Le voyageur de commerce

1

Généré par Doxygen 1.7.1

Thu Jan 10 2013 11 :42 :06

Table des matières

Chapitre 1

Le problème du voyageur de commerce - Projet d'algorithmique en langage C

Auteur

L2 Antoine de Roquemaurel (G1.1)

1.1 problème

Étant donné n points (des "villes") et les distances les séparant, trouver un chemin de longueur totale qui passe exactement une fois par chaque point et reviennent au point de départ (une tournée).

Ce problème peut servir tel quel a l'optimisation de trajectoires de machines-outils : par ex- emple, pour minimiser le temps total que met une fraiseuse a commande numérique pour percer n points dans une plaque de tôle ou pour percer les trous des composants d'un circuit electronique comme dans le cas qui nous intéresse.

Ce problème est plus compliqué qu'il n'y parait et on ne connait pas de méthode de résolution permettant d'obtenir des solutions exactes en un temps raisonnable pour de grandes instances (grand nombre de villes) du problème. Pour ces grandes instances, on devra donc souvent se contenter de solutions approchés, car on se retrouve face à une explosion combinatoire : le nombre de chemins possibles passant par 69 villes est déjà un nombre d'une longueur de 100 chiffres. Pour comparaison, un nombre d'une longueur de 80 chiffres permettrait déjà de représenter le nombre d'atomes dans tout l'univers connu.

Le problème du "voyageur de commerce" a été étudié depuis lontemps et on dispose d'une grande variété d'algorithmes donnant le plus souvent des solutions approchés mais calculables en un temps raisonnable.

1.2 algorithmes implémentéé

Ce problème sera implémenté via différents algorithmes :
- Brute force
- Recherche locale aléatorie
- Recherche loacle systématique
- Algorithme génétique

Chapitre 2

Index des structures de données

2.1 Structures de données

Liste des structures de données avec une brève description :

Chapitre 3

Index des fichiers

3.1 Liste des fichiers

Liste de tous les fichiers documentés avec une brève description :

Chapitre 4

Documentation des structures de données

4.1 Référence de la structure Algo

Enumération d'un algorithme.

`#include <parsing.h>`

Champs de données

- AlgoType type
  Le type de l'algorithme.
- int firstParameter
  Premier paramètre de l'algorithme.
- int secondParameter
  Second paramètre.

4.1.1 Description détaillée

Enumération d'un algorithme. Énumération d'un algorithme, contient le type de l'algorithme avec les eventuels paramètres

La documentation de cette structure a été générée à partir du fichier suivant :
- lib/parsing.h

4.2 Référence de la structure Distance

Objet des distances.

`#include <distance.h>`

Champs de données

– Town firstTown
Première ville.

– Town secondTown
Seconde ville.

– double distance
Distance entre les deux villes.

4.2.1 Description détaillée

Objet des distances. Distance entre deux villes

Voir également

Town

La documentation de cette structure a été générée à partir du fichier suivant :
– lib/distance.h

4.3 Référence de la structure Errors

Objet des erreurs.

`#include <errors.h>`

Champs de données

– char * errorNbArguments
Nombre d'argument incorrect.

– char * errorTagFNotFound
Tag -f non trouvé.

– char * errorFileNotFound
Fichier non trouvé.

– char * errorNoAlgoSpecified
Algorithme non spécifié.

– char * errorMissingParameterLsr
Paramètre après -lsr manquant.

– char * errorMissingParameterLsnr
Paramètre après -lsnr manquant.

– char * errorMissingParameterGa
Paramètre après -ga manquant.

– char * errorNoValidParameterLsr
Paramètre après -lsr non valide.

– char * errorNoValidParameterLsnr
Paramètre après -lsnr non valide.

– char * errorNoValidParameterGa
Paramètre après -ga non valide.

– int nbErrors
Nombre d'erreurs.

4.3.1 Description détaillée

Objet des erreurs. Toutes les chaines de caractères des erreurs. Si une variable vaut NULL, l'erreur n'est pas présente, sinon elle sera affiché. nbErrors est le nombre d'erreur, si celui-ci =0 alors le programme peut fonctionner correctement

La documentation de cette structure a été générée à partir du fichier suivant :
– lib/errors.h

4.4 Référence de la structure Instance

Objet des instances.

`#include <instance.h>`

Champs de données

– Town towns [N]
Tableau des villes(Town) classés par ID.

– Distance distances [N]
Tableau linéaire contenant toutes les distances entre les villes.

– int nbTowns
Nombre de ville de l' Instance.

– char * name
Nom de l' Instance.

– char * type
Type de l' Instance.

4.4.1 Description détaillée

Objet des instances.

Voir également

Town
Distance

Une instance contient toutes les villes (Town) classés par ID, les calculs des algorithmes utilisent une instance afin d'en retourner la meilleur tournée.

La documentation de cette structure a été générée à partir du fichier suivant :
– lib/instance.h

4.5 Référence de la structure Tour

Objet d'une tournée.

`#include <tour.h>`

Champs de données

– Town towns [N]
Tableau de ville. Les villes sont triés dans l'ordre de la tournée.

– int nbTowns
Nombre de ville de la tournée.

– double length
Longueur de la tournée.

– Distance * distances
Matrice de distances.

4.5.1 Description détaillée

Objet d'une tournée. Informations concernant une tournée.

La documentation de cette structure a été générée à partir du fichier suivant :
– lib/tour.h

4.6 Référence de la structure Town

Objet des ville.

`#include <town.h>`

Champs de données

– int x
Abscisse de la ville.

– int y
Ordonnée de la ville.

– int id
Id de la ville.

4.6.1 Description détaillée

Objet des ville. Structure représentant une ville

La documentation de cette structure a été générée à partir du fichier suivant :
– lib/town.h

Chapitre 5

Documentation des fichiers

5.1 Référence du fichier lib/bruteForce.h

Fonctions de brute force.

```
#include "instance.h"
#include "tour.h"
#include "util.h"
```

Fonctions

– Tour bruteForce_bestPath (Instance pInstance)
Permet d'obtenir le meilleur chemin d'une instance via la force brute.

5.1.1 Description détaillée

Fonctions de brute force.
Auteur

Antoine de Roquemaurel

Date

27/12/2012 17 :58 :36

Entêtes des fonctions servant à la force brute. Ces fonctions ne sont appelés que depuis le main

5.1.2 Documentation des fonctions

5.1.2.1 Tour bruteForce_bestPath (Instance *pInstance*)

Permet d'obtenir le meilleur chemin d'une instance via la force brute.

Il est conseillé de ne pas essayer avec des instances de plus de 8 villes.

Paramètres

pInstance L'instance pour laquelle on doit calculer le plus court chemin

Renvoie

La meilleur tournée.

5.2 Référence du fichier lib/distance.h

Fonctions des distances.

```
#include <math.h>
#include "town.h"
```

Structures de données

– struct Distance
Objet des distances.

Fonctions

– Distance distance_new (Town ∗pFirstTown, Town ∗pSecondTown)
Créer une nouvelle distance.

– double distance_calculDistance (const Town pTown1, const Town pTown2)
Calcul la distance entre deux villes.

– double distance_betweenTowns (Distance ∗pDistances, int i, int j)
Calcul la distance entre deux ID de villes.

– Distance distance_searchDistance (Distance ∗pDistances, const int pFirst, const int pSecond)
Recherche la distance entre deux villes dans le tableau de distance.

5.2.1 Description détaillée

Fonctions des distances.
Auteur

Antoine de Roquemaurel

Date

01/12/2012 20 :33 :44

Entêtes des fonctions se rapportant aux distances

5.2.2 Documentation des fonctions

5.2.2.1 double distance_betweenTowns (Distance ∗ *pDistances,* int *i,* int *j*)

Calcul la distance entre deux ID de villes.

Paramètres

pDistances La matrice de distances

i La première ville

j La seconde ville

Renvoie

La distance

5.2.2.2 double distance_calculDistance (const Town *pTown1*, const Town *pTown2*)

Calcul la distance entre deux villes.

Paramètres

pTown1 Première ville

pTown2 Seconde ville

Renvoie

La distance

5.2.2.3 Distance distance_new (Town * *pFirstTown*, Town * *pSecondTown*)

Créer une nouvelle distance.

Paramètres

pFirstTown Première ville

pSecondTown Seconde ville

Renvoie

Distance entre les deux villes

5.2.2.4 Distance distance_searchDistance (Distance * *pDistances*, const int *pFirst*, const int *pSecond*)

Recherche la distance entre deux villes dans le tableau de distance.

Paramètres

pDistances La matrice de distance

pFirst La première ville

pSecond la seconde ville

Renvoie

La distance

5.3 Référence du fichier lib/errors.h

Fonctions sur les erreurs.

```
#include <stdio.h>
#include <stdlib.h>
```

Structures de données

– struct Errors
Objet des erreurs.

Fonctions

– Errors errors_new ()
Initialisation de l'objet.

– void errors_displayErrorsMessage (const Errors pErrors)
Affiches toutes les erreurs de pErrors.

– void errors_setNbArguments (Errors *pErrors)
Signale que le nombre d'argument est incorrect.

– void errors_setTagFNotFound (Errors *pErrors)
Signale que la balise -f n'a pas été trouvée.

– void errors_setFileNotFound (Errors *pErrors, char *fileName)
Signale que le fichier fileName n'existe pas.

– void errors_setNoAlgoSpecified (Errors *pErrors)
Signale qu'aucun algorithme n'a été spécifié.

– void errors_setNoValidParameterLsnr (Errors *pErrors)
Signale que les paramètres derrière -lsnr ne sont pas valides (non entier).

– void errors_setNoValidParameterLsr (Errors *pErrors)
Signale que le paramètre derrière -lsr n'est pas valide (n'est pas un entier).

– void errors_setMissingParameterGa (Errors *pErrors)
Signale qu'aucun paramètre n'a été spécifié derrière -ga.

– void errors_setMissingParameterLsnr (Errors *pErrors)
Signale qu'aucun paramètre n'a été spécifié derrière -lsnr.

– void errors_setMissingParameterLsr (Errors *pErrors)
Signale qu'aucun paramètre n'a été spécifié derrière -lsr.

5.3.1 Description détaillée

Fonctions sur les erreurs.

Auteur

Antoine de Roquemaurel

Date

21/11/2012 17 :42 :37

Entêtes des fonctions concernant les erreurs du programme.

5.3.2 Documentation des fonctions

5.3.2.1 void errors_displayErrorsMessage (const Errors *pErrors*)

Affiches toutes les erreurs de pErrors.

Paramètres

pErrors L'objet pour lequelle on doit afficher les erreurs

5.3.2.2 Errors errors_new ( )

Initialisation de l'objet.

Renvoie

Une instance de Erreur initialisée

5.3.2.3 void errors_setFileNotFound (Errors * *pErrors*, char * *fileName*)

Signale que le fichier fileName n'existe pas.

Paramètres

pErrors L'objet des erreurs

fileName Le nom de fichier non trouvé

5.3.2.4 void errors_setMissingParameterGa (Errors * *pErrors*)

Signale qu'aucun paramètre n'a été spécifié derrière -ga.

Paramètres

pErrors L'objet des erreurs

5.3.2.5 void errors_setMissingParameterLsnr (Errors * *pErrors*)

Signale qu'aucun paramètre n'a été spécifié derrière -lsnr.

Paramètres

pErrors L'objet des erreurs

5.3.2.6 void errors_setMissingParameterLsr (Errors * *pErrors*)

Signale qu'aucun paramètre n'a été spécifié derrière -lsr.

Paramètres

pErrors L'objet des erreurs

5.3.2.7 void errors_setNbArguments (Errors * *pErrors*)

Signale que le nombre d'argument est incorrect.

Paramètres

pErrors L'objet des erreurs

5.3.2.8 void errors_setNoAlgoSpecified (Errors * *pErrors*)

Signale qu'aucun algorithme n'a été spécifié.

Paramètres

pErrors L'objet des erreurs

5.3.2.9 void errors_setNoValidParameterLsnr (Errors * *pErrors*)

Signale que les paramètres derrière -lsnr ne sont pas valides (non entier).

Paramètres

pErrors L'objet des erreurs

5.3.2.10 void errors_setNoValidParameterLsr (Errors * *pErrors*)

Signale que le paramètre derrière -lsr n'est pas valide (n'est pas un entier).

Paramètres

pErrors L'objet des erreurs

5.3.2.11 void errors_setTagFNotFound (Errors * *pErrors*)

Signale que la balise -f n'a pas été trouvée.

Paramètres

pErrors L'objet des erreurs

5.4 Référence du fichier lib/instance.h

Fonctions sur les instances.

```
#include <stdbool.h>
#include <string.h>
#include "town.h"
#include "distance.h"
```

Structures de données

– struct Instance
Objet des instances.

Macros

– #define N 1024
Taille maximale des tableaux.

Fonctions

– void instance_display (const Instance pInstance)
Affiche une Instance.

– Instance instance_new (FILE ∗pFile)
Créer une nouvelle Instance à partir d'un fichier.

– void instance_push (Instance ∗pInstance, const Town pTown)
Ajoute une nouvelle ville dans une Instance.

– void instance_initializeDistancesMatrix (Instance ∗pInstance)
Initialise la matrice des distances.

– void instance_displayLinearVector (Instance pInstance)
Affiche la matrice des distances sous forme linéaire.

– void instance_displayMatrix (Instance pInstance)
Affiche la matrice des distances sous forme de matrice symétrique.

5.4.1 Description détaillée

Fonctions sur les instances.
Auteur

Antoine de Roquemaurel

Date

21/11/2012 22 :03 :34

Toutes les entêtes des fonctions se rapportant à une instance.

5.4.2 Documentation des fonctions

5.4.2.1 void instance_display (const Instance *pInstance*)

Affiche une Instance.

Paramètres

pInstance l' Instance à afficher

5.4.2.2 void instance_displayLinearVector (Instance *pInstance*)

Affiche la matrice des distances sous forme linéaire.

Paramètres

pInstance L'instance à afficher

Voir également

Distance

5.4.2.3 void instance_displayMatrix (Instance *pInstance*)

Affiche la matrice des distances sous forme de matrice symétrique.

Paramètres

pInstance L'instance à afficher

Voir également

Distance

5.4.2.4 void instance_initializeDistancesMatrix (Instance * *pInstance*)

Initialise la matrice des distances.

Paramètres

pInstance L'instance à modifier

Voir également

Distance

5.4.2.5 Instance instance_new (FILE * *pFile*)

Créer une nouvelle Instance à partir d'un fichier.

Paramètres

pFile Le pointeur sur fichier contenant les informations de l'instance

Renvoie

la nouvelle Instance

5.4.2.6 void instance_push (Instance * *pInstance*, const Town *pTown*)

Ajoute une nouvelle ville dans une Instance.

Paramètres

pInstance L'instance à modifier

pTown La ville à ajouter

5.5 Référence du fichier lib/localSearch.h

Fonctions utiles.

```
#include "tour.h"
#include <time.h>
#include <stdlib.h>
```

Fonctions

– Tour **localSearch_randomBestPath** (Instance pInstance, int pTryNb)
– Tour **localSearch_systematicBestPath** (Instance pInstance, int pTryNb)

5.5.1 Description détaillée

Fonctions utiles. Fonctions de recherche locale.

Auteur

Antoine de Roquemaurel

Date

27/12/2012 18 :00 :30

Entêtes des fonctions pouvant être utiles dans tout le projet. Ces fonctions ne sont appelés que depuis le main

Auteur

Antoine de Roquemaurel

Date

27/12/2012 18 :00 :18

Entêtes des fonctions ayant rapport avec les algorithmes de recherches locales.

5.6 Référence du fichier lib/parsing.h

Fonctions de parsing des arguments.

```
#include <stdbool.h>
#include "util.h"
#include "errors.h"
```

Structures de données

– struct Algo
Enumération d'un algorithme.

Énumérations

– enum AlgoType {
BRUTEFORCE, LOCALSEARCH_RANDOM, LOCALSEARCH_SYSTEMATIC, GENETIC,
END }
parsing.h

Fonctions

– bool parsing_parseVerboseMode (char **pTab, const int pSize)
Cherche si le mode verbeux à été spécifié ou non.

– char * parsing_parseFileName (char **pTab, const int pSize, Errors *pErrors)
Cherche le nom du fichier de l'Instance.

– void parsing_algoType (char **pTab, const int pSize, Errors *pErrors, Algo *algos)

5.6.1 Description détaillée

Fonctions de parsing des arguments.
Auteur

Antoine de Roquemaurel

Date

21/11/2012 17 :17 :24

Entêtes des fonctions permettant de parser les arguments, et ainsi d'appeler les différents algorithmes demandés, d'utiliser le mode verbeux et de spécifier le fichier.

5.6.2 Documentation du type de l'énumération

5.6.2.1 enum AlgoType

parsing.h

Enumération des types d'algorithmes Les différents types d'algorithmes qui peuvent être appellés.
Valeurs énumérées :

BRUTEFORCE L'algorithme de brute force.

LOCALSEARCH_RANDOM L'algorithme de recherche locale aléatoire.

LOCALSEARCH_SYSTEMATIC L'algorithme de recherche locale systématique.

GENETIC L'algorithme génétique.

END Correspond au marqueur de fin des algorithmes.

5.6.3 Documentation des fonctions

5.6.3.1 void parsing_algoType (char ** *pTab*, const int *pSize*, Errors * *pErrors*, Algo * *algos*)

Paramètres

pTab Le tableau contenant les arguments

pSize Le nombre des arguments

pErrors L'objet des erreurs, il est modifié si des erreurs interviennent

algos Tableau d'algorithmes, ceci au cas ou l'utilisateur entre plusieurs algorithmes. La fin du tableau est marqué par END

Voir également

Algo

5.6.3.2 char* parsing_parseFileName (char ** *pTab*, const int *pSize*, Errors * *pErrors*)

Cherche le nom du fichier de l'Instance.

Paramètres

pTab Le tableau contenant les arguments

pSize Le nombre des arguments

pErrors L'objet des erreurs, il est modifié si des erreurs interviennent

Renvoie

Le nom du fichier

Voir également

Instance

5.6.3.3 bool parsing_parseVerboseMode (char ** *pTab*, const int *pSize*)

Cherche si le mode verbeux à été spécifié ou non.

Paramètres

pTab Le tableau contenant les arguments

pSize Le nombre des arguments

Renvoie

Vrai si mode verbeux, Faux sinon

5.7 Référence du fichier lib/tour.h

Fonctions des tournées.

```
#include <stdlib.h>
#include <stdbool.h>
#include <time.h>
#include "town.h"
#include "instance.h"
#include "distance.h"
```

Structures de données

– struct Tour
Objet d'une tournée.

Fonctions

– Tour tour_new (Instance pInstance)
Créer une nouvelle tournée initialisée avec les données d'une instance.

– bool tour_nextPermutation (Tour ∗pPermutation)
Génère la permutation de ville suivante d'une tournée.

– void tour_calculLength (Tour ∗pTour)
Calcul la longueur d'une tournée.

– void tour_display (const Tour pTour)
Affiche une tournée.

– Tour tour_randomWalk (const Instance pInstance)
Génère une tournée aléatoire.

– void tour_2opt (Tour ∗pTour, int pFirst, int pSecond)
Fait une 2opt.

5.7.1 Description détaillée

Fonctions des tournées.
Auteur

Antoine de Roquemaurel

Date

21/11/2012 22 :04 :13

Entêtes des fonctions se rapportant à une tournée. ∗

5.7.2 Documentation des fonctions

5.7.2.1 void tour_2opt (Tour ∗ *pTour,* int *pFirst,* int *pSecond*)

Fait une 2opt.

Paramètres

pTour Le tour pourlaquelle on veut faire une 2opt

pFirst L'id du début du premier trajet

pSecond L'id du début du second trajet

5.7.2.2 void tour_calculLength (Tour ∗ *pTour*)

Calcul la longueur d'une tournée.

Paramètres

pTour La tournée pourlaquelle on veut calculer la longueur

5.7.2.3 void tour_display (const Tour *pTour*)

Affiche une tournée.

Paramètres

pTour La tournée à afficher

5.7.2.4 Tour tour_new (Instance *pInstance*)

Créer une nouvelle tournée initialisée avec les données d'une instance.

Paramètres

pInstance Instance servant à initialisée la tournée

Renvoie

la nouvelle tournée

5.7.2.5 bool tour_nextPermutation (Tour * *pPermutation*)

Génère la permutation de ville suivante d'une tournée.

Paramètres

pPermutation La tournée pour laquelle la permutation doit être générée

Renvoie

Vrai si une permutation à été généré faux s'il ne reste plus de permutation.

5.7.2.6 Tour tour_randomWalk (const Instance *pInstance*)

Génère une tournée aléatoire.

Paramètres

pInstance L'instance pour laquelle générer un random walk

Renvoie

La tournée aléatoire

5.8 Référence du fichier lib/town.h

Fonctions des villes.

```
#include <stdio.h>
#include <stdlib.h>
```

Structures de données

– struct Town
Objet des ville.

Fonctions

– Town town_new (const int pId, const int pX, const int pY)
Création d'une nouvelle ville.

5.8.1 Description détaillée

Fonctions des villes.
Auteur

Antoine de Roquemaurel

Date

21/11/2012 22 :35 :19

Entêtes des fonctions se rapportant à une ville.

5.8.2 Documentation des fonctions

5.8.2.1 Town town_new (const int *pId*, const int *pX*, const int *pY*)

Création d'une nouvelle ville.

Paramètres

pId Id de la ville

pX Abscisse

pY Ordonnée

5.9 Référence du fichier lib/util.h

Fonctions utiles.

```
#include <stdbool.h>
#include <stdio.h>
#include <string.h>
#include "tour.h"
```

Fonctions

– int util_searchFirstOccurenceInArray (char **pArray, const int pSize, char *pSearch)
Cherche la première occurence d'une chaine de caractère dans un tableau de chaine caractère.

- void util_reverseArray (Town *pTab, const int pBegin, const int pEnd)
  Inverse les éléments d'un tableau pTag entre les cases pBegin et pEnd.
- void util_displayArray (const int *pTab, const int pSize)
  Affiche le contenu d'un tableau d'entiers.
- int util_sum (const int pBegin, const int pEnd)
  Calcul la somme des éléments allant de pBegin à pEnd.
- void util_swap (int *a, int *b)
  Échange deux variables.
- int util_rand (const int pMin, const int pMax)
  Calcul une valeur aléatoire entre pMin et pMax.
- bool util_sousTabExist (Tour pChild, const int pBegin, const int pEnd, Tour pParent, bool pRecursvite)
  Indique si un sous-tableau d'un tableau est présent dans un autre tableau.

Variables

- bool gVerboseMode
  Mode verbose.

5.9.1 Description détaillée

Fonctions utiles.
Auteur

Antoine de Roquemaurel

Date

19/11/2012 16 :27 :39

Entêtes des fonctions pouvant être utiles dans tout le projet. Ce sont des fonctions simples, qui doivent être indépendantes du projet.

5.9.2 Documentation des fonctions

5.9.2.1 void util_displayArray (const int * *pTab*, const int *pSize*)

Affiche le contenu d'un tableau d'entiers.

Paramètres

pTab Le tableau à afficher

pSize La taille du tableau

5.9.2.2 int util_rand (const int *pMin*, const int *pMax*)

Calcul une valeur aléatoire entre pMin et pMax.

Paramètres

pMin Minimum

pMax Maximum

Renvoie

5.9.2.3 void util_reverseArray (Town * *pTab*, const int *pBegin*, const int *pEnd*)

Inverse les éléments d'un tableau pTag entre les cases pBegin et pEnd.

Paramètres

pTab Tableau à inverser

pBegin Début de la section à inverser

pEnd Fin de la section à inverser

5.9.2.4 int util_searchFirstOccurenceInArray (char ** *pArray*, const int *pSize*, char * *pSearch*)

Cherche la première occurence d'une chaine de caractère dans un tableau de chaine caractère.

Paramètres

pArray Le tableau de chaines de caractères dans lequel chercher

pSize La taille du tableau

pSearch La chaine de caractère à chercher

Renvoie

La position de la chaine dans le tableau ou -1 si elle n'a pas été trouvée

5.9.2.5 bool util_sousTabExist (Tour *pChild*, const int *pBegin*, const int *pEnd*, Tour *pParent*, bool *pRecursvite*)

Indique si un sous-tableau d'un tableau est présent dans un autre tableau.

Paramètres

pChild Le tableau à chercher

pBegin Le début de la section à chercher

pEnd La fin de la section à chercher

pParent Le tableau dans lequel chercher

pRecursvite Pour simplifier vis-à-vis de la récursivité, doit toujours être à false

Renvoie

Vrai si le sous-tableau à été trouvé faux sinon

5.9.2.6 int util_sum (const int *pBegin*, const int *pEnd*)

Calcul la somme des éléments allant de pBegin à pEnd.

Paramètres

pBegin Début de la somme

pEnd Fin de la somme

Renvoie

La somme des éléments

5.9.2.7 void util_swap (int * *a*, int * *b*)

Échange deux variables.

Paramètres

a Première variable à échanger

b Seconde variable

5.10 Référence du fichier src/bruteForce.c

Fonctions de brute force.

```
#include "bruteForce.h"
#include "util.h"
```

Fonctions

– Tour bruteForce_bestPath (Instance pInstance)
Permet d'obtenir le meilleur chemin d'une instance via la force brute.

5.10.1 Description détaillée

Fonctions de brute force. Fonctions sur les erreurs.

Auteur

Antoine de Roquemaurel

Date

27/12/2012 17 :58 :29

Implémentation des fonctions utilisant la force brute. Ces fonctions ne sont appelés que depuis le main

Auteur

Antoine de Roquemaurel

Date

21/11/2012 17 :42 :31

Implémentation des fonctions concernant les erreurs du programme.

5.10.2 Documentation des fonctions

5.10.2.1 Tour bruteForce_bestPath (Instance *pInstance*)

Permet d'obtenir le meilleur chemin d'une instance via la force brute.

Il est conseillé de ne pas essayer avec des instances de plus de 8 villes.

Paramètres

pInstance L'instance pour laquelle on doit calculer le plus court chemin

Renvoie

La meilleur tournée.

5.11 Référence du fichier src/distance.c

Fonctions des distances.

```
#include "distance.h"
#include "util.h"
```

Fonctions

– Distance distance_new (Town ∗pFirstTown, Town ∗pSecondTown)
Créer une nouvelle distance.

– double distance_calculDistance (const Town pTown1, const Town pTown2)
Calcul la distance entre deux villes.

– Distance distance_searchDistance (Distance ∗pDistances, const int pFirst, const int pSecond)
Recherche la distance entre deux villes dans le tableau de distance.

– double distance_betweenTowns (Distance ∗pDistances, int i, int j)
Calcul la distance entre deux ID de villes.

5.11.1 Description détaillée

Fonctions des distances.
Auteur

Antoine de Roquemaurel

Date

01/12/2012 20 :33 :39

Entêtes des fonctions se rapportant aux distances

5.11.2 Documentation des fonctions

5.11.2.1 double distance_betweenTowns (Distance * *pDistances,* int *i,* int *j*)

Calcul la distance entre deux ID de villes.

Paramètres

pDistances La matrice de distances

i La première ville

j La seconde ville

Renvoie

La distance

5.11.2.2 double distance_calculDistance (const Town *pTown1,* const Town *pTown2*)

Calcul la distance entre deux villes.

Paramètres

pTown1 Première ville

pTown2 Seconde ville

Renvoie

La distance

5.11.2.3 Distance distance_new (Town * *pFirstTown,* Town * *pSecondTown*)

Créer une nouvelle distance.

Paramètres

pFirstTown Première ville

pSecondTown Seconde ville

Renvoie

Distance entre les deux villes

5.11.2.4 Distance distance_searchDistance (Distance * *pDistances,* const int *pFirst,* const int *pSecond*)

Recherche la distance entre deux villes dans le tableau de distance.

Paramètres

pDistances La matrice de distance

pFirst La première ville

pSecond la seconde ville

Renvoie

La distance

5.12 Référence du fichier src/genetic.c

Fonctions d'algorithmes génétiques.

```
#include "genetic.h"
#include "tour.h"
```

Fonctions

– Tour **genetic_distancePreservingCrossover** (Tour pParent1, Tour pParent2)

5.12.1 Description détaillée

Fonctions d'algorithmes génétiques.

Auteur

Antoine de Roquemaurel

Date

27/12/2012 18 :00 :25

Implémentation des fonctions servant aux algorithmes génétiques. Ces fonctions ne sont appelés que depuis le main

5.13 Référence du fichier src/instance.c

Fonctions utiles.

```
#include "instance.h"
```

Fonctions

– Instance instance_new (FILE ∗pFile)
Créer une nouvelle Instance à partir d'un fichier.

– void instance_display (const Instance pInstance)
Affiche une Instance.

– void instance_push (Instance ∗pInstance, const Town pTown)
Ajoute une nouvelle ville dans une Instance.

– void instance_initializeDistancesMatrix (Instance ∗pInstance)
Initialise la matrice des distances.

– void instance_displayLinearVector (Instance pInstance)
Affiche la matrice des distances sous forme linéaire.

– void instance_displayMatrix (Instance pInstance)
Affiche la matrice des distances sous forme de matrice symétrique.

5.13.1 Description détaillée

Fonctions utiles.
Auteur

Antoine de Roquemaurel

Date

21/11/2012 22 :03 :26

Entêtes des fonctions pouvant être utiles dans tout le projet. Ce sont des fonctions simples, qui doivent être indépendantes du projet.

5.13.2 Documentation des fonctions

5.13.2.1 void instance_display (const Instance *pInstance*)

Affiche une Instance.

Paramètres

pInstance l' Instance à afficher

5.13.2.2 void instance_displayLinearVector (Instance *pInstance*)

Affiche la matrice des distances sous forme linéaire.

Paramètres

pInstance L'instance à afficher

Voir également

Distance

5.13.2.3 void instance_displayMatrix (Instance *pInstance*)

Affiche la matrice des distances sous forme de matrice symétrique.

Paramètres

pInstance L'instance à afficher

Voir également

Distance

5.13.2.4 void instance_initializeDistancesMatrix (Instance * *pInstance*)

Initialise la matrice des distances.

Paramètres

pInstance L'instance à modifier

Voir également

Distance

5.13.2.5 Instance instance_new (FILE * *pFile*)

Créer une nouvelle Instance à partir d'un fichier.

Paramètres

pFile Le pointeur sur fichier contenant les informations de l'instance

Renvoie

la nouvelle Instance

5.13.2.6 void instance_push (Instance * *pInstance,* const Town *pTown*)

Ajoute une nouvelle ville dans une Instance.

Paramètres

pInstance L'instance à modifier

pTown La ville à ajouter

5.14 Référence du fichier src/localSearch.c

Fonctions de recherche locale.

```
#include "util.h"
#include "localSearch.h"
#include "instance.h"
#include "tour.h"
```

Fonctions

– Tour **localSearch_randomBestPath** (Instance pInstance, int pTryNb)
– Tour **localSearch_systematicBestPath** (Instance pInstance, int pTryNb)

5.14.1 Description détaillée

Fonctions de recherche locale.
Auteur

Antoine de Roquemaurel

Date

27/12/2012 18 :00 :13

Entêtes des fonctions ayant rapport avec les algorithmes de recherches locales.

5.15 Référence du fichier src/main.c

Fonction main.

```
#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>
#include "parsing.h"
#include "errors.h"
#include "bruteForce.h"
#include "localSearch.h"
```

Fonctions

– int main (int argc, char ∗∗argv)
Fonction d'entrée du programme.

5.15.1 Description détaillée

Fonction main.
Auteur

Antoine de Roquemaurel

Date

19/11/2012 10 :42 :29

Point d'entrée du programme. Aucune fonction ne doit être déclarée Ce sont des fonctions simples, qui doivent être indépendantes du projet.

5.15.2 Documentation des fonctions

5.15.2.1 int main (int *argc,* char ∗∗ *argv*)

Fonction d'entrée du programme.

Paramètres

argc Nombre d'arguments du programme
argv Tableau contenant la liste des arguments du programme

Renvoie

Code d'erreur du programme

5.16 Référence du fichier src/parsing.c

Fonctions de parsing des arguments.

```
#include "parsing.h"
```

Fonctions

– bool parsing_parseVerboseMode (char **pTab, const int pSize)
Cherche si le mode verbeux à été spécifié ou non.

– char * parsing_parseFileName (char **pTab, const int pSize, Errors *pErrors)
Cherche le nom du fichier de l'Instance.

– void parsing_algoType (char **pTab, const int pSize, Errors *pErrors, Algo *algos)

5.16.1 Description détaillée

Fonctions de parsing des arguments.
Auteur

Antoine de Roquemaurel

Date

21/11/2012 17 :17 :18

Implémentation des fonctions permettant de parser les arguments, et ainsi d'appeler les différents algorithmes demandés, d'utiliser le mode verbeux et de spécifier le fichier.

5.16.2 Documentation des fonctions

5.16.2.1 void parsing_algoType (char ** *pTab*, const int *pSize*, Errors * *pErrors*, Algo * *algos*)

Paramètres

pTab Le tableau contenant les arguments
pSize Le nombre des arguments
pErrors L'objet des erreurs, il est modifié si des erreurs interviennent
algos Tableau d'algorithmes, ceci au cas ou l'utilisateur entre plusieurs algorithmes. La fin du tableau est marqué par END

Voir également

Algo

5.16.2.2 char* parsing_parseFileName (char ** *pTab*, const int *pSize*, Errors * *pErrors*)

Cherche le nom du fichier de l'Instance.

Paramètres

pTab Le tableau contenant les arguments
pSize Le nombre des arguments
pErrors L'objet des erreurs, il est modifié si des erreurs interviennent

Renvoie

Le nom du fichier

Voir également

Instance

5.16.2.3 bool parsing_parseVerboseMode (char ** *pTab*, const int *pSize*)

Cherche si le mode verbeux à été spécifié ou non.

Paramètres

pTab Le tableau contenant les arguments

pSize Le nombre des arguments

Renvoie

Vrai si mode verbeux, Faux sinon

5.17 Référence du fichier src/tour.c

Fonctions des tournées.

```
#include "tour.h"
```

Fonctions

- Tour tour_new (Instance pInstance)
  Créer une nouvelle tournée initialisée avec les données d'une instance.
- bool tour_nextPermutation (Tour *pPermutation)
  Génère la permutation de ville suivante d'une tournée.
- void tour_calculLength (Tour *pTour)
  Calcul la longueur d'une tournée.
- void tour_display (const Tour pTour)
  Affiche une tournée.
- Tour tour_randomWalk (const Instance pInstance)
  Génère une tournée aléatoire.
- void tour_2opt (Tour *pTour, int pFirst, int pSecond)
  Fait une 2opt.

5.17.1 Description détaillée

Fonctions des tournées.

Auteur

Antoine de Roquemaurel

Date

21/11/2012 22 :04 :08

Implémentation des fonctions se rapportant à une tournée.

5.17.2 Documentation des fonctions

5.17.2.1 void tour_2opt (Tour * *pTour,* int *pFirst,* int *pSecond*)

Fait une 2opt.

Paramètres

pTour Le tour pourlaquelle on veut faire une 2opt

pFirst L'id du début du premier trajet

pSecond L'id du début du second trajet

5.17.2.2 void tour_calculLength (Tour * *pTour*)

Calcul la longueur d'une tournée.

Paramètres

pTour La tournée pourlaquelle on veut calculer la longueur

5.17.2.3 void tour_display (const Tour *pTour*)

Affiche une tournée.

Paramètres

pTour La tournée à afficher

5.17.2.4 Tour tour_new (Instance *pInstance*)

Créer une nouvelle tournée initialisée avec les données d'une instance.

Paramètres

pInstance Instance servant à initialisée la tournée

Renvoie

la nouvelle tournée

5.17.2.5 bool tour_nextPermutation (Tour * *pPermutation*)

Génère la permutation de ville suivante d'une tournée.

Paramètres

pPermutation La tournée pour laquelle la permutation doit être générée

Renvoie

Vrai si une permutation à été généré faux s'il ne reste plus de permutation.

5.17.2.6 Tour tour_randomWalk (const Instance *pInstance*)

Génère une tournée aléatoire.

Paramètres

pInstance L'instance pour laquelle générer un random walk

Renvoie

La tournée aléatoire

5.18 Référence du fichier src/town.c

Fonctions des villes.

```
#include "town.h"
```

Fonctions

– Town town_new (const int pId, const int pX, const int pY)
Création d'une nouvelle ville.

5.18.1 Description détaillée

Fonctions des villes.
Auteur

Antoine de Roquemaurel

Date

21/11/2012 22 :35 :14

implémentation des fonctions se rapportant à une ville.

5.18.2 Documentation des fonctions

5.18.2.1 Town town_new (const int *pId,* const int *pX,* const int *pY*)

Création d'une nouvelle ville.

Paramètres

pId Id de la ville

pX Abscisse

pY Ordonnée

5.19 Référence du fichier src/util.c

Fonctions utiles.

```
#include "tour.h"
#include "util.h"
```

Fonctions

– int util_searchFirstOccurenceInArray (char **pArray, const int pSize, char *pSearch)
Cherche la première occurence d'une chaine de caractère dans un tableau de chaine caractère.

– void util_reverseArray (Town *pTab, const int pBegin, const int pEnd)
Inverse les éléments d'un tableau pTag entre les cases pBegin et pEnd.

– void util_displayArray (const int *pTab, const int pSize)
Affiche le contenu d'un tableau d'entiers.

– int util_sum (const int pBegin, const int pEnd)
Calcul la somme des éléments allant de pBegin à pEnd.

– void util_swap (int *a, int *b)
Échange deux variables.

– int util_rand (const int pMin, const int pMax)
Calcul une valeur aléatoire entre pMin et pMax.

– bool util_sousTabExist (Tour pChild, const int pBegin, const int pEnd, Tour pParent, bool pRecursvite)
Indique si un sous-tableau d'un tableau est présent dans un autre tableau.

5.19.1 Description détaillée

Fonctions utiles.
Auteur

Antoine de Roquemaurel

Date

19/11/2012 16 :27 :39

Entêtes des fonctions pouvant être utiles dans tout le projet. Ce sont des fonctions simples, qui doivent être indépendantes du projet.

5.19.2 Documentation des fonctions

5.19.2.1 void util_displayArray (const int * *pTab*, const int *pSize*)

Affiche le contenu d'un tableau d'entiers.

Paramètres

pTab Le tableau à afficher
pSize La taille du tableau

5.19.2.2 int util_rand (const int *pMin*, const int *pMax*)

Calcul une valeur aléatoire entre pMin et pMax.

Paramètres

pMin Minimum

pMax Maximum

Renvoie

5.19.2.3 void util_reverseArray (Town * *pTab,* const int *pBegin,* const int *pEnd*)

Inverse les éléments d'un tableau pTag entre les cases pBegin et pEnd.

Paramètres

pTab Tableau à inverser

pBegin Début de la section à inverser

pEnd Fin de la section à inverser

5.19.2.4 int util_searchFirstOccurenceInArray (char ** *pArray,* const int *pSize,* char * *pSearch*)

Cherche la première occurence d'une chaine de caractère dans un tableau de chaine caractère.

Paramètres

pArray Le tableau de chaines de caractères dans lequel chercher

pSize La taille du tableau

pSearch La chaine de caractère à chercher

Renvoie

La position de la chaine dans le tableau ou -1 si elle n'a pas été trouvée

5.19.2.5 bool util_sousTabExist (Tour *pChild,* const int *pBegin,* const int *pEnd,* Tour *pParent,* bool *pRecursvite*)

Indique si un sous-tableau d'un tableau est présent dans un autre tableau.

Paramètres

pChild Le tableau à chercher

pBegin Le début de la section à chercher

pEnd La fin de la section à chercher

pParent Le tableau dans lequel chercher

pRecursvite Pour simplifier vis-à-vis de la récursivité, doit toujours être à false

Renvoie

Vrai si le sous-tableau à été trouvé faux sinon

5.19.2.6 int util_sum (const int *pBegin*, const int *pEnd*)

Calcul la somme des éléments allant de pBegin à pEnd.

Paramètres

pBegin Début de la somme

pEnd Fin de la somme

Renvoie

La somme des éléments

5.19.2.7 void util_swap (int * *a*, int * *b*)

Échange deux variables.

Paramètres

a Première variable à échanger

b Seconde variable

Index