Projet de Sciences de la Décision
Ordonnancement et routing Min Σ Tj

Pierre-Antoine MORIN, Armand RENAUDEAU
DI4

Introduction

Plan

I. Présentation du problème
II. Objectifs
III. Algorithme
IV. Mise en œuvre
V. Analyse des résultats

Présentation du problème

Présentation du problème

- Situation du problème
  - Des clients
  - Des jobs
  - Des sites

Présentation du problème

- Fabrication (Flowshop)
  - 1 job = 1 machine à la fois
  - 1 machine = 1 job à la fois

Présentation du problème

- Livraison
  - La fabrication des jobs à livrer doit être achevée.
  - Possibilité de livrer plusieurs jobs dans une même tournée de livraison
  - On ne dispose que d'un seul camion pour effectuer les livraisons.

Présentation du problème

- Quelle organisation en atelier et quelles tournées de livraison effectuer afin de minimiser la somme des retards ?

- Retard = (Date de dépot chez le client - Date due)$^{+}$

Objectifs

Objectifs

- Écrire l'algorithme de l'heuristique
- Implémenter l'heuristique
- Tester l'heuristique
- Analyser les résultats

Algorithme

Algorithme

- **Algorithme de génération de la solution initiale :**

Trier les jobs selon EDD
Créer une tournée initiale contenant le premier job
Pour tous les autres jobs (dans cet ordre) ***Faire***
    Simuler l'insertion dans la dernière tournée
    Simuler la création d'une nouvelle tournée
    Si retard(création) < retard(insertion) ***Alors***
        Créer une nouvelle tournée
    Fin Si
    Ajouter le job dans la dernière tournée
Fin Pour

Algorithme

- **Algorithme de mutation :**

1) Selectionner *aléatoirement* un job
2) Changer *aléatoirement* la tournée à laquelle il appartient (+1 ou -1)
3) Générer la nouvelle solution :
    a) Ordonnancements : Algorithme de NEH
    b) Tournées : Recherche du plus proche voisin

Algorithme

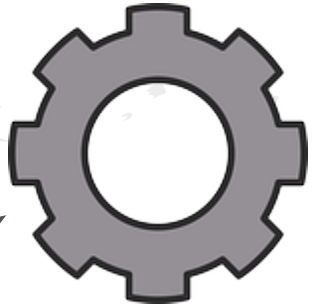
- **Algorithme de descente locale :**

solution mère ← solution initiale
Faire
    Générer *nb_mutations* solutions filles
    Si au moins une solution fille est meilleure ***Alors***
        solution mère ← meilleure solution fille
    Fin Si
Tant que au moins une solution fille est meilleure

Mise en œuvre

Programme principal

instance



programme

solution

nb_mutations
(optionnel)

Entrées

Sortie

Invocation automatisée

programme

instances

nb_mutations
(optionnel)

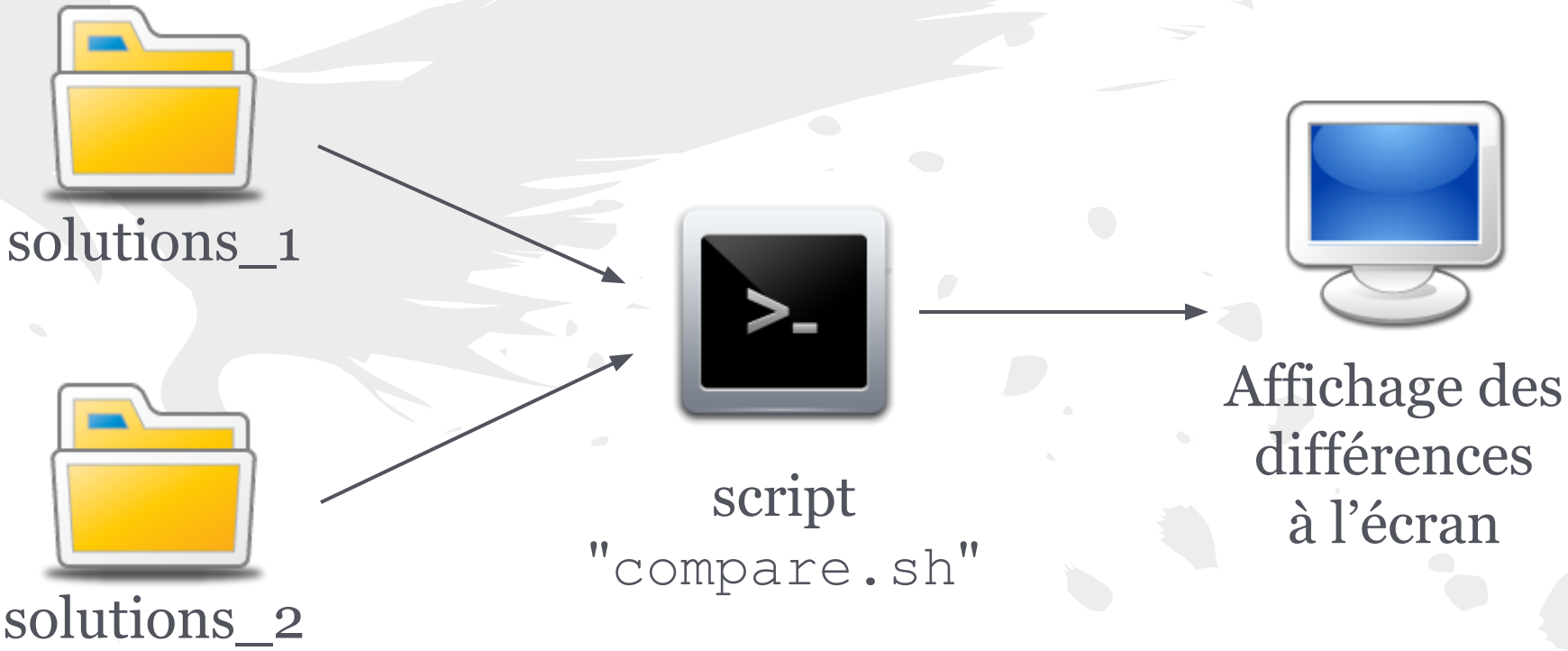
script
"call.sh"

solutions

Entrées

Sortie

Comparaisons automatisées



Entrées

Sortie

Analyse des résultats

Analyse des résultats

- Application de l'heuristique sur 160 instances (16 classes de 10 instances), avec différentes valeurs pour le nombre de mutations par itération

- Comparaisons des solutions associées à une même instance

→ Instabilité : résultats très variables

Analyse des résultats

- Lorsque le nombre de mutations augmente :
  - Les chances de se rapprocher d'une solution optimale sont plus élevées.
  - Le temps d'exécution de l'heuristique augmente plus ou moins, en fonction de la taille de l'instance.
  - L'instabilité de l'algorithme est accrue.

Conclusion

Conclusion

- Implémentation de l'algorithme réussie
- Solutions générées = minima locaux
- Instabilité de l'heuristique
- Amélioration envisageable

Merci de votre attention