Développement de NOMAD 4

Présentation à Rio Tinto

Viviane Rochon Montplaisir

Mars 2018

Historique

- **Version 1** (2001) Prototype - Charles Audet et Gilles Couture
- **Version 2** (2004) Charles Audet et Gilles Couture
- **Version 3** (2008) Sébastien Le Digabel
- **Version 4** Viviane Rochon Montplaisir et Christophe Tribes
  - Nouveau cycle
  - Rafraîchir l'architecture, la structure

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

Environnement de développement

- Le langage C++ a été choisi de nouveau, pour sa rapidité, sa versatilité, et afin de réutiliser du code de NOMAD 3
- Le standard C++11 est maintenant utilisé
- Le code est sur GitHub pour faciliter la gestion de version, la revue de code, et plus tard la distribution
- Développement à saveur agile. Code stable disponible en tout temps

Travail accompli

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

Redémarrage à chaud

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

Redémarrage à chaud

- Permet de continuer si NOMAD s'est interrompu pour des raisons externes
- Permet de piloter la résolution

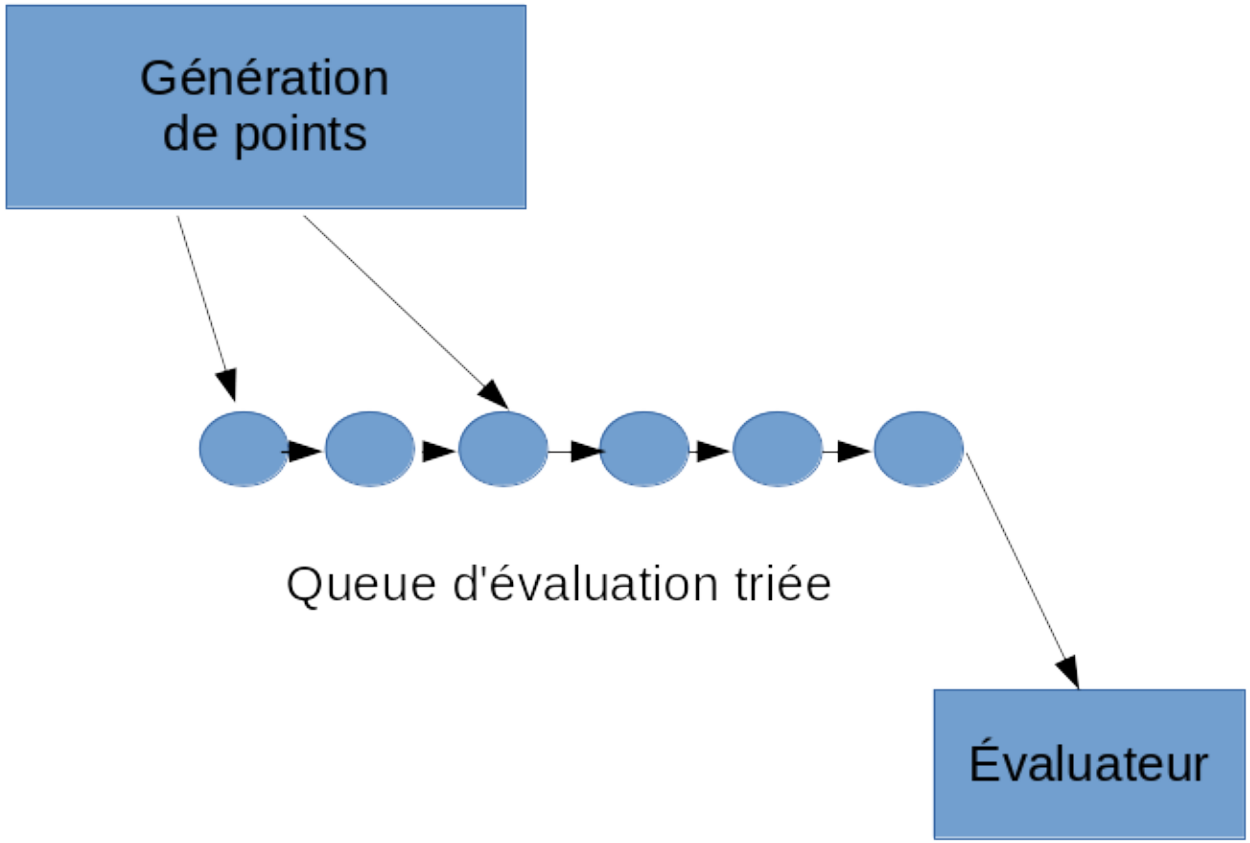
Redémarrage à chaud - Exemples

- Utiliser un algorithme jusquà ce qu'une solution réalisable soit trouvée, puis en utiliser un autre
- Changer la valeur d'un paramètre après un certain nombre d'itérations
- Changer les contraintes progressives PB en EB
- Utilisation de l'intelligence artificielle pour ajuster les paramètres

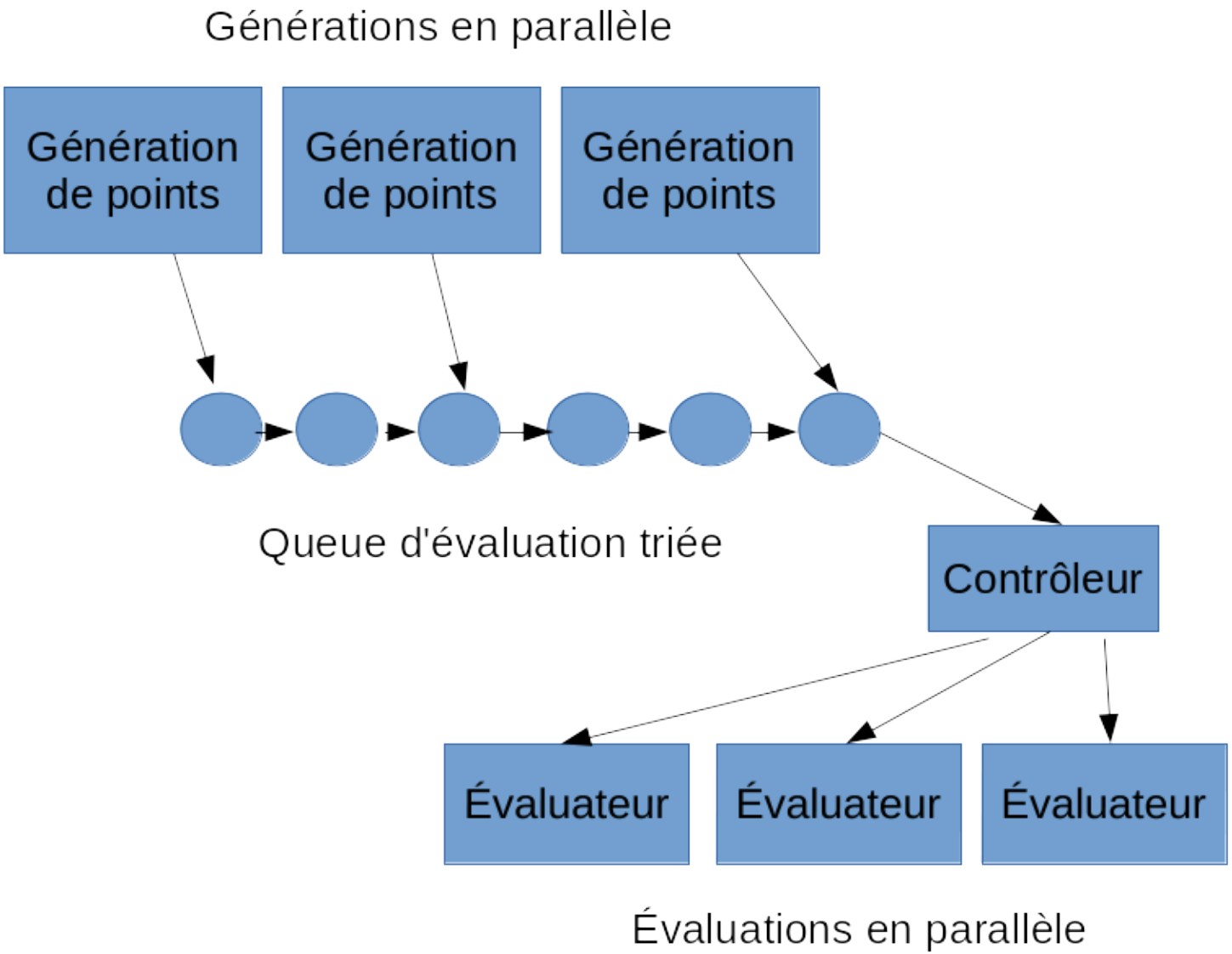
Queue d'évaluation

- Queue de tous les points à évaluer
- Seuls les points qui ne sont pas déjà évalués (dans la cache) sont ajoutés à la queue
- Chaque point a une priorité d'évaluation
- Les points sont évalués par ordre de priorité
- La queue se remplit en même temps qu'elle se vide

Queue d'évaluation



Queue d'évaluation avec parallélisme



Parallélisme

- Le parallélisme est central à NOMAD 4
- NOMAD 4 supporte OpenMP pour utiliser plusieurs cœurs à la fois
- Compilation sans OpenMP disponible
- Possibilité de limiter le nombre de fils d'exécution
- À venir: Utilisation de MPI pour rouler sur plusieurs ordinateurs en même temps. Grappe de calcul

Runner

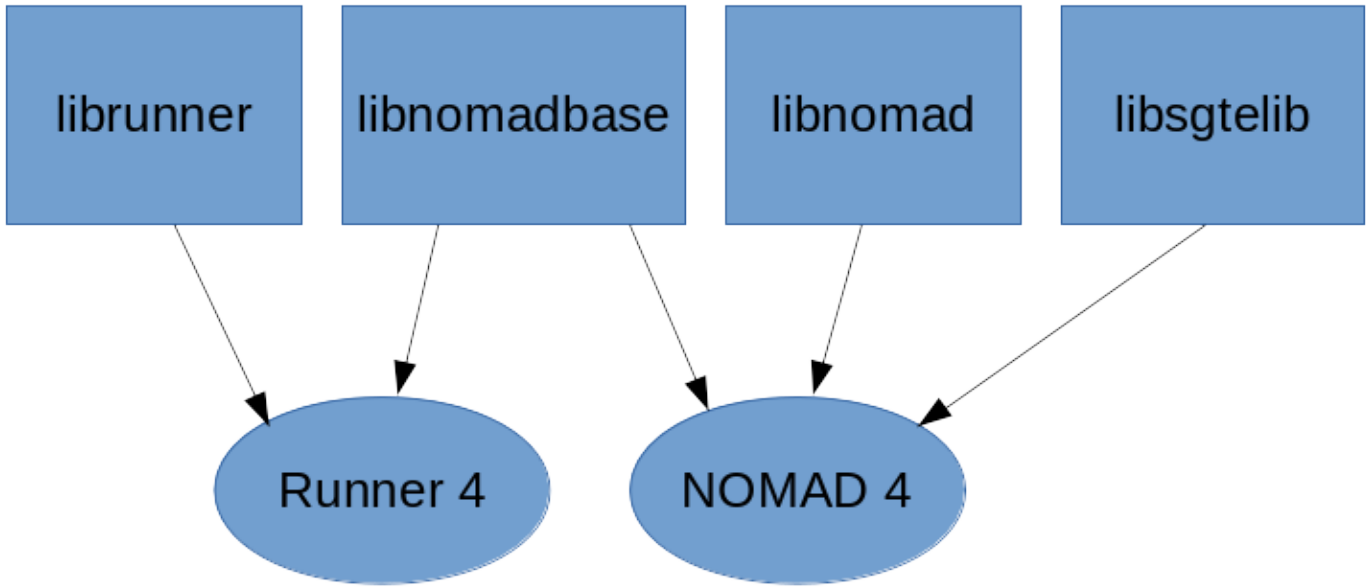
- Exécutable complémentaire à NOMAD
- Banque de plus de 130 problèmes
- Permet de rouler des tests sur plusieurs problèmes, en variant les algorithmes
- Plusieurs optimiseurs différents peuvent être utilisés
- Comparaison des résultats pour validation des algorithmes

Runner

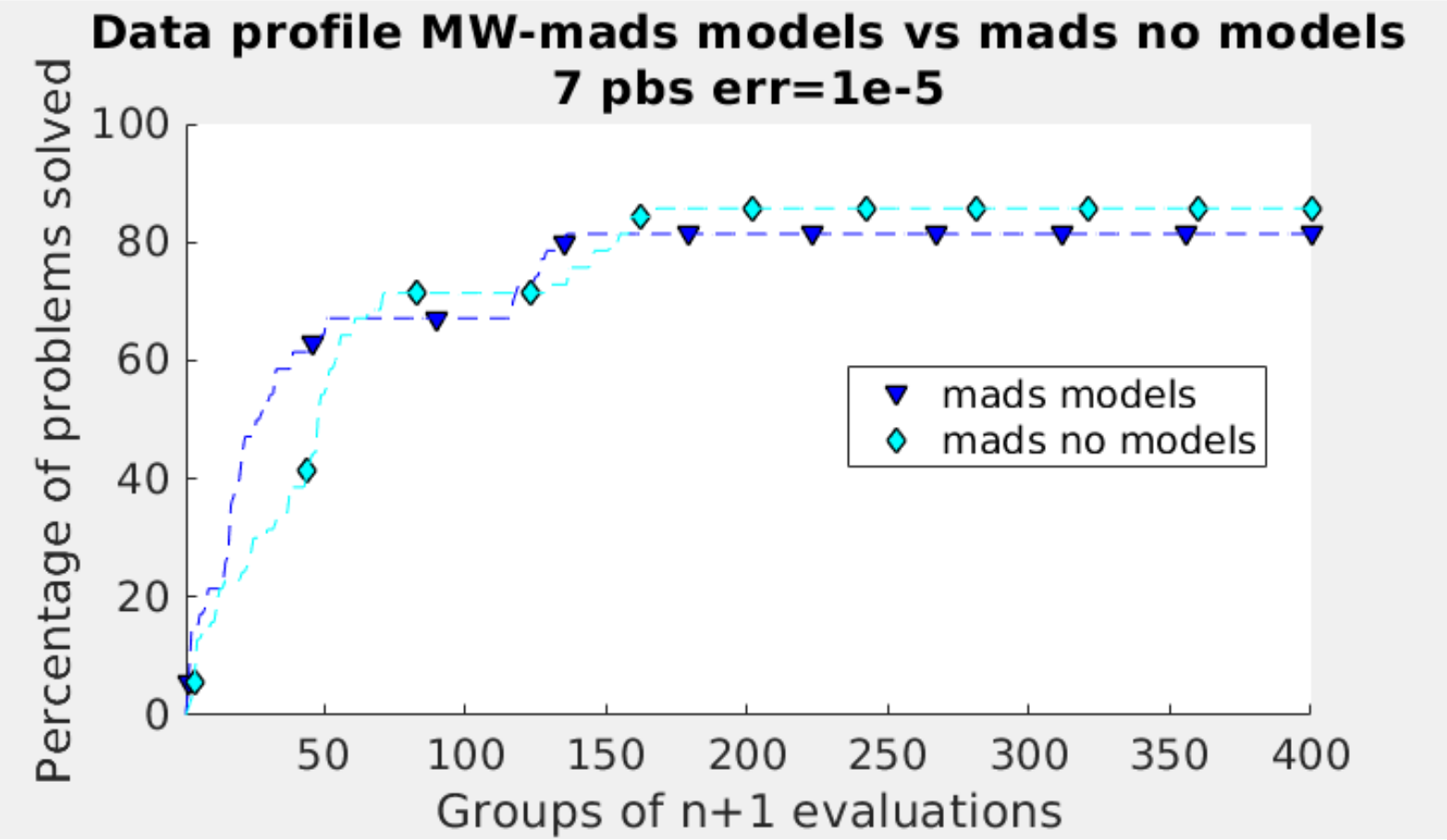
- Le Runner est basé sur le code de NOMAD 3
- NOMAD 4, version en développement, est intégré dans le Runner de NOMAD 3

Runner - Plans pour le futur

- Interface graphique pour faciliter l'utilisation ainsi que l'interprétation des résultats
- Réusinage (Refactoring)



Runner - Data Profile



Versions batch et library

- Utilisation identique à NOMAD 3
- Les mêmes paramètres
- Migration facile pour les utilisateurs de NOMAD 3
- Version simple et fonctionnelle

```
Start step Poll {
    Thread = 4
    Start evaluation of ( 4 4.12 4.48 4.52 4.67 )
    New feasible point found: ( 4 4.12 4.48 4.52 4.67 ) -4.54018 < 10
    Thread = 3
    Start evaluation of ( 5 4.89 4.53 4.49 4.34 )
    Thread = 1
    Start evaluation of ( 4.52 4.48 4.5 4 4.52 )
    New feasible point found: ( 4.52 4.48 4.5 4 4.52 ) -4.67055 < -4.54018
    Thread = 2
    Start evaluation of ( 4.49 4.53 4.51 5 4.49 )
    New poll center: ( 4.52 4.48 4.5 4 4.52 )   Evaluation OK   f = -4.67055    h = 0
} End step Poll
```

Paramètres

- Les paramètres de base de NOMAD 3 sont supportés
- Facile d'ajouter un nouveau paramètre avec un minimum de code
- Support générique
  ```
  p.setAttributeValue("DIMENSION", n);
  p.getAttributeValue<int>("DIMENSION");
  ```
- L'interface NOMAD 3 est aussi conservée pour rétrocompatibilité
- À venir: Classes de paramètres (problème, algorithme, mesh, display, etc)

Cache

- Ensemble des points déjà visités
- Peut devenir très grande
- Réutilisable pour d'autres optimisations
- Une interface de base permet à l'usager de réimplémenter la cache. Par exemple, utilisation de SQLite
- Deux structures de données sont actuellement disponibles: `set` et `unordered_set`
- Plusieurs tests ont été effectués pour maximiser la performance: `set` a été choisi
- `multi_set` sera disponible

Contraintes

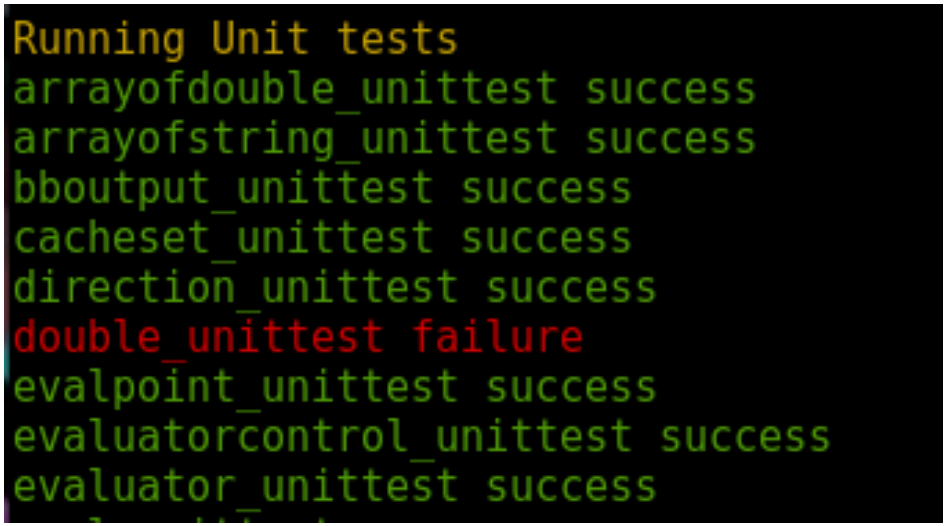
- Contraintes hiérarchiques
- Contraintes linéaires, scaling de l'output
- Contraintes d'exclusion mutuelle, décomposition en sous-problèmes
- Contraintes d'égalité
- Traitement différent des contraintes selon leur taxonomie (quantifiable, relaxable, analytique, cachée)
- Post-analyse de sensibilité des contraintes

Sortie

- Affichage à l'écran ou écriture dans un fichier
- Chaque fil d'exécution ajoute de l'information à une queue
- L'information a un niveau d'importance (debug, info, warning, high, erreur)
- La queue est vidée et imprimée régulièrement
- Les sorties des différents fils d'exécution ne s'emmêlent pas
- La queue contrôle le format et la quantité d'informations à écrire
- La classe génère aussi un fichier de stats

Tests unitaires

- Les tests unitaires utilisent Google Test
- Chaque classe a son test unitaire, qui vérifie que ses méthodes fonctionnent correctement
- Lors d'une compilation, tous les tests unitaires sont lancés et on peut voir tout de suite s'il y en a un de cassé



Problème avec la classe Double

En résumé

NOMAD 4 peut actuellement:

- Résoudre des problèmes avec contraintes
- Démarrer avec un X0 non réalisable
- Utiliser plusieurs cœurs pour l'évaluation
- On peut mesurer sa performance grâce au Runner
- La qualité du code est soutenue par des revues de code et par des tests unitaires

Prochaines étapes

- Boîtes grises
- Versions Windows, Python et Matlab
- Génération de points en parallèle
- Stratégies de recherche (Search)
- Catégories de paramètres
- MPI
- Variables fixes, Groupes de variables
- Variables de catégories
- Spécialisations de MADS: PSD-MADS, UniMADS, etc

Étapes à plus long terme

- Faciliter l'utilisation de librairies externes, par exemple pour des calculs matriciels (SVD)
- Le projet, déjà sur GitHub, deviendra publiquement accessible
- Le site web sera revampé via GitHub

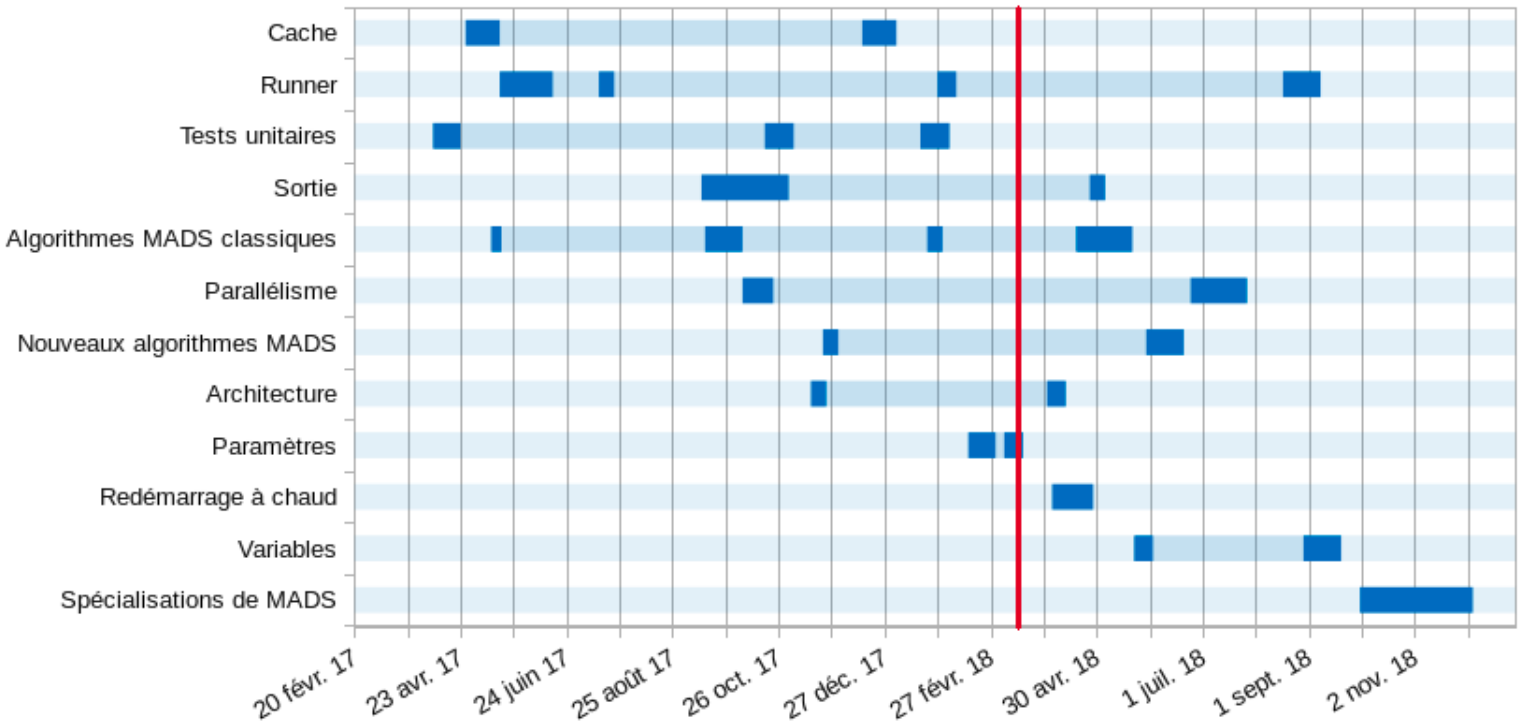
Conclusion

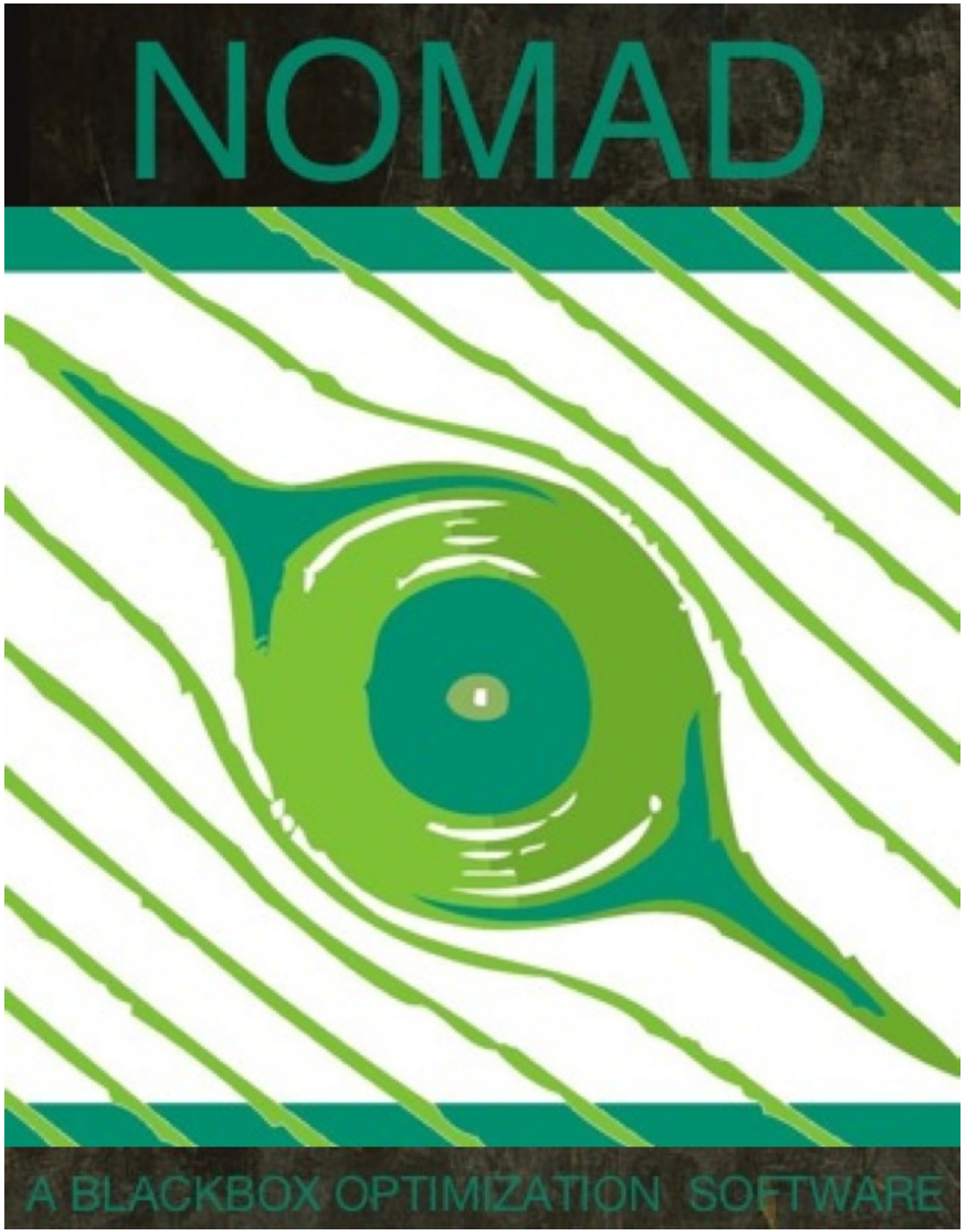
Tel que spécifié dans l'échéancier, nous avons actuellement:

- Version alpha qui inclut de nouvelles classes pour Cache, Queue d'évaluation, Paramètres, Sortie
- Parallélisme possible
- Redémarrage à chaud: en cours
- Runner qui supporte NOMAD 4

Une première version Beta est prévue pour cet été.

Avancement du projet




NOMAD
A BLACKBOX OPTIMIZATION SOFTWARE