Algorithmes génétiques

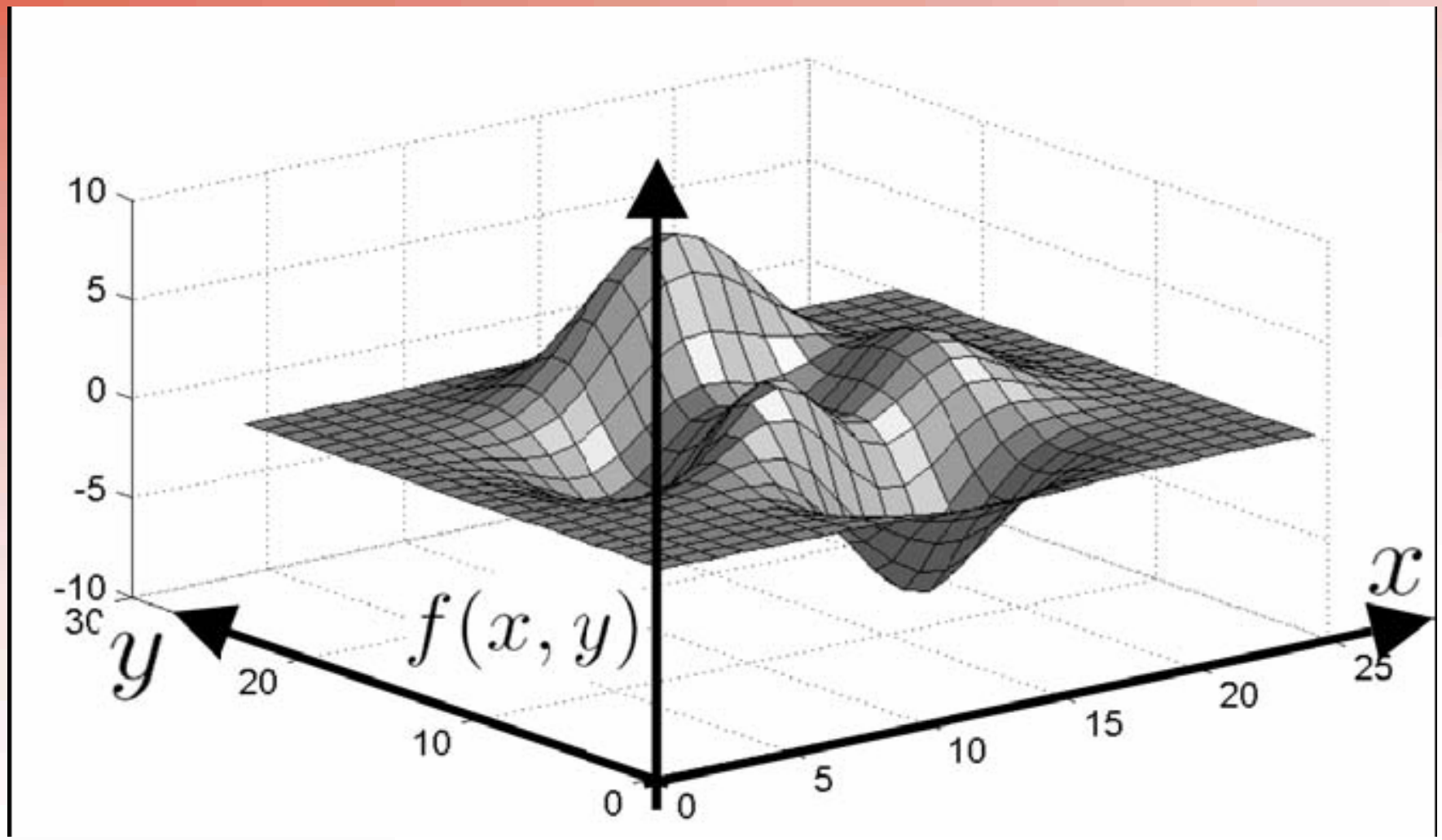
Algorithmes génétiques

- Présentation
  - Problèmes « non classiques »
  - Algorithmes basés sur les heuristiques
  - Algorithmes génétiques
- Fonctionnement
  - Principes de base
  - Optimisation
- Utilisation
  - Modélisation
  - Démos
  - Cas réels
- Conclusion

Présentation

- Problèmes « non classiques »
  - Pas de méthode pour résoudre
    - *Déplacement d'un robot*
  - Modélisation trop complexe
    - *Comportement social*
  - Évolution / Adaptation / Tolérance à l'erreur
    - *Systèmes de perception, d'analyse*

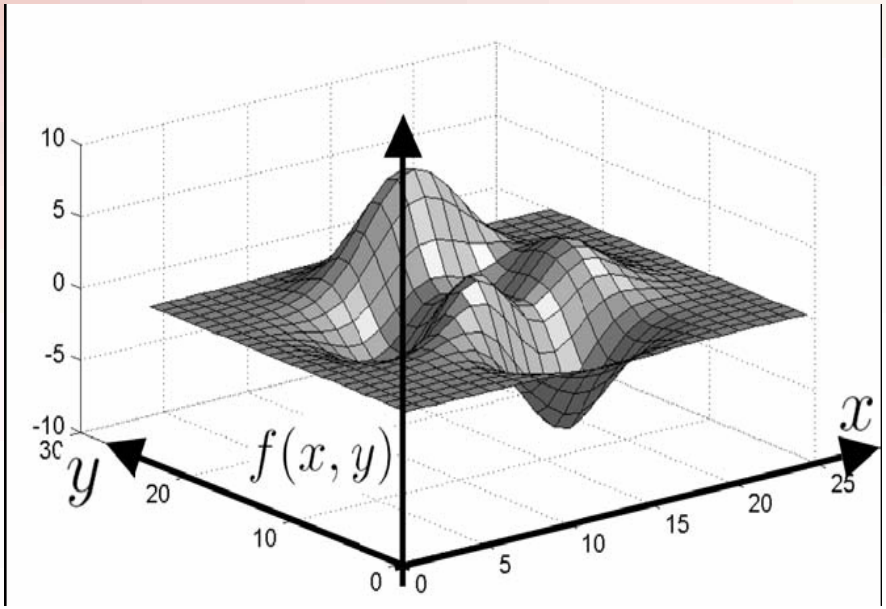
Présentation



Présentation

R = {x,y} tel que g(f(x,y)) est optimal avec (x,y) ϵ I

- *R: Meilleure solution*
- *(x,y): Solution*
- *I: Ensemble des solutions*
- *f(x,y) : Fonction coût*
- *g(): Fonction objectif*



Présentation

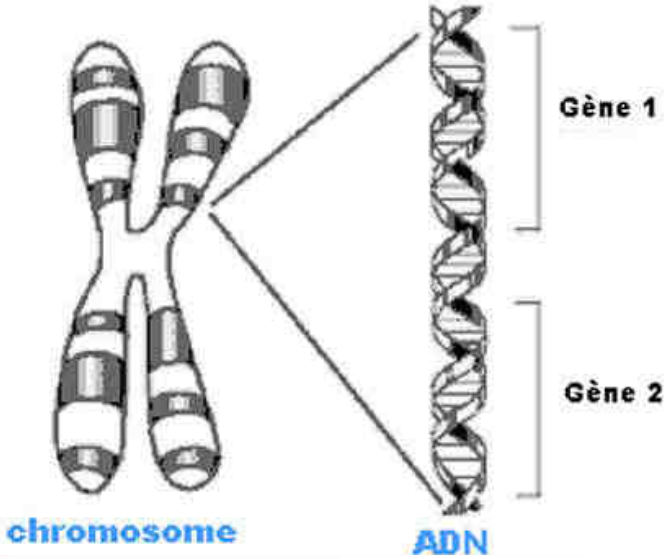
- Une solution: les heuristiques
  - Principaux algorithmes
    - *Brute force (Monte Carlo)*
    - *Hill climbers(gradient descent , annealing, tabu search)*
    - *Evolutionary algorithm (Genetic, ant colony, neurals networks)*
    - *Constraints algorithms (Local consistency, hybrid algorithms)*

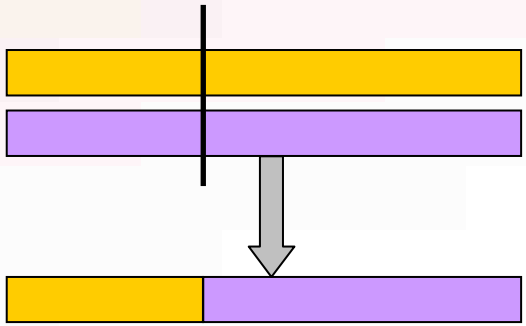
Présentation

- Algorithmes génétiques
  - Origine : Théorie Darwinienne de l'évolution
    - *Struggle for life*
    - *Sélection naturelle*
  - AG inspirés du paradigme
    - *Terminologie identique (population, individu, chromosome, gène)*
    - *Traduction du phénomène*
  - Opérateurs d'évolution
    - *Sélection*
    - *Croisement*
    - *Mutation*

Présentation

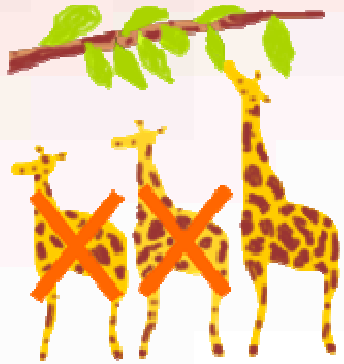
- Algorithmes génétiques
  - Gène et génotype
  - Crossing over



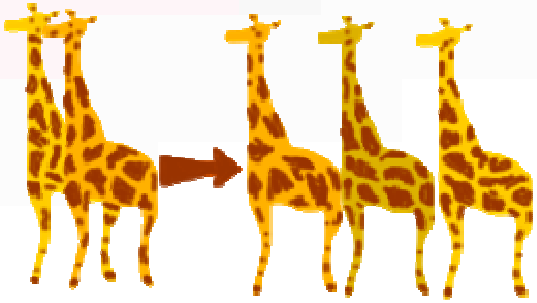


Présentation

- Algorithmes génétiques
  - Individus différents

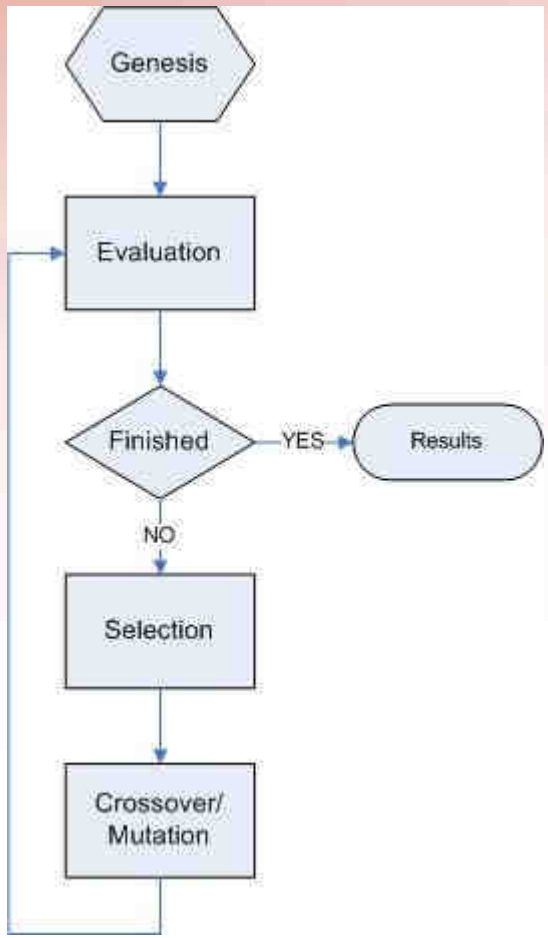


  - Sélection des mieux adaptés
  - Hérédité



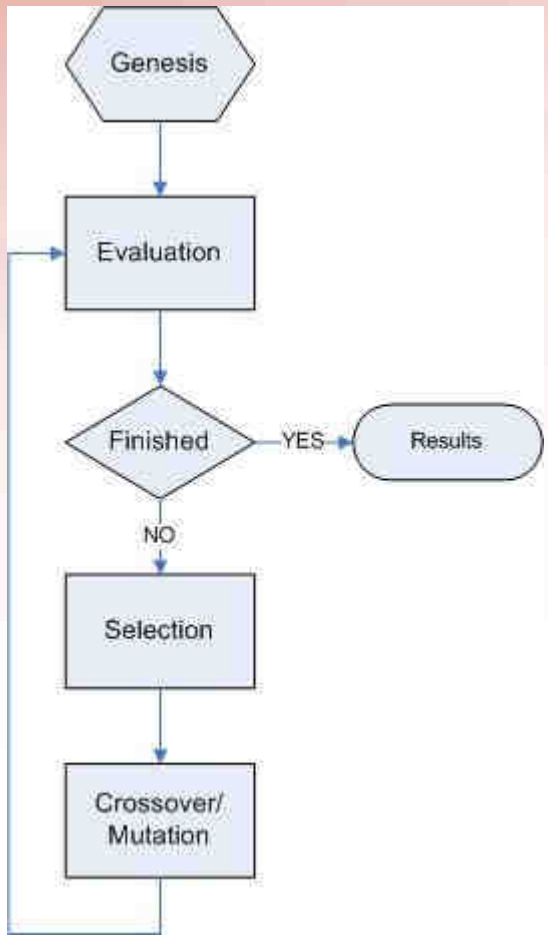
Fonctionnement

- Principes de base
  - Modélisation de la sélection naturelle
    - *Etape d'évaluation*
  - C'est donc une sélection artificielle
    - *Intervention humaine*

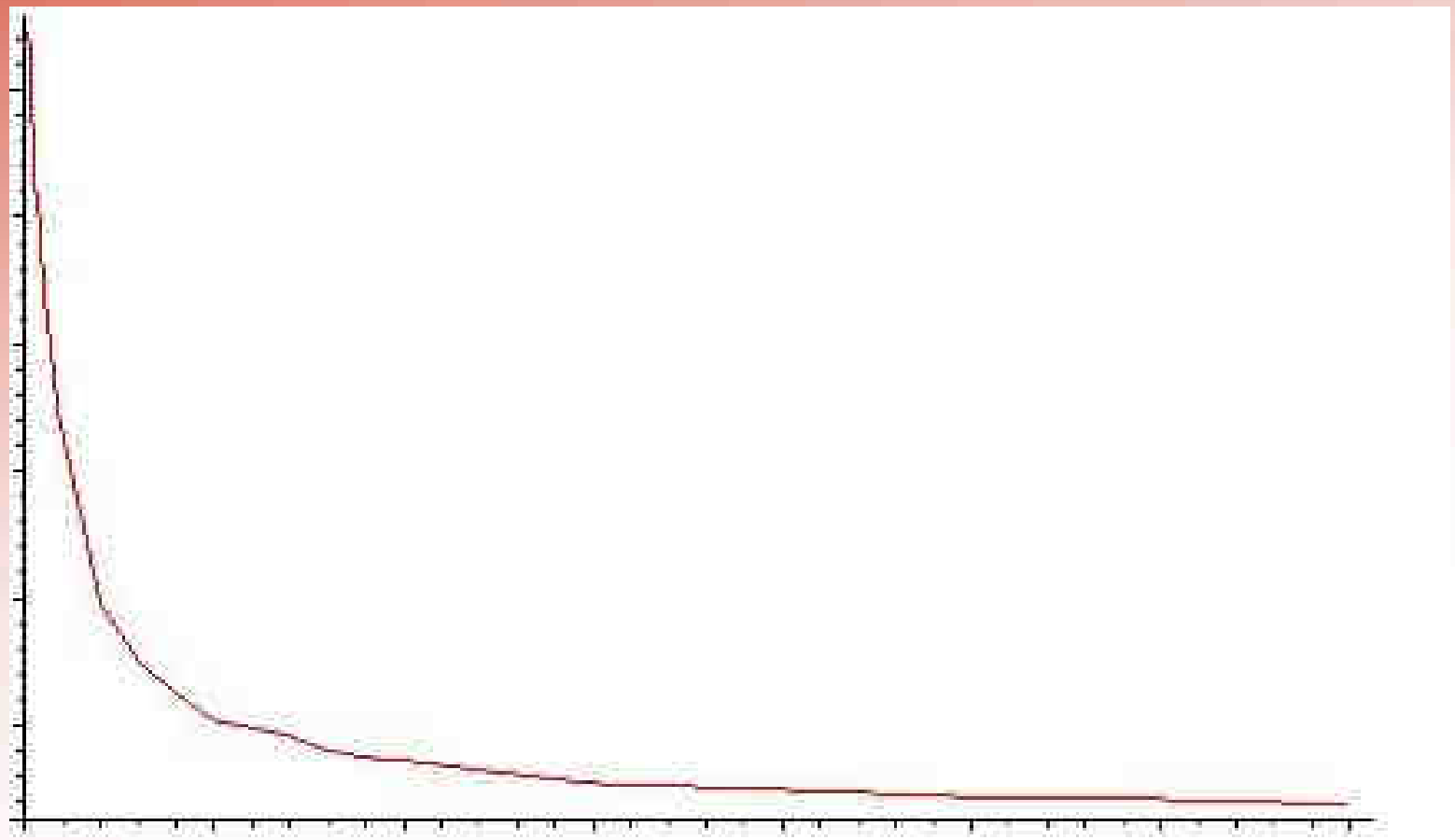


Fonctionnement

- Principes de base
  - Génération
    - *Création d'un population aléatoire*
  - Evaluation
    - *Comparaison des individus*
  - Sélection
    - *On ne garde que les meilleurs*
  - Croisement/Mutation
    - *On les fait se reproduire / Évoluer*
  - Retour à l'évaluation



Fonctionnement



Fonctionnement

- Optimisations: Implémentation
  - Gestion mémoire
    - *Problèmes:*
      - Algorithmes avec de **très nombreuses** itérations
      - Variables temporaires nombreuses
    - *Solutions:*
      - Utilisation de tampons (buffers)
      - Stratégies de réutilisation
      - Design pattern Flyweight

Utilisation

- Modélisation
  - Cas concret : Problème
    - $f(x_1, x_2, x_3, \ldots, x_n) = x_1 * x_2 + x_3 - \ldots / x_n$
    - $f(x_1, x_2, x_3, \ldots, x_n) = 1000$
    - $x_1 = ?, x_2 = ?, x_3 = ?, \ldots, x_n = ?$
    - *La suite des opérations aléatoirement choisie en début d'algorithme*
  - Comment modéliser le problème ?
    - *Que cherche-t-on ?*
    - *On définit la population, les individus, les gènes*

Utilisation

- Modélisation
  - Population
    - *Ensemble de suites de valeurs (solutions potentielles)*
  - Individu
    - *Suite de valeurs*
  - Gène
    - *Une des valeurs*
  - On définit alors les opérateurs nécessaires à l'algorithme

Utilisation

- Modélisation
  - Evaluation
    - *Calcul de f(x,y,z,…) avec les opérandes de la fonction*
    - *Ecart du résultat / résultat attendu*
  - Sélection
    - *On ne retient que les meilleurs solutions en les triant*
  - Croisement
    - *Deux suites parent donnent deux suites enfant*
    - *Le début de l'une devient le début de l'autre en coupant aléatoirement*
  - Mutation
    - *On modifie une des valeurs de certaines suites de manière aléatoire*

Utilisation

- Modélisation

```
générer population
Pour i de 0 a nb_essai_max{
    pour tous les individus{
        evaluer(individu)
    }
    Si meilleur_individu == resultat_attendu{
        arret
    }
    sélectionner(population)
    croiser(population)
    muter(population)
}
```

  - Solution = meilleur_individu
  - Il est possible de ne pas exiger l'égalité et de se contenter d'une valeur approchée

Utilisation

- Exemple JAVA avec JGAP
  - JGAP: Algorithme génétique « fin »
  - Problème:

$$f(x,y,z,\ldots) = x / y * z + \ldots$$

$$f(x,y,z,\ldots) = 10000$$

$$x = ?,\ y = ?\ z = ?,\ \ldots$$

  - Modélisation Java:
    - *CalculChromosome*
    - *CalculGene*
    - *CalculFitnessFunction*
    - *Operation*

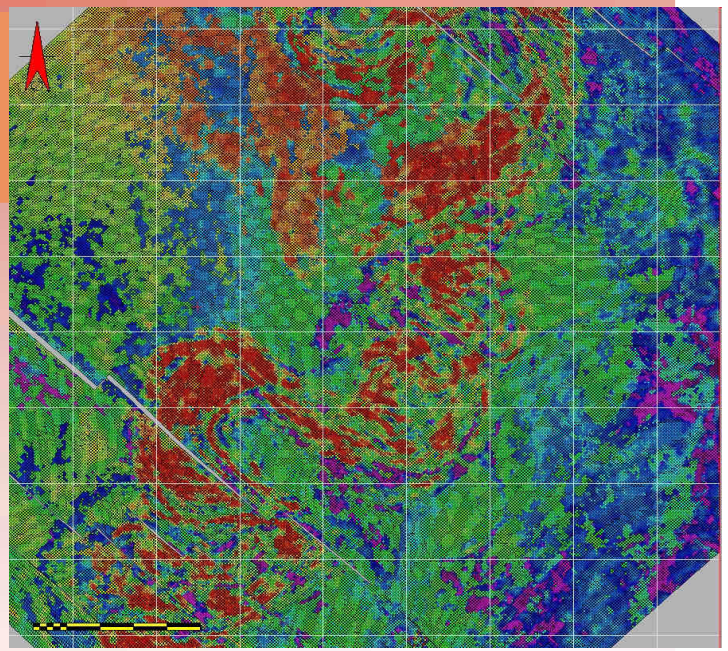
Utilisation

- Démo
  - Problème classique : le voyageur de commerce:
  - Recherche d'un cycle hamiltonien de plus courte distance dans un graphe
  - Comparaison de l'algorithme Monte Carlo avec un algorithme génétique.

Utilisation

- Démo: biobloc
  - Évolution
  - Problème: Apprendre a un robot a effectuer des opérations sans connaître ses capacités exactes.
  - Modélisation subodorée:
    - *Robot*
    - *« Biobloc »*
    - *Fonctions d'adéquations simples (meilleur performance retenue)*

Utilisation



- Cas réels
  - Total: Sismage
    - *Introduction: Réservoir pétroliers*
    - *Problème: Petro Elastic Model*

      PEM(statiques, inconnus) = IP/IS mesurés

      Inconnus = PEM_Inverse(statiques,IP/IS)

    - *Problématiques supplémentaire: performances*

Conclusion

- Conclusion
  - Algorithmes génétiques:
    - *Une solution basée sur l'optimisation*
    - *Un algorithme évolutionniste*
  - Problématiques associées
    - *Modélisation*
    - *Optimisation*
      - Algorithmique
      - Implémentation
  - Applications / Avenir