

# Développement de NOMAD 4

Mise à jour

Viviane Rochon Montplaisir  
Christophe Tribes



POLYTECHNIQUE  
MONTREAL



LE GÉNIE  
EN PREMIÈRE CLASSE



# Pourquoi NOMAD 4?

Motivé par discussions avec Pascal Côté (Rio Tinto) et Stéphane Alarie (Hydro-Québec)

- Redémarrage à chaud: Piloter MADS, modifier des paramètres durant l'exécution
- Runner: Exécutable qui permet de rouler plusieurs problèmes avec différents algorithmes, aux fins de comparaison
- Fluidité de la résolution: Éviter les goulots d'étranglement
- Identifier les variables importantes
- Exploiter le parallélisme, grappes de calcul

# Pourquoi NOMAD 4?

- Faciliter les ajouts et la modification de code:
  - ▶ Ajout de paramètres
  - ▶ Notation fidèle au livre de Charles Audet et Warren Hare
  - ▶ Une classe par étape d'algorithme
  - ▶ Code clair, modulaire et générique
- Séparer l'algorithme MADS de base de ses spécialisations
  - ▶ PSD-MADS, UniMADS, OrthoMADS, QR-MADS, BiMADS, RobustMADS, STATS-MADS
  - ▶ Variables de catégories

# NOMAD 4 Alpha

NOMAD 4 Alpha est disponible en Janvier 2019.

- Innovations
  - ▶ Redémarrage à chaud
  - ▶ Queue d'évaluation
  - ▶ Parallélisme avec OpenMP
- Mises à jour et améliorations
  - ▶ Version batch et library
  - ▶ Paramètres
  - ▶ Cache
  - ▶ Contraintes
  - ▶ Sortie
  - ▶ Tests unitaires

# Redémarrage à chaud

- Interrompre NOMAD
- Modifier des paramètres
- Continuer la résolution de là où NOMAD était rendu
- Permet de piloter la résolution

# Redémarrage à chaud - Exemple 1 (Hot Restart)

```
351 351 ( 7.8 5.5 8.6 5.9 6.7 4.3 ) -871.0 0
352 352 ( 7.7 5.2 8.7 5.8 5.6 4.2 ) -1978.8 0
^C
NOMAD caught User interruption.
Hot restart: Enter a parameter file name,
or enter parameter values, ending with CTRL-D.
UPPER_BOUND * 8.1

383 383 ( 7.3 5.3 8.1 5.7 6.4 3.8 ) -1498.4 0
384 384 ( 8.1 5 8.1 5.8 6.9 4.5 ) -741.6 0
386 386 ( 8.1 5.2 8.1 5.7 6.5 4 ) -1691.4 0
```

- Les valeurs des variables sont trop élevées
- L'utilisateur interrompt la résolution
- Il ajuste le paramètre UPPER\_BOUND
- La résolution continue avec la nouvelle valeur.

# Redémarrage à chaud - Exemple 2 (Warm Restart)

```
97 97 ( 7.1 6.1 8 5 7.1 3 ) -1678.0 0
99 99 ( 7.4 5 7.9 5 7.2 4.1 ) -1914.4 0
A termination criterion is reached: Maximum number of evaluations, including cache hits
Save information for hot restart.

Best feasible solution: ( 7.5 6.2 7.8 5 6 4.1 ) Evaluation OK f = -1997.3 h = 0
```

- La résolution arrête après 100 évaluations
- Modification de MAX\_EVAL dans le fichier de paramètres
- La nouvelle exécution reprend là où NOMAD était rendu

```
[nomad]% nomad param.txt
Read cache file ./cache.txt
Read hot restart file ./hotrestart.txt
105 105 ( 7.6 6.2 8.2 5.2 5 4.1 ) -1946.5 0
107 107 ( 7.6 6.2 8 4 6.2 4.1 ) -1828.0 0
```

Dans NOMAD 4 Alpha:

- OpenMP pour utiliser plusieurs cœurs à la fois
- Les points à évaluer sont dans une queue ordonnée
- Les évaluations sont effectuées en parallèle, de manière asynchrone

À venir:

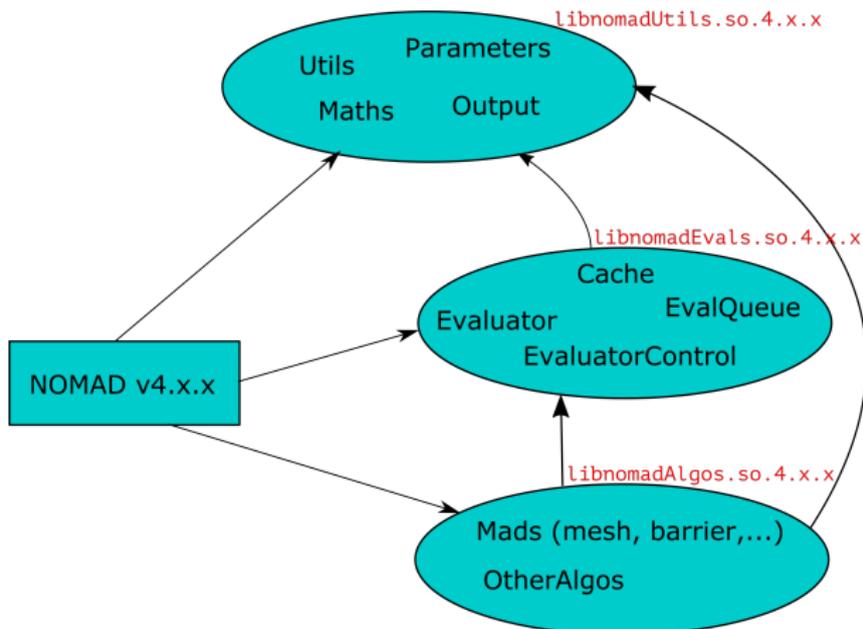
- Utilisation de MPI pour rouler sur plusieurs ordinateurs en même temps. Grappe de calcul

# Architecture modulaire

- Modules Utilitaires
  - ▶ Output
  - ▶ Maths
  - ▶ Parameters
  - ▶ Utils: autres fonctions utilitaires
- Modules d'Evaluation
  - ▶ Eval: Queue d'évaluation, points d'évaluation
  - ▶ Cache
- Modules d'Algorithmes
  - ▶ Mads
- Executable NOMAD

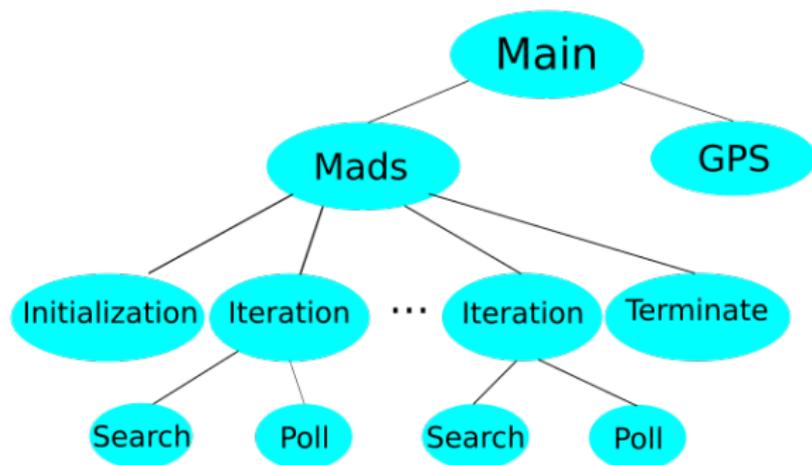
# Structure hiérarchique

Les composantes du code sont regroupées dans des bibliothèques.



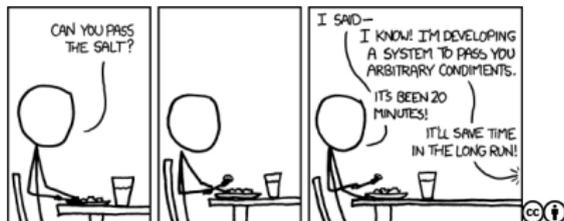
# Structure hiérarchique

Les étapes d'un algorithme sont organisées de manière hiérarchique



# Nouvelle syntaxe pour l'accès aux paramètres

## Généraliser et simplifier l'accès aux paramètres



- Accès générique, en mode librairie et dans le code
- Classe AllParameters pour l'accès facile

```
auto allParams = std::make_shared<NOMAD::AllParameters>();
allParams.setAttributeValue( "DIMENSION", 10 );
allParams.setAttributeValue( "X0", NOMAD::Point(10, 5.0) );
allParams.setAttributeValue( "DISPLAY_DEGREE", 2 );
...

auto n = allParams.getAttributeValue<int>( "DIMENSION" );
```

# Nouvelle syntaxe pour l'accès aux paramètres

Les paramètres sont divisés en composantes principales:

Problem, Run, Cache, Display, Evaluator, EvaluatorControl.

Seuls les paramètres utiles à une classe sont passés en arguments au constructeur.

```
// Creation of a Cache with CacheParameters
    NOMAD::CacheSet::setInstance( _allParams->getCacheParams() );

...

// Creation of a Mads step using RunParameters, PbParameters
// and an EvaluatorControl
    NOMAD::Mads mads( this,
        _allParams->getRunParams(),
        _allParams->getPbParams(),
        evaluatorControl );
```

# Nouvelle syntaxe pour l'accès aux paramètres

Facile d'ajouter un nouveau paramètre dans un fichier de définition!

```
#####~
##### ..... Definition of CacheParameters attributes ..... #####~
#####~
##### . Each definition must be separated by at least a comment line~
##### ..... (starting by #)~
#####~
##### . The definition must provide the name, type and default value~
##### ..... (no default: '-' or 'N/A') in this order, one by line.~
#####~
##### . Only a single word is considered for the name,~
##### ..... type and default value (extra is comment)~
#####~
##### . The short info, the help and the keywords must be provided in this~
##### ..... order within: "(" and ")". For keywords, the plural part of a word~
##### ..... can be put in parenthesis; both singular and plural will be keywords~
#####~
#####~
##### ..... #####~
MAX_CACHE_SIZE~
int~
-1~
\(. Termination criterion on the number of evaluation points stored in the cache.\)~
\(~
~
. The program terminates as soon as the cache reaches this size.~
~
. Argument: one positive integer (expressed in number of~
. evaluation points).~
~
. No default (-1)~
~
. Example: MAX_CACHE_SIZE=10000~
~
\~
\(. advanced termination cache.\)~
#####~
#####~
```

# Compatibilité NOMAD 3 - NOMAD 4

- Mode batch:
  - ▶ Les appels de boîtes noires externes sont identiques.
  - ▶ Les paramètres ont les mêmes noms lorsque la fonctionnalité existe dans NOMAD 4.
  - ▶ Un fichier de paramètres de base pour NOMAD 3 peut être lu par NOMAD 4.
- Mode librairie:
  - ▶ Requiert modification du code.
  - ▶ Nouvelle syntaxe pour l'accès aux paramètres.
  - ▶ Nouvelle syntaxe pour Step, Evaluator, ...
  - ▶ Fonction d'évaluation `eval_x()` presque identique.

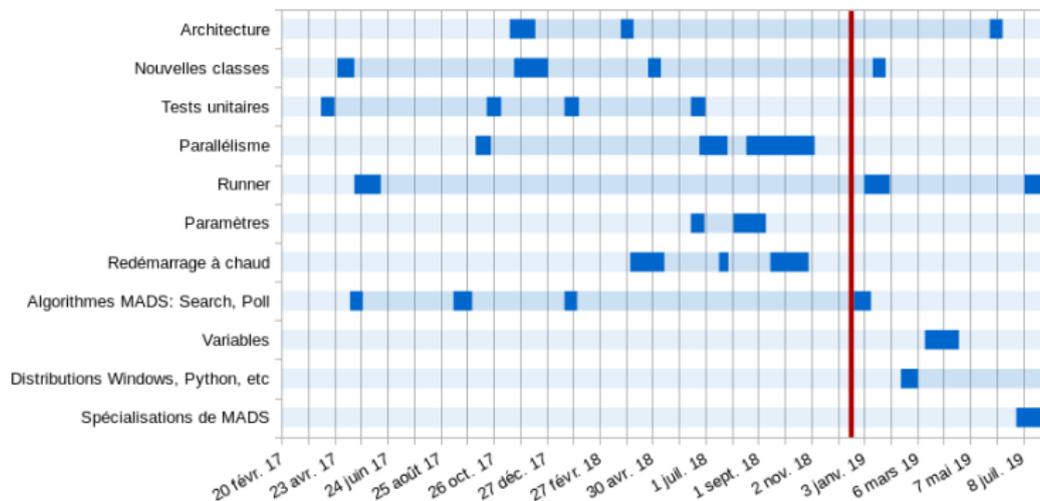


- Exécutable complémentaire à NOMAD
- Banque de plus de 130 problèmes
- Permet de rouler des tests sur plusieurs problèmes, en variant les algorithmes
- Plusieurs optimiseurs différents peuvent être utilisés
- Comparaison des résultats pour validation des algorithmes

# Structure hiérarchique — Nouveau Runner

- Le Runner doit pouvoir comparer la performance de plusieurs versions de NOMAD (3.9.1 et 4.x.x)
- NOMAD 3 a un Runner
- NOMAD 4 (pré Alpha) a été testé avec ce Runner
- Problèmes du Runner basé sur NOMAD 3:
  - ▶ Complicé d'inclure un nouvel algo
  - ▶ Travail à recommencer à chaque nouvelle version de NOMAD 4
- Le nouveau Runner est un package externe à NOMAD 4
- Tient compte de la structure hiérarchique de NOMAD 4
- Réutilise la librairie de NOMAD 4 `libnomadUtils.so.4.x.x`

# Avancement du projet



# Conclusion

Tel que spécifié dans l'échéancier, NOMAD 4 Alpha inclut:

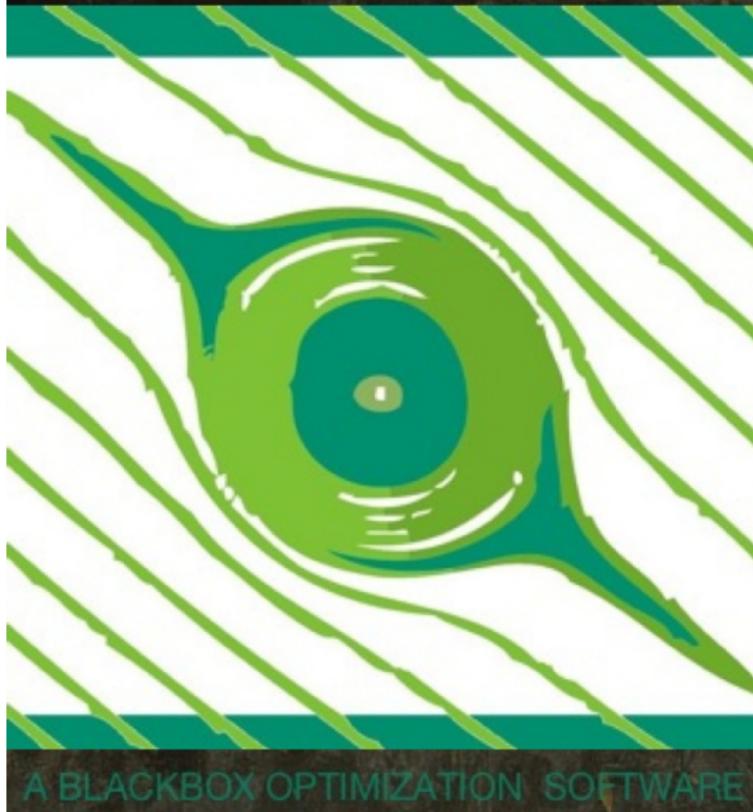
- Architecture revampée
- Nouvelles classes pour Cache, Output
- Queue d'évaluation
- Classes de Paramètres
- Évaluations en parallèle
- Redémarrage à chaud

Une première version Beta est prévue pour ce printemps.

# Prochains développements (Beta)

- Parallélisme avec MPI
- Stratégies de recherche (Search) - Hypercube Latin, User Search
- Nouveau Runner
- Classe générale d'algorithmes
- Variables
  - ▶ Variables fixes
  - ▶ Groupes de variables
  - ▶ Variables de catégories
- Distributions
  - ▶ Version Python
  - ▶ Version Windows
- Bibliothèques externes
  - ▶ Bibliothèque sgtelib
  - ▶ Faciliter l'ajout de bibliothèques externes (SVD, SQLite, etc)

# NOMAD



A BLACKBOX OPTIMIZATION SOFTWARE