mathbb{D} \leftarrow \left\{ \pm \delta^k \mathsf{round}\left(\frac{\Delta^k}{\delta^k} \frac{h_j}{\|h_j\|_\infty} \right) \ : \ j \in [\![1, n]\!] \right\} \subset \delta^k \mathbb{Z}^n$.

[2]. Génération de points de sonde :

- $\mathbb{P} \leftarrow \{x^k + d \ : \ d \in \mathbb{D}\}$.
- **RETOURNER** $\mathbb{P}$.



Rappel : POLL classique

Algorithme 12 POLL classique - Génération de points

[0]. Initialisation :

- Δ^k : Pas de cadre
- δ^k : Pas de maillage
- x^k : Point donnant lieu au succès courant
- $\mathbb{D} \leftarrow \varnothing$: Ensemble de direction de POLL
- $\mathbb{P} \leftarrow \varnothing$: Ensemble de points de POLL

[1]. Génération de directions de sonde :

Sélectionner $u \in \mathbb{R}^n$ non nulle.

Poser $v = \frac{u}{\|u\|}$.

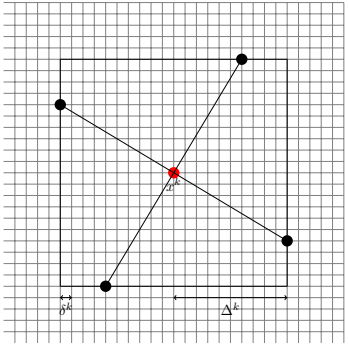
Poser $H = I - 2vv^\top$.

$$\mathbb{D} \leftarrow \left\{ \pm \delta^k \text{round}\left(\frac{\Delta^k}{\delta^k} \frac{h_j}{\|h_j\|_\infty} \right) \ : \ j \in [\![1, n]\!] \right\} \subset \delta^k \mathbb{Z}^n.$$

[2]. Génération de points de sonde :

$\mathbb{P} \leftarrow \{x^k + d \ : \ d \in \mathbb{D}\}$.

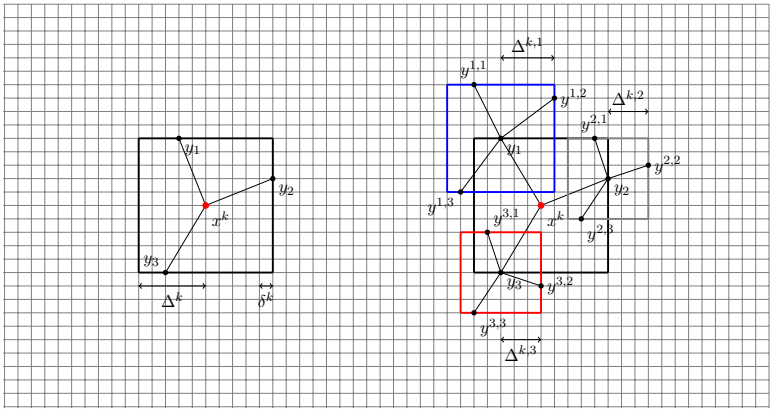
RETOURNER $\mathbb{P}$.



Rappel : POLL classique

$$\begin{aligned} \text{classicalPoll} : \mathbb{R}_+^* \times \mathbb{R}_+^* \times \mathbb{R}^n &\to \mathbb{R}^n \\ (\delta^k, \Delta^k, x^k) &\mapsto \mathbb{P}^k \end{aligned}$$

Multi POLL


y_1
y_2
x^k
y_3
Δ^k
δ^k
$\Delta^{k,1}$
$y^{1,1}$
$y^{1,2}$
$\Delta^{k,2}$
$y^{2,1}$
y_1
$y^{2,2}$
y_2
$y^{1,3}$
$y^{3,1}$
x^k
$y^{2,3}$
y_3
$y^{3,2}$
$y^{3,3}$
$\Delta^{k,3}$

Multi POLL

Algorithme 13 Multi-POLL - Génération de points

[0]. Initialisation :

- δ^k : Pas de maillage
- Δ^k : Pas de cadre
- x^k : Meilleur point connu
- $\mathbb{P} \leftarrow \varnothing$: Ensemble de points de sonde

[1]. Génération de points de sonde primaire :

$\mathbb{P} \leftarrow \text{CLASSICALPOLL}(\delta^k, \Delta^k, x^k)$

[2]. Génération de points de sonde secondaire :

$T \leftarrow \varnothing$.

Pour $y \in \mathbb{P}$ **faire** :

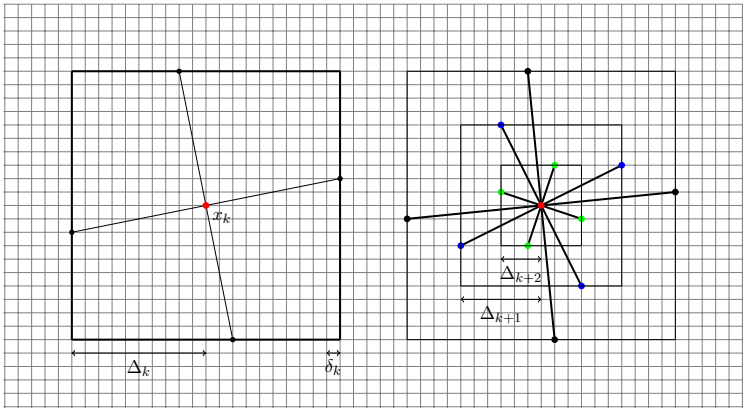
$T \leftarrow T \cup \text{CLASSICALPOLL}(\delta^k, \Delta^k, y)$.

Fin

$\mathbb{P} \leftarrow \mathbb{P} \cup T$.

RETOURNER $\mathbb{P}$.

POLL en oignon



POLL en oignon

Algorithme 14 POLL en oignon - Génération de points

[0]. Initialisation :

δ^k	: Pas de maillage.
Δ^k	: Pas de sonde.
x^k	: Meilleur point connu.
c	: Nombre de couches.
$\mathbb{P} \leftarrow \varnothing$	: Ensemble de points de sonde.

[1]. Génération des sous cadres :

Choisir $\Gamma \subseteq [\![1, \frac{\Delta^k}{\delta^k}]\!]$, $|\Gamma| \leq c$.

[2]. Génération de points de sonde :

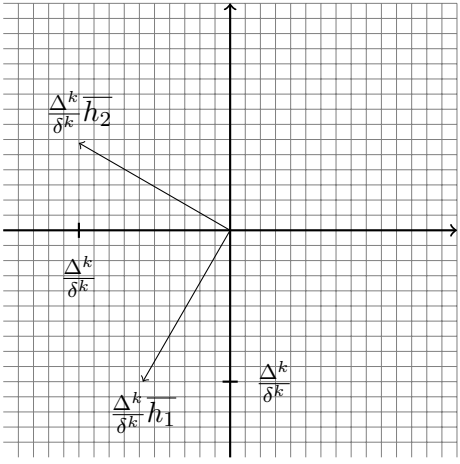
Pour $\gamma \in \Gamma$ **faire** :

$\mathbb{P} \leftarrow \mathbb{P} \cup \text{CLASSICALPOLL}(\delta^k, \gamma\delta^k, x^k)$.

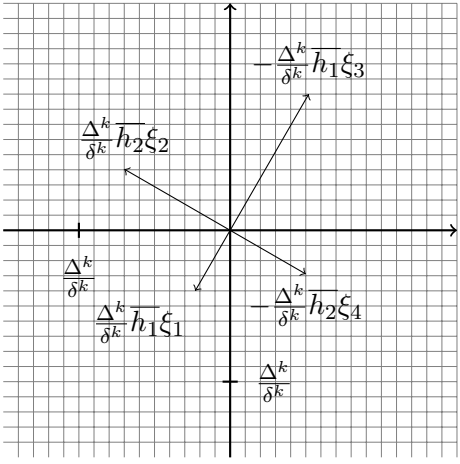
Fin

RETOURNER $\mathbb{P}$.

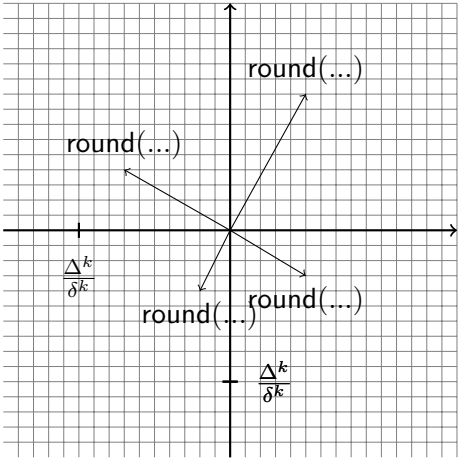
POLL enrichie



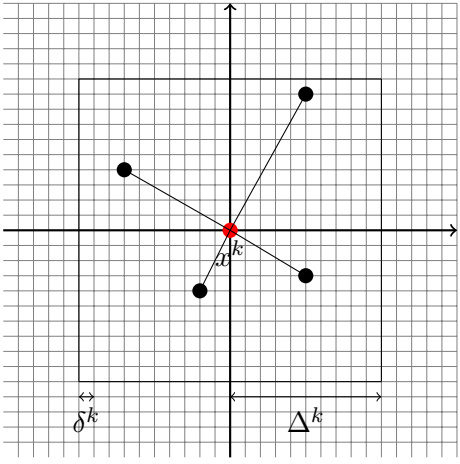
POLL enrichie



POLL enrichie



POLL enrichie



POLL enrichie

Algorithme 15 POLL enrichie - Génération de points

[0]. Initialisation :

- Δ^k : Pas de cadre de POLL
- δ^k : Pas de maillage
- q : Nombre de bases positives à générer
- x^k : Point donnant lieu au succès courant
- $\mathbb{D} \leftarrow \varnothing$: Ensemble de direction de POLL
- $\mathbb{P} \leftarrow \varnothing$: Ensemble de points de POLL

[1]. Génération de direction de POLL :

Pour $i \in [\![1, q]\!]$ **faire** :

Sélectionner $u \in \mathbb{R}^n$ non nulle.

Poser $v = \frac{u}{\|u\|_2}$.

Poser $H = I - 2vv^\top$.

Sélectionner $\xi = (\xi_1, ..., \xi_{2n}) \in [0, 1]^{2n}$ de manière aléatoire.

$$\mathbb{D} \leftarrow \mathbb{D} \cup \left\{ + \delta^k \text{round}\left(\frac{\Delta^k}{\delta^k} \frac{h_j}{\|h_j\|_\infty} \xi_j \right) \ : \ j \in [\![1, n]\!] \right\}$$

$$\cup \left\{ - \delta^k \text{round}\left(\frac{\Delta^k}{\delta^k} \frac{h_j}{\|h_j\|_\infty} \xi_{n+j} \right) \ : \ j \in [\![1, n]\!] \right\}.$$

Fin

[2]. Génération de points de POLL :

$\mathbb{P} \leftarrow \{x^k + d \ : \ d \in \mathbb{D}\}$.

RETOURNER $\mathbb{P}$.

Intensification dynamique lors de l'étape de POLL

- Générer un grand nombre de points que lorsque cela s'avère nécessaire.
- Définir "nécessaire".
- Quantifier "intensifier".

POLL dynamique

$$\text{DynamicPoll} : \mathbb{R}_+^* \times \mathbb{R}_+^* \times \mathbb{R}^n \times \mathbb{N}^* \to \mathbb{R}^n$$
$$(\delta^k, \Delta^k, x^k, n_p^k) \mapsto \mathbb{P}^k$$

avec :

$$|\mathbb{P}^k| = n_p^k.$$

POLL dynamique

$$\begin{aligned} \text{DynamicPoll} : \mathbb{R}_+^* \times \mathbb{R}_+^* \times \mathbb{R}^n \times \mathbb{N}^* &\to \mathbb{R}^n \\ (\delta^k, \Delta^k, x^k, n_p^k) &\mapsto \mathbb{P}^k \end{aligned}$$

avec :

$$|\mathbb{P}^k| = n_p^k.$$

Comment définir n_p^k à chaque itération k ?

POLL dynamique : historique sans mémoire

$$e^0 = \begin{cases} 1 & \text{Si l'étape de POLL à l'itération } 0 \text{ est un échec.} \\ 0 & \text{Sinon.} \end{cases}$$

$$e^k = \begin{cases} e^{k-1} + 1 & \text{Si l'étape de POLL à l'itération } k \\ & \text{est un échec.} \\ 0 & \text{Sinon.} \end{cases}$$

POLL dynamique : historique avec mémoire

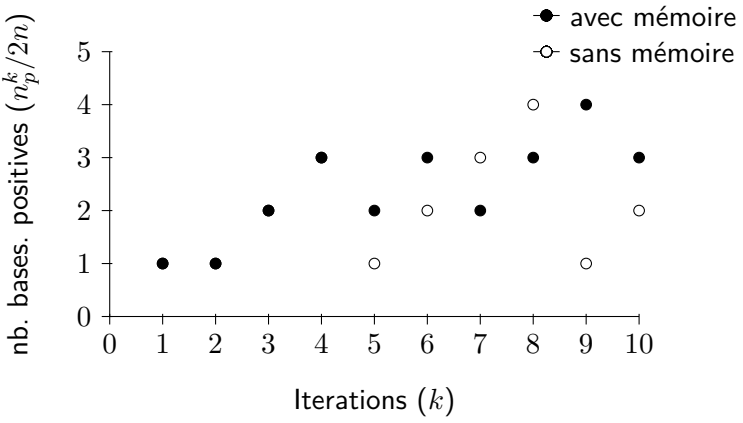
$$e^0 = \begin{cases} 1 & \text{Si l'étape de POLL à l'itération } 0 \text{ est un échec.} \\ 0 & \text{Sinon.} \end{cases}$$

$$e^k = \begin{cases} e^{k-1} + 1 & \text{Si l'étape de POLL} \\ & \text{à l'itération } k \text{ est un échec.} \\ \max\{e^{k-1} - 1, 0\} & \text{Sinon.} \end{cases}$$

POLL dynamique

$$
\begin{aligned}
n_p^k &= g((e^\ell)_{1\leq\ell\leq k-1}, n_p^{max}) \\
&= \min(2n \times (1 + e^{k-1}), n_p^{max}) \text{ : intensification linéraire} \\
&= \min(2n \times 2^{e^{k-1}}, n_p^{max}) \text{ : intensification exponentielle}
\end{aligned}
$$

Effet de la mémoire sur le nombre de points de sonde



Résultats numériques

Métriques

$\mathfrak{P}$: ensemble de problèmes.
$\mathfrak{A}$: ensemble d'algorithmes.
τ : tolérance $\in [0, 1]$.
f_L^p : meilleure valeur connue de f^p, toutes résolutions confondues.

Métriques

$N_{p,a}$: valeur d'une variable algorithmique d'intérêt lorsque

$$\frac{f^p(x^0) - f^p(x^k)}{f^p(x^0) - f_L^p} \geq (1 - \tau)$$

est satisfaite pour la première fois lors de l'exécution de a sur p.
$r_{p,a}$: ratio de performance

$$r_{p,a} := \frac{N_{p,a}}{\min\{N_{p,\tilde{a}} : \tilde{a} \in \mathfrak{A}\}}.$$

Métriques

Définition (Profil de performance Moré and Wild (2009))

Le profil de performance de l'algorithme a est le graphe de la fonction :

$$\rho_a : \alpha \mapsto \frac{|\{p \in \mathfrak{P} : r_{p,a} \leq \alpha\}|}{|\mathfrak{P}|},$$

qui représente la proportion de problèmes résolus par a à l'aide de au plus α fois le budget du meilleur algorithme sur chaque problème.

Métriques

Définition (Profil de donnée en évaluation Moré and Wild (2009))

Le profil de donnée de l'algorithme a est le graphe de la fonction :

$$d_a : \kappa \mapsto \frac{|\{p \in \mathfrak{P} : \frac{N_{p,a}}{n_p+1} \leq \kappa\}|}{|\mathfrak{P}|},$$

qui représente le nombre de problèmes résolus par a en un budget d'au plus κ groupes de $n_p + 1$ évaluations.

$N_{p,a} \equiv$ évaluations

Métriques

Définition (Profil de donnée en itération (inspiré de Moré and Wild (2009)))

Le profil de donnée en itération de l'algorithme a est le graphe de la fonction :

$$d_a : \kappa \mapsto \frac{|\{p \in \mathfrak{P} : N_{p,a} \leq \kappa\}|}{|\mathfrak{P}|},$$

qui représente le nombre de problèmes résolus par a en un budget d'au plus κ itérations.

$N_{p,a} \equiv$ itérations

Problèmes analytiques

- 24 problèmes sans contraintes Finck et al. (2010).
- Possibilité de générer plusieurs instances du même problème (graine aléatoire), 5 instances sont générées.
- $n \in \{2, 4, 8, 16, 32, 64\}$.
- $\mathcal{X} = [-5, 5]^n$.
- $f(x^*) = 0$ pour tous les problèmes (en théorie, pas toujours atteint en pratique).

Problème réaliste : STYRENE

- simulation numérique (`https://github.com/bbopt/styrene`).
- $n = 8$.
- 4 contraintes binaires.
- 7 contraintes relaxables.
- 10 instances générées en changeant le point initial.
- 1 évaluation $\simeq$ 1 seconde.

Paramètres algorithmique utilisés

- Toutes les options de `SEARCH` sont désactivées.
- À chaque étape de `POLL`, tous les points générés sont évalués (pas d'opportunisme).
- Le maillage anisotropique n'est pas utilisé.

Intéret de l'exploration de l'intérieur du cadre

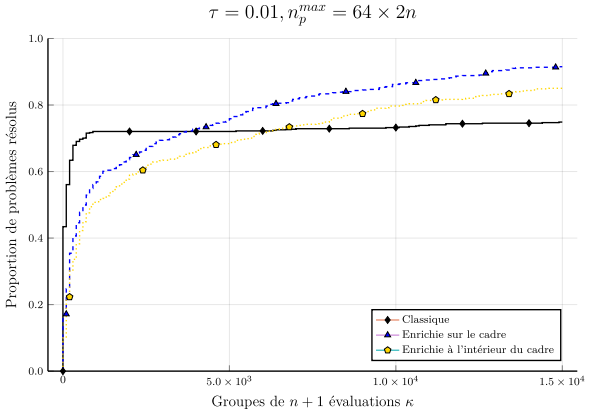


FIGURE – Profils de données : comparaison entre génération de points à la frontière et à l'intérieur du cadre - toutes dimensions confondues.

Intéret de l'exploration de l'intérieur du cadre

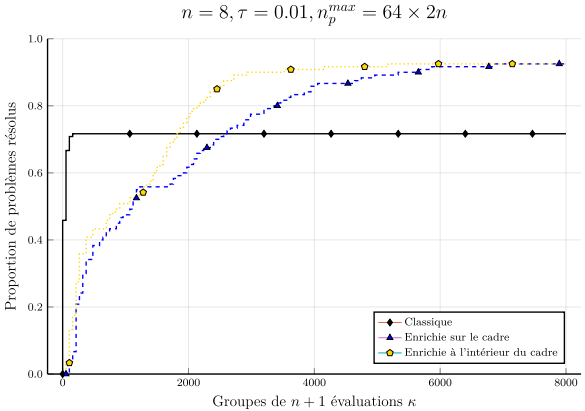


FIGURE – Profils de données : comparaison entre génération de points à la frontière et à l'intérieur du cadre - dimension 8.

Valeurs finales moyennes

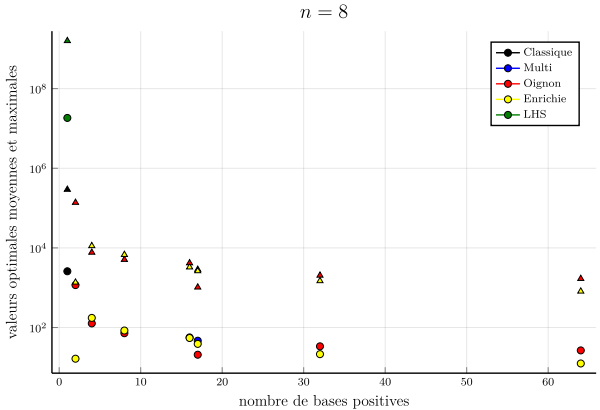


FIGURE – Valeurs moyennes et maximum en fonction du nombre de bases positives.

Influence de la géométrie

$\tau = 0.01, n_p^{max} = (2n+1) \times 2n$

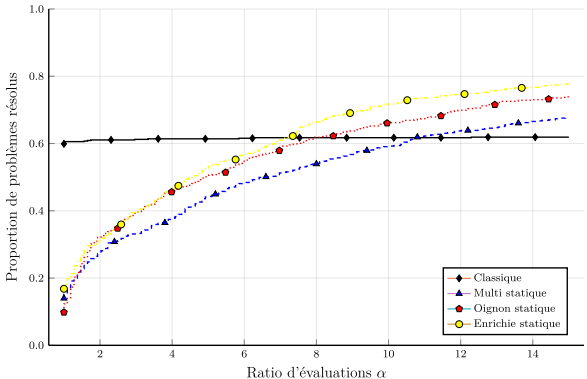


FIGURE – Profils de performances en évaluations.

Influence de la géométrie

$\tau = 0.01, n_p^{max} = (2n+1) \times 2n$

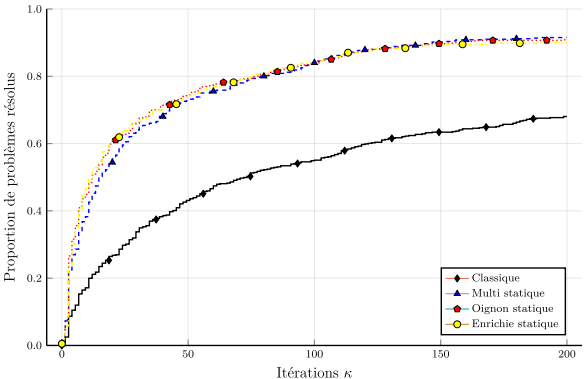


FIGURE – Profils de données en itérations.

Influence de la géométrie

$\tau = 0.01, n_p^{max} = (2n+1) \times 2n$

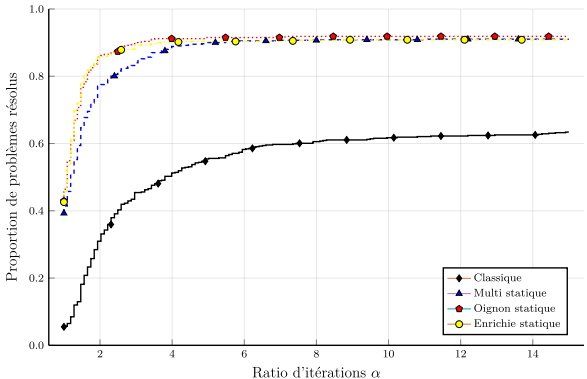


FIGURE – Profils de performance en itérations.

Résultats numériques

- Il faut générer une grande quantité de points pour observer une amélioration des performances
- Pas de différence notable en terme de performance selon la géométrie donnée à $\mathbb{P}^k$.
- Peut-on faire mieux ?

POLL dynamique

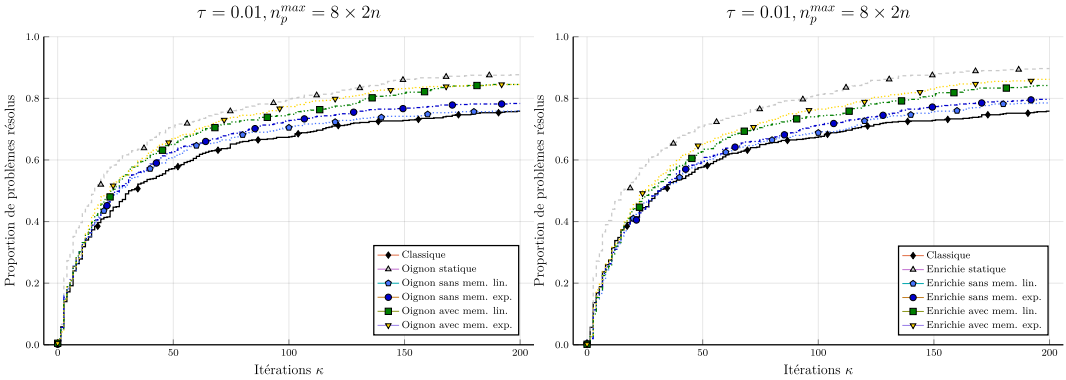


FIGURE – Profils de données en itérations : différences entre Oignon et Enrichie.

POLL dynamique

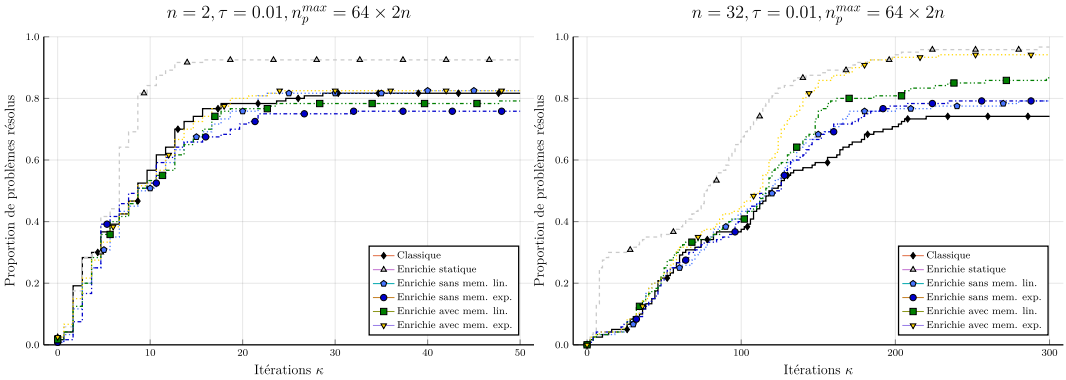


FIGURE – Profils de données en itération : influence de l'augmentation de la dimension.

POLL dynamique

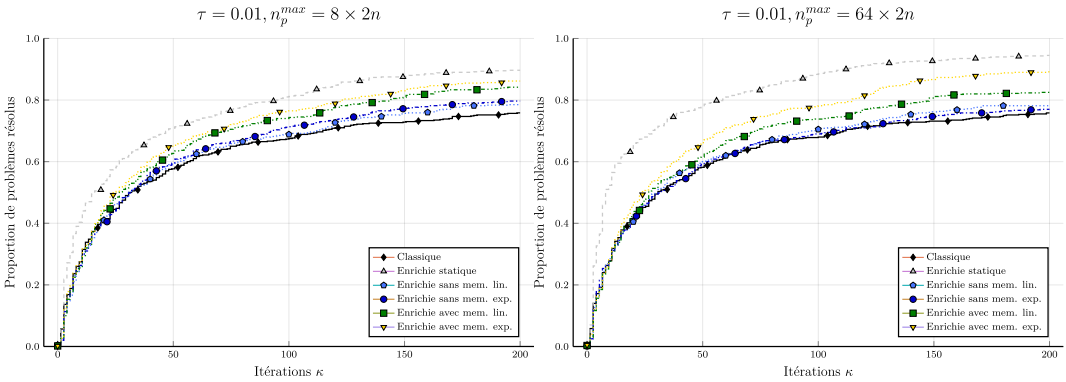
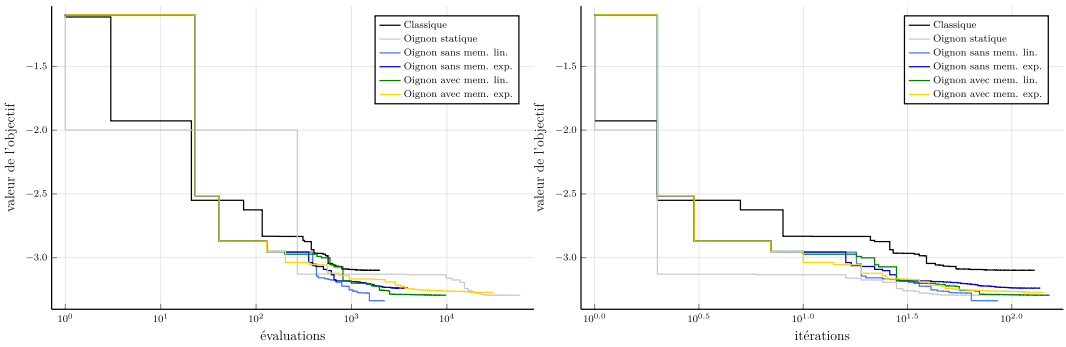


FIGURE – Profils de données en itérations : influence de l'augmentation de n_p^{max}.

Comportement dans un contexte réaliste

- Comparaison d'exécutions réalisées avec et sans mécanismes additionnels d'aide à la résolution (`SEARCH`, `ANISOTROPIC_MESH`, `OPPORTUNISTIC_EVAL`, etc.)
- Exécutions effectuées sur un problème réaliste avec contraintes : `STYRENE`.

Graphes de convergence



(a) (b)

FIGURE – Graphe de convergence lors de la résolution de STYRENE, POLL seule.

Graphes de convergence

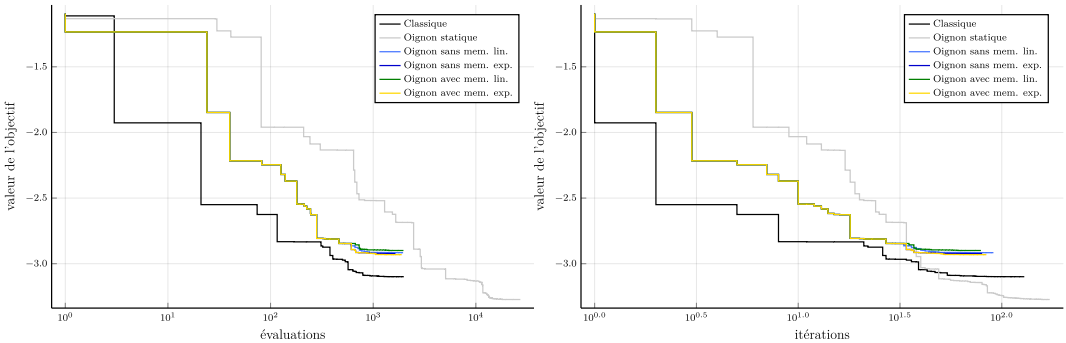


FIGURE – Graphe de convergence lors de la résolution de STYRENE, POLL + mécanismes supplémentaires.

Profils de donnée

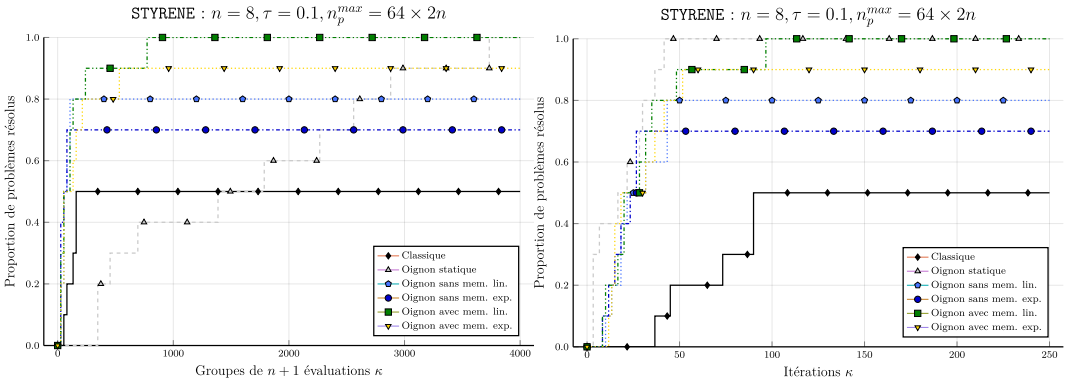
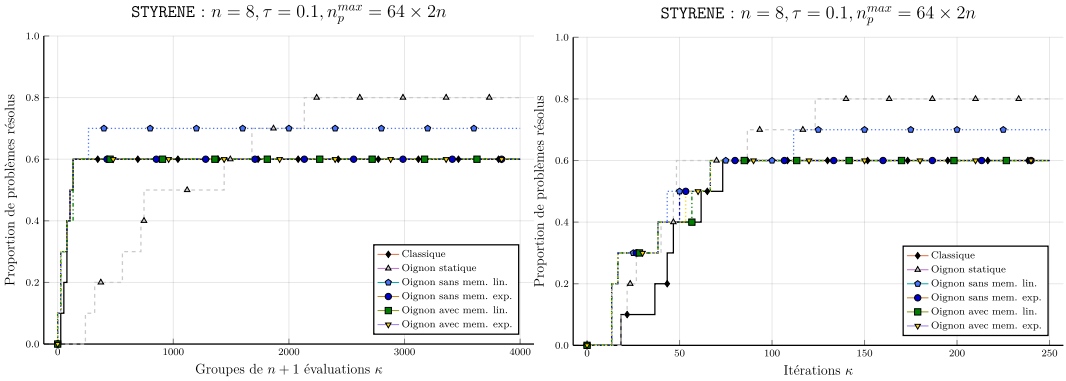


FIGURE – Profils de données lors de la résolution de STYRENE, POLL seule.

Profils de donnée



(a) (b)

FIGURE – Profils de données lors de la résolution de STYRENE, POLL + mécanismes supplémentaires.

Travail à suivre

Travail à suivre

- Ordonnancement de la pile d'évaluations à l'aide de fonctions d'estimation du temps et des ressources nécessaire à chaque évaluation.
- Définir DynamicSearch similaire à DynamicPoll, fonctionnant de manière complémentaire.
- Autoriser des «sauts» dans la stratégie de mise à jour du maillage lorsque la DynamicPoll est utilisée.

Audet, C. and Dennis, Jr., J. (2006). Mesh Adaptive Direct Search Algorithms for Constrained Optimization. *SIAM Journal on Optimization*, 17(1) :188–217.

Finck, S., Hansen, N., Ros, R., and Auger, A. (2010). Real-parameter black-box optimization benchmarking 2010 : Presentation of the noiseless functions. Technical report, INRIA.

Moré, J. and Wild, S. (2009). Benchmarking derivative-free optimization algorithms. *SIAM Journal on Optimization*, 20(1) :172–191.

Talgorn, B., Alarie, S., and Kokkolaras, M. (2020). Parallel Surrogate-based Optimization Using Mesh Adaptive Direct Search. Technical Report G-2020-38, Les cahiers du GERAD.