

Le voyageur de commerce

1

Généré par Doxygen 1.7.1

Sun Jan 13 2013 16 :47 :43

Table des matières

1	Le problème du voyageur de commerce - Projet d'algorithmique en langage C	1
1.1	problème	1
1.2	algorithmes implémenté	1
2	Index des structures de données	3
2.1	Structures de données	3
3	Index des fichiers	5
3.1	Liste des fichiers	5
4	Documentation des structures de données	7
4.1	Référence de la structure Algo	7
4.1.1	Description détaillée	7
4.2	Référence de la structure Distance	7
4.2.1	Description détaillée	8
4.3	Référence de la structure Errors	8
4.3.1	Description détaillée	9
4.4	Référence de la structure Instance	9
4.4.1	Description détaillée	9
4.5	Référence de la structure Path	10
4.5.1	Description détaillée	10
4.6	Référence de la structure Tour	10
4.6.1	Description détaillée	11
4.7	Référence de la structure Town	11
4.7.1	Description détaillée	11
5	Documentation des fichiers	13
5.1	Référence du fichier lib/bruteForce.h	13
5.1.1	Description détaillée	13
5.1.2	Documentation des fonctions	13

5.1.2.1	<code>bruteForce_bestPath</code>	13
5.2	Référence du fichier <code>lib/distance.h</code>	14
5.2.1	Description détaillée	14
5.2.2	Documentation des fonctions	14
5.2.2.1	<code>distance_betweenTowns</code>	14
5.2.2.2	<code>distance_calculDistance</code>	15
5.2.2.3	<code>distance_new</code>	15
5.3	Référence du fichier <code>lib/errors.h</code>	15
5.3.1	Description détaillée	16
5.3.2	Documentation des fonctions	16
5.3.2.1	<code>errors_displayErrorsMessage</code>	16
5.3.2.2	<code>errors_new</code>	17
5.3.2.3	<code>errors_setFileNotFound</code>	17
5.3.2.4	<code>errors_setMissingParameterGa</code>	17
5.3.2.5	<code>errors_setMissingParameterLsnr</code>	17
5.3.2.6	<code>errors_setMissingParameterLsr</code>	17
5.3.2.7	<code>errors_setNbArguments</code>	17
5.3.2.8	<code>errors_setNoAlgoSpecified</code>	18
5.3.2.9	<code>errors_setNoValidParameterGa</code>	18
5.3.2.10	<code>errors_setNoValidParameterLsnr</code>	18
5.3.2.11	<code>errors_setNoValidParameterLsr</code>	18
5.3.2.12	<code>errors_setTagFNotFound</code>	18
5.4	Référence du fichier <code>lib/genetic.h</code>	18
5.4.1	Description détaillée	19
5.4.2	Documentation des fonctions	19
5.4.2.1	<code>genetic_DPX</code>	19
5.4.2.2	<code>genetic_getBestPath</code>	19
5.4.2.3	<code>genetic_mutation</code>	20
5.5	Référence du fichier <code>lib/instance.h</code>	20
5.5.1	Description détaillée	21
5.5.2	Documentation des fonctions	21
5.5.2.1	<code>instance_display</code>	21
5.5.2.2	<code>instance_displayLinearVector</code>	21
5.5.2.3	<code>instance_displayMatrix</code>	21
5.5.2.4	<code>instance_initializeDistancesMatrix</code>	22
5.5.2.5	<code>instance_new</code>	22

5.5.2.6	<code>instance_push</code>	22
5.6	Référence du fichier <code>lib/localSearch.h</code>	22
5.6.1	Description détaillée	23
5.6.2	Documentation des fonctions	23
5.6.2.1	<code>localSearch_randomBestPath</code>	23
5.6.2.2	<code>localSearch_systematicBestPath</code>	23
5.7	Référence du fichier <code>lib/parsing.h</code>	24
5.7.1	Description détaillée	24
5.7.2	Documentation du type de l'énumération	24
5.7.2.1	<code>AlgoType</code>	24
5.7.3	Documentation des fonctions	25
5.7.3.1	<code>parsing_algoType</code>	25
5.7.3.2	<code>parsing_parseFileName</code>	25
5.7.3.3	<code>parsing_parseVerboseMode</code>	25
5.8	Référence du fichier <code>lib/path.h</code>	26
5.8.1	Description détaillée	26
5.8.2	Documentation des fonctions	26
5.8.2.1	<code>path_addNearNeighbor</code>	26
5.8.2.2	<code>path_display</code>	27
5.8.2.3	<code>path_new</code>	27
5.9	Référence du fichier <code>lib/tour.h</code>	27
5.9.1	Description détaillée	28
5.9.2	Documentation des fonctions	28
5.9.2.1	<code>tour_2opt</code>	28
5.9.2.2	<code>tour_addSeveralTowns</code>	29
5.9.2.3	<code>tour_addTown</code>	29
5.9.2.4	<code>tour_calculLength</code>	29
5.9.2.5	<code>tour_display</code>	29
5.9.2.6	<code>tour_new</code>	29
5.9.2.7	<code>tour_nextPermutation</code>	30
5.9.2.8	<code>tour_randomWalk</code>	30
5.9.2.9	<code>tour_replaceTheWorstTour</code>	30
5.10	Référence du fichier <code>lib/town.h</code>	30
5.10.1	Description détaillée	31
5.10.2	Documentation des fonctions	31
5.10.2.1	<code>town_display</code>	31

5.10.2.2	town_new	31
5.11	Référence du fichier lib/util.h	31
5.11.1	Description détaillée	32
5.11.2	Documentation des fonctions	32
5.11.2.1	util_deleteArrayValue	32
5.11.2.2	util_displayArray	33
5.11.2.3	util_rand	33
5.11.2.4	util_reverseArray	33
5.11.2.5	util_searchFirstOccurenceInArray	33
5.11.2.6	util_sousTabExist	34
5.11.2.7	util_sum	34
5.11.2.8	util_swap	34
5.12	Référence du fichier src/bruteForce.c	34
5.12.1	Description détaillée	35
5.12.2	Documentation des fonctions	35
5.12.2.1	bruteForce_bestPath	35
5.13	Référence du fichier src/distance.c	35
5.13.1	Description détaillée	35
5.13.2	Documentation des fonctions	36
5.13.2.1	distance_betweenTowns	36
5.13.2.2	distance_calculDistance	36
5.13.2.3	distance_new	36
5.14	Référence du fichier src/errors.c	37
5.14.1	Description détaillée	37
5.14.2	Documentation des fonctions	38
5.14.2.1	errors_displayErrorsMessage	38
5.14.2.2	errors_new	38
5.14.2.3	errors_setFileNotFound	38
5.14.2.4	errors_setMissingParameterGa	38
5.14.2.5	errors_setMissingParameterLsnr	38
5.14.2.6	errors_setMissingParameterLsr	39
5.14.2.7	errors_setNoAlgoSpecified	39
5.14.2.8	errors_setNoValidParameterGa	39
5.14.2.9	errors_setNoValidParameterLsnr	39
5.14.2.10	errors_setNoValidParameterLsr	39
5.14.2.11	errors_setTagFNotFound	39

5.15	Référence du fichier src/genetic.c	39
5.15.1	Description détaillée	40
5.15.2	Documentation des fonctions	40
5.15.2.1	genetic_DPX	40
5.15.2.2	genetic_getBestPath	40
5.15.2.3	genetic_mutation	41
5.16	Référence du fichier src/instance.c	41
5.16.1	Description détaillée	41
5.16.2	Documentation des fonctions	42
5.16.2.1	instance_display	42
5.16.2.2	instance_displayLinearVector	42
5.16.2.3	instance_displayMatrix	42
5.16.2.4	instance_initializeDistancesMatrix	42
5.16.2.5	instance_new	43
5.16.2.6	instance_push	43
5.17	Référence du fichier src/localSearch.c	43
5.17.1	Description détaillée	43
5.17.2	Documentation des fonctions	44
5.17.2.1	localSearch_randomBestPath	44
5.17.2.2	localSearch_systematicBestPath	44
5.18	Référence du fichier src/main.c	44
5.18.1	Description détaillée	45
5.18.2	Documentation des fonctions	45
5.18.2.1	main	45
5.19	Référence du fichier src/parsing.c	45
5.19.1	Description détaillée	45
5.19.2	Documentation des fonctions	45
5.19.2.1	parsing_algoType	45
5.19.2.2	parsing_parseFileName	46
5.19.2.3	parsing_parseVerboseMode	46
5.20	Référence du fichier src/path.c	46
5.20.1	Description détaillée	47
5.20.2	Documentation des fonctions	47
5.20.2.1	path_addNearNeighbor	47
5.20.2.2	path_display	47
5.20.2.3	path_new	47

5.21	Référence du fichier src/tour.c	48
5.21.1	Description détaillée	48
5.21.2	Documentation des fonctions	49
5.21.2.1	tour_2opt	49
5.21.2.2	tour_addSeveralTowns	49
5.21.2.3	tour_addTown	49
5.21.2.4	tour_calculLength	49
5.21.2.5	tour_display	50
5.21.2.6	tour_new	50
5.21.2.7	tour_nextPermutation	50
5.21.2.8	tour_randomWalk	50
5.21.2.9	tour_replaceTheWorstTour	51
5.22	Référence du fichier src/town.c	51
5.22.1	Description détaillée	51
5.22.2	Documentation des fonctions	51
5.22.2.1	town_display	51
5.22.2.2	town_new	51
5.23	Référence du fichier src/util.c	52
5.23.1	Description détaillée	52
5.23.2	Documentation des fonctions	53
5.23.2.1	util_deleteArrayValue	53
5.23.2.2	util_displayArray	53
5.23.2.3	util_rand	53
5.23.2.4	util_reverseArray	53
5.23.2.5	util_searchFirstOccurenceInArray	53
5.23.2.6	util_sousTabExist	54
5.23.2.7	util_sum	54
5.23.2.8	util_swap	54

Chapitre 1

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

Auteur

L2 Antoine de Roquemaurel (G1.1)

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.

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 à l'optimisation de trajectoires de machines-outils : par exemple, pour minimiser le temps total que met une fraiseuse à commande numérique pour percer n points dans une plaque de tôle ou pour percer les trous des composants d'un circuit électronique comme dans le cas qui nous intéresse.

Ce problème est plus compliqué qu'il n'y paraît et on ne connaît 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 approchées, 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 longtemps et on dispose d'une grande variété d'algorithmes donnant le plus souvent des solutions approchées mais calculables en un temps raisonnable.

1.2 algorithmes implémentée

Ce problème sera implémenté via différents algorithmes :

- Brute force

- Recherche locale aléatoire
- Recherche locale systématique
- Algorithme génétique

Chapitre 2

Index des structures de données

2.1 Structures de données

Liste des structures de données avec une brève description :

Algo (Enumération d'un algorithme)	7
Distance (Objet des distances)	7
Errors (Objet des erreurs)	8
Instance (Objet des instances)	9
Path (Objet d'un trajet)	10
Tour (Objet d'une tournée)	10
Town (Objet des ville)	11

Chapitre 3

Index des fichiers

3.1 Liste des fichiers

Liste de tous les fichiers documentés avec une brève description :

lib/ bruteForce.h (Fonctions de brute force)	13
lib/ distance.h (Fonctions des distances)	14
lib/ errors.h (Fonctions sur les erreurs)	15
lib/ genetic.h (Fonctions d'algorithmes génétiques)	18
lib/ instance.h (Fonctions sur les instances)	20
lib/ localSearch.h (Fonctions de recherche locale)	22
lib/ parsing.h (Fonctions de parsing des arguments)	24
lib/ path.h (Fonctions des trajets)	26
lib/ tour.h (Fonctions des tournées)	27
lib/ town.h (Fonctions des villes)	30
lib/ util.h (Fonctions utiles)	31
src/ bruteForce.c (Fonctions de brute force)	34
src/ distance.c (Fonctions des distances)	35
src/ errors.c (Fonctions sur les erreurs)	37
src/ genetic.c (Fonctions d'algorithmes génétiques)	39
src/ instance.c (Fonctions utiles)	41
src/ localSearch.c (Fonctions de recherche locale)	43
src/ main.c	44
src/ parsing.c (Fonctions de parsing des arguments)	45
src/ path.c (Fonctions des trajets)	46
src/ tour.c (Fonctions des tournées)	48
src/ town.c (Fonctions des villes)	51
src/ util.c (Fonctions utiles)	52

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.
- double [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

Voir également

[AlgoType](#)

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 * [errorTagNotFound](#)
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 chaînes de caractères des erreurs. Si une variable vaut NULL, l'erreur n'est pas présente, sinon elle sera affichée. 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](#)
Tableau linéaire contenant toutes les distances entre les villes.
- int [nbTowns](#)
Nombre de ville de l' [Instance](#).
- char * [name](#)
Nom 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 Path

Objet d'un trajet.

```
#include <path.h>
```

Champs de données

- [Town towns](#) [N]
Tableau de ville. Les villes sont triés dans l'ordre du trajet.
- int [nbTowns](#)
Nombre de ville du trajet.

4.5.1 Description détaillée

Objet d'un trajet. Informations concernant un trajet.

Voir également

[Town](#)

La documentation de cette structure a été générée à partir du fichier suivant :

- lib/[path.h](#)

4.6 Référence de la structure Tour

Objet d'une tournée.

```
#include <tour.h>
```

Champs de données

- [Town towns](#) [256]
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.

4.6.1 Description détaillée

Objet d'une tournée.

Voir également

[Town](#), [Distance](#)

Informations concernant une tournée.

La documentation de cette structure a été générée à partir du fichier suivant :

– [lib/tour.h](#)

4.7 Référence de la structure Town

Objet des ville.

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

Champs de données

- float [x](#)
Abscisse de la ville.
- float [y](#)
Ordonnée de la ville.
- int [id](#)
Id de la ville.

4.7.1 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.

Voir également

[Instance](#)

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.
- [Distance](#) [distance_betweenTowns](#) ([Distance](#) *pDistances, [Town](#) i, [Town](#) j)
Calcul la distance entre deux villes.

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 [Distance](#) [distance_betweenTowns](#) ([Distance](#) * *pDistances*, [Town](#) *i*, [Town](#) *j*)

Calcul la distance entre deux villes.

Paramètres

pDistances La matrice de distances

i La première ville

j La seconde ville

Renvoie

La distance

Voir également

[Town](#)

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

Voir également

[Town](#)

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

Voir également

[Town](#)

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_setNoValidParameterGa (Errors *pErrors)`
Signale que le ou les paramètre après -ga ne sont pas valides (non entier ou décimal).
- `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 laquelle 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_setNoValidParameterGa (Errors * *pErrors*)

Signale que le ou les paramètre après -ga ne sont pas valides (non entier ou décimal).

Paramètres

pErrors L'objet des erreurs

5.3.2.10 void errors_setNoValidParameterLsnr (Errors * *pErrors*)

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

Paramètres

pErrors L'objet des erreurs

5.3.2.11 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

5.3.2.12 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/genetic.h

Fonctions d'algorithmes génétiques.

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

```
#include "path.h"
```

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

Fonctions

- [Tour genetic_DPX](#) ([Tour](#) pParent1, [Tour](#) pParent2)
Retourne la tournée après un croisement DPX entre pParent1 et pParent2.
- bool [genetic_mutation](#) ([Tour](#) *pTour, float pProba)
Fais muter la tournée pTour.
- [Tour genetic_getBestPath](#) ([Instance](#) pInstance, const int pNbTour, const int pNbGeneration, const float pProba)
Retourne la meilleure tournée à l'aide d'un algorithme génétique.

5.4.1 Description détaillée

Fonctions d'algorithmes génétiques.

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

5.4.2 Documentation des fonctions

5.4.2.1 [Tour genetic_DPX](#) ([Tour](#) pParent1, [Tour](#) pParent2)

Retourne la tournée après un croisement DPX entre pParent1 et pParent2.

Paramètres

pParent1 La première tournée parent

pParent2 La seconde tournée parent

Renvoie

La fille créée à l'aide d'un croisement DPX

Voir également

[Tour](#)

5.4.2.2 [Tour genetic_getBestPath](#) ([Instance](#) pInstance, const int pNbTour, const int pNbGeneration, const float pProba)

Retourne la meilleure tournée à l'aide d'un algorithme génétique.

Paramètres

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

pNbTour Le nombre d'individus, ou tournée, que l'on veut initialiser

pNbGeneration Le nombre de génération que l'on veut faire avant de s'arrêter

pProba La probabilité que la fille mute

Renvoie

La meilleure tournée

Voir également

[Instance](#)

5.4.2.3 bool genetic_mutation (Tour * pTour, float pProba)

Fais muter la tournée pTour.

Paramètres

pTour La tournée à faire muter

pProba La probabilité que la tournée pTour mute

Renvoie

Vrai si la tournée à muter faux sinon.

5.5 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](#) (const [Instance](#) pInstance)
Affiche la matrice des distances sous forme linéaire.
- void [instance_displayMatrix](#) (const [Instance](#) pInstance)
Affiche la matrice des distances sous forme de matrice symétrique.

5.5.1 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.

5.5.2 Documentation des fonctions

5.5.2.1 void [instance_display](#) (const [Instance](#) *pInstance*)

Affiche une [Instance](#).

Paramètres

pInstance l' [Instance](#) à afficher

5.5.2.2 void [instance_displayLinearVector](#) (const [Instance](#) *pInstance*)

Affiche la matrice des distances sous forme linéaire.

Paramètres

pInstance L'instance à afficher

Voir également

[Distance](#)

5.5.2.3 void [instance_displayMatrix](#) (const [Instance](#) *pInstance*)

Affiche la matrice des distances sous forme de matrice symétrique.

Paramètres

pInstance L'instance à afficher

Voir également

[Distance](#)

5.5.2.4 void instance_initializeDistancesMatrix (Instance * *pInstance*)

Initialise la matrice des distances.

Paramètres

pInstance L'instance à modifier

Voir également

[Distance](#)

5.5.2.5 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](#)

5.5.2.6 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

Voir également

[Town](#)

5.6 Référence du fichier lib/localSearch.h

Fonctions de recherche locale.

```
#include <time.h>
```

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

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

Fonctions

- [Tour localSearch_randomBestPath](#) ([Instance](#) pInstance, int pTryNb)
Effectue une recherche locale aléatoire.
- [Tour localSearch_systematicBestPath](#) ([Instance](#) pInstance, int pTryNb)
Effectue une recherche locale systématique.

5.6.1 Description détaillée

Fonctions de recherche locale.

Auteur

Antoine de Roquemaurel

Date

27/12/2012 18:00:18

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

5.6.2 Documentation des fonctions

5.6.2.1 Tour localSearch_randomBestPath (Instance pInstance, int pTryNb)

Effectue une recherche locale aléatoire.

Paramètres

pInstance L'instance pour laquelle on veut rechercher le plus court chemin

pTryNb Le nombre d'essais à effectuer

Renvoie

La meilleure tournée

Voir également

[Instance](#)

5.6.2.2 Tour localSearch_systematicBestPath (Instance pInstance, int pTryNb)

Effectue une recherche locale systématique.

Paramètres

pInstance L'instance pour laquelle on veut rechercher le plus court chemin

pTryNb Le nombre d'essais à effectuer

Renvoie

La meilleure tournée

Voir également

[Instance](#)

5.7 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)

5.7.1 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.

5.7.2 Documentation du type de l'énumération

5.7.2.1 enum AlgoType

[parsing.h](#)

Enumération des types d'algorithmes Les différents types d'algorithmes qui peuvent être appelés.

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.

5.7.3 Documentation des fonctions

5.7.3.1 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 où l'utilisateur entre plusieurs algorithmes. La fin du tableau est marqué par END

Voir également

[Algo](#), [Errors](#)

5.7.3.2 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

[Errors](#), [Instance](#)

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

Cherche si le mode verbeux a été spécifié ou non.

Paramètres

pTab Le tableau contenant les arguments
pSize Le nombre des arguments

Renvoie

Vrai si mode verbeux, Faux sinon

5.8 Référence du fichier lib/path.h

Fonctions des trajets.

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

Structures de données

- struct [Path](#)
Objet d'un trajet.

Macros

- #define [N](#) 1024
Taille maximale du tableau de ville.

Fonctions

- [Path](#) [path_new](#) ([Tour](#) pTour, int pBegin, int pEnd)
Créer un nouveau trajet.
- void [path_addNearNeighbor](#) ([Tour](#) *pTour, [Path](#) *pPathsList, int *pNbPathsList)
Ajoute le trajet le plus proche à la suite de pTour.
- void [path_display](#) (const [Path](#) pPath)
Afficher un trajet.

5.8.1 Description détaillée

Fonctions des trajets.

Auteur

Antoine de Roquemaurel

Date

11/01/2013 03 :12 :00

Entêtes des fonctions se rapportant à un trajet.

5.8.2 Documentation des fonctions

5.8.2.1 void [path_addNearNeighbor](#) ([Tour](#) * *pTour*, [Path](#) * *pPathsList*, int * *pNbPathsList*)

Ajoute le trajet le plus proche à la suite de pTour.

Paramètres

- pTour* Le tour à compléter
- pPathsList* La liste des trajets possibles
- pNbPathsList* Le nombre de trajets possibles

Voir également

[Tour](#)

5.8.2.2 void path_display (const Path pPath)

Afficher un trajet.

Paramètres

- pPath* Le trajet à afficher

5.8.2.3 Path path_new (Tour pTour, int pBegin, int pEnd)

Créer un nouveau trajet.

Paramètres

- pTour* La tournée dans laquelle on doit prendre le trajet
- pBegin* Le début du trajet
- pEnd* La fin du trajet

Renvoie

Le nouveau trajet

Voir également

[Tour](#)

5.9 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)`
Cr  er une nouvelle tourn  e initialis  e avec les donn  es d'une instance.
- `bool tour_nextPermutation (Tour *pPermutation)`
G  n  re la permutation de ville suivante d'une tourn  e.
- `void tour_calculLength (Tour *pTour)`
Calcul la longueur d'une tourn  e.
- `void tour_display (const Tour pTour)`
Affiche une tourn  e.
- `Tour tour_randomWalk (const Instance pInstance)`
G  n  re une tourn  e al  atoire.
- `void tour_2opt (Tour *pTour, int pFirst, int pSecond)`
Fait une 2opt.
- `void tour_addTown (Tour *pTour, Town pTown)`
Ajoute une ville    la fin de la tourn  e pTour.
- `void tour_addSeveralTowns (Tour *pTour, Town *pTowns, const int pNbTowns)`
Ajoute un tableau de ville    la fin de la tourn  e pTour.
- `void tour_replaceTheWorstTour (Tour *pTours, const int pSize, Tour pTour)`
Remplace la pire tourn  e d'un tableau de tourn  e par la tourn  e pTour.

5.9.1 Description d  taill  e

Fonctions des tourn  es.

Auteur

Antoine de Roquemaurel

Date

21/11/2012 22 :04 :13

Ent  tes des fonctions se rapportant    une tourn  e. *

5.9.2 Documentation des fonctions

5.9.2.1 `void tour_2opt (Tour * pTour, int pFirst, int pSecond)`

Fait une 2opt.

Param  tres

pTour Le tour pour laquelle on veut faire une 2opt

pFirst L'id du d  but du premier trajet

pSecond L'id du d  but du second trajet

5.9.2.2 void tour_addSeveralTowns (Tour * *pTour*, Town * *pTowns*, const int *pNbTowns*)

Ajoute un tableau de ville à la fin de la tournée *pTour*.

Paramètres

pTour La tournée à compléter
pTowns Le tableau de villes à ajouter
pNbTowns Le nombre de villes à ajouter

Voir également

[Town](#)

5.9.2.3 void tour_addTown (Tour * *pTour*, Town *pTown*)

Ajoute une ville à la fin de la tournée *pTour*.

Paramètres

pTour La tournée à compléter
pTown La ville à ajouter

Voir également

[Town](#)

5.9.2.4 void tour_calculLength (Tour * *pTour*)

Calcul la longueur d'une tournée.

Paramètres

pTour La tournée pour laquelle on veut calculer la longueur

5.9.2.5 void tour_display (const Tour *pTour*)

Affiche une tournée.

Paramètres

pTour La tournée à afficher

5.9.2.6 Tour tour_new (Instance *pInstance*)

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

Paramètres

pInstance [Instance](#) servant à initialiser la tournée

Renvoie

la nouvelle tournée

Voir également

[Instance](#)

5.9.2.7 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.

5.9.2.8 Tour tour_randomWalk (const Instance *pInstance*)

Génère une tournée aléatoire.

Paramètres

pInstance L'instance pour laquelle générer un random walk

Renvoie

La tournée aléatoire

Voir également

[Instance](#)

5.9.2.9 void tour_replaceTheWorstTour (Tour * *pTours*, const int *pSize*, Tour *pTour*)

Remplace la pire tournée d'un tableau de tournée par la tournée pTour.

Paramètres

pTours Le tableau de tournées

pSize La taille du tableau de tournées

pTour Le tour à ajouter

5.10 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 float pX, const float pY)
Création d'une nouvelle ville.
- void `town_display` (const `Town` pTown)
Affiche une ville.

5.10.1 Description détaillée

Fonctions des villes.

Auteur

Antoine de Roquemaurel

Date

21/11/2012 22 :35 :19

Entêtes des fonctions se rapportant à une ville.

5.10.2 Documentation des fonctions

5.10.2.1 void town_display (const `Town` pTown)

Affiche une ville.

Paramètres

pTown La ville à afficher

5.10.2.2 `Town` town_new (const int pId, const float pX, const float pY)

Création d'une nouvelle ville.

Paramètres

pId Id de la ville

pX Abscisse

pY Ordonnée

5.11 Référence du fichier lib/util.h

Fonctions utiles.

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

Fonctions

- int `util_searchFirstOccurenceInArray` (char **pArray, const int pSize, char *pSearch)
Cherche la première occurrence d'une chaîne de caractère dans un tableau de chaîne caractère.
- void `util_reverseArray` (Town *pTab, const int pBegin, const int pEnd)
Inverse les éléments d'un tableau pTab 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` (Town *a, Town *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.
- void `util_deleteArrayValue` (Path *pArray, int pSize, int pDeleteIndice)
Supprime la valeur pDeleteIndice d'un tableau.

Variables

- bool `gVerboseMode`
Mode verbose.

5.11.1 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.

5.11.2 Documentation des fonctions

5.11.2.1 void `util_deleteArrayValue` (Path * pArray, int pSize, int pDeleteIndice)

Supprime la valeur pDeleteIndice d'un tableau.

Paramètres

pArray Le tableau

pSize La taille du tableau

pDeleteIndice L'indice à supprimer

5.11.2.2 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

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

Calcul une valeur aléatoire entre pMin et pMax.

Paramètres

pMin Minimum

pMax Maximum

Renvoie

la valeur aléatoire

5.11.2.4 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

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

Cherche la première occurrence d'une chaîne de caractère dans un tableau de chaîne caractère.

Paramètres

pArray Le tableau de chaînes de caractères dans lequel chercher

pSize La taille du tableau

pSearch La chaîne de caractère à chercher

Renvoie

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

5.11.2.6 `bool util_sousTabExist (Tour pChild, const int pBegin, const int pEnd, Tour pParent, bool pRecursive)`

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

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

Renvoie

Vrai si le sous-tableau a été trouvé faux sinon

5.11.2.7 `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

5.11.2.8 `void util_swap (Town * a, Town * b)`

Échange deux variables.

Paramètres

a Première variable à échanger

b Seconde variable

5.12 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.

5.12.1 Description détaillée

Fonctions de brute force.

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

5.12.2 Documentation des fonctions

5.12.2.1 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.

Voir également

[Instance](#)

5.13 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_betweenTowns](#) ([Distance](#) *pDistances, [Town](#) i, [Town](#) j)
Calcul la distance entre deux villes.

5.13.1 Description détaillée

Fonctions des distances.

Auteur

Antoine de Roquemaurel

Date

01/12/2012 20 :33 :39

Entêtes des fonctions se rapportant aux distances

5.13.2 Documentation des fonctions**5.13.2.1 Distance `distance_betweenTowns` (`Distance` * `pDistances`, `Town i`, `Town j`)**

Calcul la distance entre deux villes.

Paramètres

pDistances La matrice de distances

i La première ville

j La seconde ville

Renvoie

La distance

Voir également

[Town](#)

5.13.2.2 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

Voir également

[Town](#)

5.13.2.3 Distance `distance_new` (`Town pFirstTown`, `Town pSecondTown`)

Créer une nouvelle distance.

Paramètres

pFirstTown Première ville

pSecondTown Seconde ville

Renvoie

[Distance](#) entre les deux villes

Voir également

[Town](#)

5.14 Référence du fichier src/errors.c

Fonctions sur les erreurs.

```
#include "errors.h"
```

Fonctions

- [Errors errors_new](#) ()
Initialisation de l'objet.
- void [errors_displayErrorMessage](#) (const [Errors](#) pErrors)
Affiches toutes les erreurs de pErrors.
- 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_setMissingParameterLsr](#) ([Errors](#) *pErrors)
Signale qu'aucun paramètre n'a été spécifié derrière -lsr.
- void [errors_setMissingParameterLsnr](#) ([Errors](#) *pErrors)
Signale qu'aucun paramètre n'a été spécifié derrière -lsnr.
- void [errors_setMissingParameterGa](#) ([Errors](#) *pErrors)
Signale qu'aucun paramètre n'a été spécifié derrière -ga.
- void [errors_setNoValidParameterLsr](#) ([Errors](#) *pErrors)
Signale que le paramètre derrière -lsr n'est pas valide (n'est pas un entier).
- void [errors_setNoValidParameterLsnr](#) ([Errors](#) *pErrors)
Signale que les paramètres derrière -lsnr ne sont pas valides (non entier).
- void [errors_setNoValidParameterGa](#) ([Errors](#) *pErrors)
Signale que le ou les paramètre après -ga ne sont pas valides (non entier ou décimal).

5.14.1 Description détaillée

Fonctions sur les erreurs.

Auteur

Antoine de Roquemaurel

Date

21/11/2012 17 :42 :31

Implémentation des fonctions concernant les erreurs du programme.

5.14.2 Documentation des fonctions**5.14.2.1 void errors_displayErrorsMessage (const Errors *pErrors*)**

Affiches toutes les erreurs de *pErrors*.

Paramètres

pErrors L'objet pour laquelle on doit afficher les erreurs

5.14.2.2 Errors errors_new ()

Initialisation de l'objet.

Renvoie

Une instance de Erreur initialisée

5.14.2.3 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

5.14.2.4 void errors_setMissingParameterGa (Errors * *pErrors*)

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

Paramètres

pErrors L'objet des erreurs

5.14.2.5 void errors_setMissingParameterLsnr (Errors * *pErrors*)

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

Paramètres

pErrors L'objet des erreurs

5.14.2.6 void errors_setMissingParameterLsr (Errors * *pErrors*)

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

Paramètres

pErrors L'objet des erreurs

5.14.2.7 void errors_setNoAlgoSpecified (Errors * *pErrors*)

Signale qu'aucun algorithme n'a été spécifié.

Paramètres

pErrors L'objet des erreurs

5.14.2.8 void errors_setNoValidParameterGa (Errors * *pErrors*)

Signale que le ou les paramètre après -ga ne sont pas valides (non entier ou décimal).

Paramètres

pErrors L'objet des erreurs

5.14.2.9 void errors_setNoValidParameterLsnr (Errors * *pErrors*)

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

Paramètres

pErrors L'objet des erreurs

5.14.2.10 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

5.14.2.11 void errors_setTagFNotFound (Errors * *pErrors*)

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

Paramètres

pErrors L'objet des erreurs

5.15 Référence du fichier src/genetic.c

Fonctions d'algorithmes génétiques.

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

Fonctions

- [Tour genetic_DPX](#) ([Tour](#) pParent1, [Tour](#) pParent2)
Retourne la tournée après un croisement DPX entre pParent1 et pParent2.
- bool [genetic_mutation](#) ([Tour](#) *pTour, float pProba)
Fais muter la tournée pTour.
- [Tour genetic_getBestPath](#) ([Instance](#) pInstance, const int pNbTour, const int pNbGeneration, const float pProba)
Retourne la meilleure tournée à l'aide d'un algorithme génétique.

5.15.1 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

5.15.2 Documentation des fonctions

5.15.2.1 [Tour genetic_DPX](#) ([Tour](#) pParent1, [Tour](#) pParent2)

Retourne la tournée après un croisement DPX entre pParent1 et pParent2.

Paramètres

pParent1 La première tournée parent

pParent2 La seconde tournée parent

Renvoie

La fille créée à l'aide d'un croisement DPX

Voir également

[Tour](#)

5.15.2.2 [Tour genetic_getBestPath](#) ([Instance](#) pInstance, const int pNbTour, const int pNbGeneration, const float pProba)

Retourne la meilleure tournée à l'aide d'un algorithme génétique.

Paramètres

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

pNbTour Le nombre d'individus, ou tournée, que l'on veut initialiser

pNbGeneration Le nombre de génération que l'on veut faire avant de s'arrêter

pProba La probabilité que la fille mute

Renvoie

La meilleure tournée

Voir également

[Instance](#)

5.15.2.3 bool genetic_mutation (Tour * pTour, float pProba)

Fais muter la tournée pTour.

Paramètres

pTour La tournée à faire muter

pProba La probabilité que la tournée pTour mute

Renvoie

Vrai si la tournée à muter faux sinon.

5.16 Référence du fichier src/instance.c

Fonctions utiles.

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

```
#include "util.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](#) (const [Instance](#) pInstance)
Affiche la matrice des distances sous forme linéaire.
- void [instance_displayMatrix](#) (const [Instance](#) pInstance)
Affiche la matrice des distances sous forme de matrice symétrique.

5.16.1 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.

5.16.2 Documentation des fonctions

5.16.2.1 void instance_display (const Instance *pInstance*)

Affiche une [Instance](#).

Paramètres

pInstance l' [Instance](#) à afficher

5.16.2.2 void instance_displayLinearVector (const Instance *pInstance*)

Affiche la matrice des distances sous forme linéaire.

Paramètres

pInstance L'instance à afficher

Voir également

[Distance](#)

5.16.2.3 void instance_displayMatrix (const Instance *pInstance*)

Affiche la matrice des distances sous forme de matrice symétrique.

Paramètres

pInstance L'instance à afficher

Voir également

[Distance](#)

5.16.2.4 void instance_initializeDistancesMatrix (Instance * *pInstance*)

Initialise la matrice des distances.

Paramètres

pInstance L'instance à modifier

Voir également

[Distance](#)

5.16.2.5 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](#)

5.16.2.6 `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

Voir également

[Town](#)

5.17 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*)
Effectue une recherche locale aléatoire.
- [Tour localSearch_systematicBestPath](#) ([Instance](#) *pInstance*, int *pTryNb*)
Effectue une recherche locale systématique.

5.17.1 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.

5.17.2 Documentation des fonctions

5.17.2.1 Tour `localSearch_randomBestPath` (Instance *pInstance*, int *pTryNb*)

Effectue une recherche locale aléatoire.

Paramètres

pInstance L'instance pour laquelle on veut rechercher le plus court chemin

pTryNb Le nombre d'essais à effectuer

Renvoie

La meilleure tournée

Voir également

[Instance](#)

5.17.2.2 Tour `localSearch_systematicBestPath` (Instance *pInstance*, int *pTryNb*)

Effectue une recherche locale systématique.

Paramètres

pInstance L'instance pour laquelle on veut rechercher le plus court chemin

pTryNb Le nombre d'essais à effectuer

Renvoie

La meilleure tournée

Voir également

[Instance](#)

5.18 Référence du fichier `src/main.c`

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

Fonctions

- int [main](#) (int argc, char **argv)
Fonction d'entrée du programme.

5.18.1 Description détaillée

5.18.2 Documentation des fonctions

5.18.2.1 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

5.19 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)

5.19.1 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.

5.19.2 Documentation des fonctions

5.19.2.1 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 où l'utilisateur entre plusieurs algorithmes. La fin du tableau est marqué par END

Voir également

[Algo](#), [Errors](#)

5.19.2.2 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

[Errors](#), [Instance](#)

5.19.2.3 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

5.20 Référence du fichier src/path.c

Fonctions des trajets.

```
#include "path.h"
```

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

Fonctions

- [Path path_new](#) ([Tour](#) pTour, int pBegin, int pEnd)
Créer un nouveau trajet.
- void [path_addNearNeighbor](#) ([Tour](#) *pTour, [Path](#) *pPathsList, int *pNbPathsList)

Ajoute le trajet le plus proche à la suite de pTour.

- void `path_display` (const `Path` pPath)
Afficher un trajet.

5.20.1 Description détaillée

Fonctions des trajets.

Auteur

Antoine de Roquemaurel

Date

11/01/2013 03 :12 :13

Implémentations des fonctions se rapportant à un trajet.

5.20.2 Documentation des fonctions

5.20.2.1 void `path_addNearNeighbor` (`Tour` * *pTour*, `Path` * *pPathsList*, `int` * *pNbPathsList*)

Ajoute le trajet le plus proche à la suite de pTour.

Paramètres

pTour Le tour à compléter

pPathsList La liste des trajets possibles

pNbPathsList Le nombre de trajets possibles

Voir également

[Tour](#)

5.20.2.2 void `path_display` (`const Path` *pPath*)

Afficher un trajet.

Paramètres

pPath Le trajet à afficher

5.20.2.3 `Path` `path_new` (`Tour` *pTour*, `int` *pBegin*, `int` *pEnd*)

Créer un nouveau trajet.

Paramètres

pTour La tournée dans laquelle on doit prendre le trajet

pBegin Le début du trajet

pEnd La fin du trajet

Renvoie

Le nouveau trajet

Voir également

[Tour](#)

5.21 Référence du fichier src/tour.c

Fonctions des tournées.

```
#include "tour.h"
#include "path.h"
#include "util.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.
- void [tour_addTown](#) ([Tour](#) *pTour, [Town](#) pTown)
Ajoute une ville à la fin de la tournée pTour.
- void [tour_addSeveralTowns](#) ([Tour](#) *pTour, [Town](#) *pTowns, const int pNbTowns)
Ajoute un tableau de ville à la fin de la tournée pTour.
- void [tour_replaceTheWorstTour](#) ([Tour](#) *pTours, const int pSize, [Tour](#) pTour)
Remplace la pire tournée d'un tableau de tournée par la tournée pTour.

5.21.1 Description détaillée

Fonctions des tournées.

Auteur

Antoine de Roquemaurel

Date

21/11/2012 22 :04 :08

Implémentation des fonctions se rapportant à une tournée.

5.21.2 Documentation des fonctions

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

Fait une 2opt.

Paramètres

pTour Le tour pour laquelle on veut faire une 2opt

pFirst L'id du début du premier trajet

pSecond L'id du début du second trajet

5.21.2.2 void tour_addSeveralTowns (Tour * *pTour*, Town * *pTowns*, const int *pNbTowns*)

Ajoute un tableau de ville à la fin de la tournée pTour.

Paramètres

pTour La tournée à compléter

pTowns Le tableau de villes à ajouter

pNbTowns Le nombre de villes à ajouter

Voir également

[Town](#)

5.21.2.3 void tour_addTown (Tour * *pTour*, Town *pTown*)

Ajoute une ville à la fin de la tournée pTour.

Paramètres

pTour La tournée à compléter

pTown La ville à ajouter

Voir également

[Town](#)

5.21.2.4 void tour_calculLength (Tour * *pTour*)

Calcul la longueur d'une tournée.

Paramètres

pTour La tournée pour laquelle on veut calculer la longueur

5.21.2.5 void tour_display (const Tour *pTour*)

Affiche une tournée.

Paramètres

pTour La tournée à afficher

5.21.2.6 Tour tour_new (Instance *pInstance*)

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

Paramètres

pInstance [Instance](#) servant à initialiser la tournée

Renvoie

la nouvelle tournée

Voir également

[Instance](#)

5.21.2.7 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 a été générée faux s'il ne reste plus de permutation.

5.21.2.8 Tour tour_randomWalk (const Instance *pInstance*)

Génère une tournée aléatoire.

Paramètres

pInstance L'instance pour laquelle générer un random walk

Renvoie

La tournée aléatoire

Voir également

[Instance](#)

5.21.2.9 void tour_replaceTheWorstTour (Tour * pTours, const int pSize, Tour pTour)

Remplace la pire tournée d'un tableau de tournée par la tournée pTour.

Paramètres

- pTours* Le tableau de tournées
- pSize* La taille du tableau de tournées
- pTour* Le tour à ajouter

5.22 Référence du fichier src/town.c

Fonctions des villes.

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

Fonctions

- [Town town_new](#) (const int pId, const float pX, const float pY)
Création d'une nouvelle ville.
- void [town_display](#) (const [Town](#) pTown)
Affiche une ville.

5.22.1 Description détaillée

Fonctions des villes.

Auteur

Antoine de Roquemaurel

Date

21/11/2012 22 :35 :14

implémentation des fonctions se rapportant à une ville.

5.22.2 Documentation des fonctions

5.22.2.1 void town_display (const Town pTown)

Affiche une ville.

Paramètres

- pTown* La ville à afficher

5.22.2.2 Town town_new (const int pId, const float pX, const float pY)

Création d'une nouvelle ville.

Paramètres

pId Id de la ville

pX Abscisse

pY Ordonnée

5.23 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 occurrence d'une chaîne de caractère dans un tableau de chaîne caractère.
- void [util_reverseArray](#) (Town *pTab, const int pBegin, const int pEnd)
Inverse les éléments d'un tableau pTab 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](#) (Town *a, Town *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.
- void [util_deleteArrayValue](#) (Path *pArray, int pSize, int pDeleteIndice)
Supprime la valeur pDeleteIndice d'un tableau.

5.23.1 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.

5.23.2 Documentation des fonctions

5.23.2.1 void util_deleteArrayValue (Path * *pArray*, int *pSize*, int *pDeleteIndice*)

Supprime la valeur *pDeleteIndice* d'un tableau.

Paramètres

pArray Le tableau
pSize La taille du tableau
pDeleteIndice L'indice à supprimer

5.23.2.2 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

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

Calcul une valeur aléatoire entre *pMin* et *pMax*.

Paramètres

pMin Minimum
pMax Maximum

Renvoie

la valeur aléatoire

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

Inverse les éléments d'un tableau *pTab* 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

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

Cherche la première occurrence d'une chaîne de caractère dans un tableau de chaînes de caractères.

Paramètres

pArray Le tableau de chaînes de caractères dans lequel chercher

pSize La taille du tableau

pSearch La chaîne de caractère à chercher

Renvoie

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

5.23.2.6 `bool util_sousTabExist (Tour pChild, const int pBegin, const int pEnd, Tour pParent, bool pRecurvite)`

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

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

Renvoie

Vrai si le sous-tableau a été trouvé faux sinon

5.23.2.7 `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

5.23.2.8 `void util_swap (Town * a, Town * b)`

Échange deux variables.

Paramètres

a Première variable à échanger

b Seconde variable

Index

- Algo, [7](#)
- AlgoType
 - [parsing.h](#), [24](#)
- BRUTEFORCE
 - [parsing.h](#), [25](#)
- [bruteForce.c](#)
 - [bruteForce_bestPath](#), [35](#)
- [bruteForce.h](#)
 - [bruteForce_bestPath](#), [13](#)
- [bruteForce_bestPath](#)
 - [bruteForce.c](#), [35](#)
 - [bruteForce.h](#), [13](#)
- Distance, [7](#)
- [distance.c](#)
 - [distance_betweenTowns](#), [36](#)
 - [distance_calculDistance](#), [36](#)
 - [distance_new](#), [36](#)
- [distance.h](#)
 - [distance_betweenTowns](#), [14](#)
 - [distance_calculDistance](#), [15](#)
 - [distance_new](#), [15](#)
- [distance_betweenTowns](#)
 - [distance.c](#), [36](#)
 - [distance.h](#), [14](#)
- [distance_calculDistance](#)
 - [distance.c](#), [36](#)
 - [distance.h](#), [15](#)
- [distance_new](#)
 - [distance.c](#), [36](#)
 - [distance.h](#), [15](#)
- END
 - [parsing.h](#), [25](#)
- Errors, [8](#)
- [errors.c](#)
 - [errors_displayErrorsMessage](#), [38](#)
 - [errors_new](#), [38](#)
 - [errors_setFileNotFound](#), [38](#)
 - [errors_setMissingParameterGa](#), [38](#)
 - [errors_setMissingParameterLsnr](#), [38](#)
 - [errors_setMissingParameterLsr](#), [38](#)
 - [errors_setNoAlgoSpecified](#), [39](#)
 - [errors_setNoValidParameterGa](#), [39](#)
 - [errors_setNoValidParameterLsnr](#), [39](#)
 - [errors_setNoValidParameterLsr](#), [39](#)
 - [errors_setTagFNotFound](#), [39](#)
- [errors.h](#)
 - [errors_displayErrorsMessage](#), [16](#)
 - [errors_new](#), [17](#)
 - [errors_setFileNotFound](#), [17](#)
 - [errors_setMissingParameterGa](#), [17](#)
 - [errors_setMissingParameterLsnr](#), [17](#)
 - [errors_setMissingParameterLsr](#), [17](#)
 - [errors_setNbArguments](#), [17](#)
 - [errors_setNoAlgoSpecified](#), [17](#)
 - [errors_setNoValidParameterGa](#), [18](#)
 - [errors_setNoValidParameterLsnr](#), [18](#)
 - [errors_setNoValidParameterLsr](#), [18](#)
 - [errors_setTagFNotFound](#), [18](#)
- [errors_displayErrorsMessage](#)
 - [errors.c](#), [38](#)
 - [errors.h](#), [16](#)
- [errors_new](#)
 - [errors.c](#), [38](#)
 - [errors.h](#), [17](#)
- [errors_setFileNotFound](#)
 - [errors.c](#), [38](#)
 - [errors.h](#), [17](#)
- [errors_setMissingParameterGa](#)
 - [errors.c](#), [38](#)
 - [errors.h](#), [17](#)
- [errors_setMissingParameterLsnr](#)
 - [errors.c](#), [38](#)
 - [errors.h](#), [17](#)
- [errors_setMissingParameterLsr](#)
 - [errors.c](#), [38](#)
 - [errors.h](#), [17](#)
- [errors_setNbArguments](#)
 - [errors.h](#), [17](#)
- [errors_setNoAlgoSpecified](#)
 - [errors.c](#), [39](#)
 - [errors.h](#), [17](#)
- [errors_setNoValidParameterGa](#)
 - [errors.c](#), [39](#)
 - [errors.h](#), [18](#)
- [errors_setNoValidParameterLsnr](#)
 - [errors.c](#), [39](#)
 - [errors.h](#), [18](#)

- errors_setNoValidParameterLsr
 - errors.c, [39](#)
 - errors.h, [18](#)
- errors_setTagFNotFound
 - errors.c, [39](#)
 - errors.h, [18](#)
- GENETIC
 - parsing.h, [25](#)
- genetic.c
 - genetic_DPX, [40](#)
 - genetic_getBestPath, [40](#)
 - genetic_mutation, [41](#)
- genetic.h
 - genetic_DPX, [19](#)
 - genetic_getBestPath, [19](#)
 - genetic_mutation, [20](#)
- genetic_DPX
 - genetic.c, [40](#)
 - genetic.h, [19](#)
- genetic_getBestPath
 - genetic.c, [40](#)
 - genetic.h, [19](#)
- genetic_mutation
 - genetic.c, [41](#)
 - genetic.h, [20](#)
- Instance, [9](#)
- instance.c
 - instance_display, [42](#)
 - instance_displayLinearVector, [42](#)
 - instance_displayMatrix, [42](#)
 - instance_initializeDistancesMatrix, [42](#)
 - instance_new, [42](#)
 - instance_push, [43](#)
- instance.h
 - instance_display, [21](#)
 - instance_displayLinearVector, [21](#)
 - instance_displayMatrix, [21](#)
 - instance_initializeDistancesMatrix, [22](#)
 - instance_new, [22](#)
 - instance_push, [22](#)
- instance_display
 - instance.c, [42](#)
 - instance.h, [21](#)
- instance_displayLinearVector
 - instance.c, [42](#)
 - instance.h, [21](#)
- instance_displayMatrix
 - instance.c, [42](#)
 - instance.h, [21](#)
- instance_initializeDistancesMatrix
 - instance.c, [42](#)
 - instance.h, [22](#)
- instance_new
 - instance.c, [42](#)
 - instance.h, [22](#)
- instance_push
 - instance.c, [43](#)
 - instance.h, [22](#)
- lib/bruteForce.h, [13](#)
- lib/distance.h, [14](#)
- lib/errors.h, [15](#)
- lib/genetic.h, [18](#)
- lib/instance.h, [20](#)
- lib/localSearch.h, [22](#)
- lib/parsing.h, [24](#)
- lib/path.h, [26](#)
- lib/tour.h, [27](#)
- lib/town.h, [30](#)
- lib/util.h, [31](#)
- localSearch.c
 - localSearch_randomBestPath, [44](#)
 - localSearch_systematicBestPath, [44](#)
- localSearch.h
 - localSearch_randomBestPath, [23](#)
 - localSearch_systematicBestPath, [23](#)
- LOCALSEARCH_RANDOM
 - parsing.h, [25](#)
- LOCALSEARCH_SYSTEMATIC
 - parsing.h, [25](#)
- localSearch_randomBestPath
 - localSearch.c, [44](#)
 - localSearch.h, [23](#)
- localSearch_systematicBestPath
 - localSearch.c, [44](#)
 - localSearch.h, [23](#)
- main
 - main.c, [45](#)
- main.c
 - main, [45](#)
- parsing.c
 - parsing_algoType, [45](#)
 - parsing_parseFileName, [46](#)
 - parsing_parseVerboseMode, [46](#)
- parsing.h
 - AlgoType, [24](#)
 - BRUTEFORCE, [25](#)
 - END, [25](#)
 - GENETIC, [25](#)
 - LOCALSEARCH_RANDOM, [25](#)
 - LOCALSEARCH_SYSTEMATIC, [25](#)
 - parsing_algoType, [25](#)
 - parsing_parseFileName, [25](#)
 - parsing_parseVerboseMode, [25](#)

parsing_algoType
 parsing.c, 45
 parsing.h, 25
parsing_parseFileName
 parsing.c, 46
 parsing.h, 25
parsing_parseVerboseMode
 parsing.c, 46
 parsing.h, 25
Path, 10
path.c
 path_addNearNeighbor, 47
 path_display, 47
 path_new, 47
path.h
 path_addNearNeighbor, 26
 path_display, 27
 path_new, 27
path_addNearNeighbor
 path.c, 47
 path.h, 26
path_display
 path.c, 47
 path.h, 27
path_new
 path.c, 47
 path.h, 27

src/bruteForce.c, 34
src/distance.c, 35
src/errors.c, 37
src/genetic.c, 39
src/instance.c, 41
src/localSearch.c, 43
src/main.c, 44
src/parsing.c, 45
src/path.c, 46
src/tour.c, 48
src/town.c, 51
src/util.c, 52

Tour, 10
tour.c
 tour_2opt, 49
 tour_addSeveralTowns, 49
 tour_addTown, 49
 tour_calculLength, 49
 tour_display, 49
 tour_new, 50
 tour_nextPermutation, 50
 tour_randomWalk, 50
 tour_replaceTheWorstTour, 50
tour.h
 tour_2opt, 28
 tour_addSeveralTowns, 28
 tour_addTown, 29
 tour_calculLength, 29
 tour_display, 29
 tour_new, 29
 tour_nextPermutation, 29
 tour_randomWalk, 30
 tour_replaceTheWorstTour, 30
tour_2opt
 tour.c, 49
 tour.h, 28
tour_addSeveralTowns
 tour.c, 49
 tour.h, 28
tour_addTown
 tour.c, 49
 tour.h, 29
tour_calculLength
 tour.c, 49
 tour.h, 29
tour_display
 tour.c, 49
 tour.h, 29
tour_new
 tour.c, 50
 tour.h, 29
tour_nextPermutation
 tour.c, 50
 tour.h, 29
tour_randomWalk
 tour.c, 50
 tour.h, 30
tour_replaceTheWorstTour
 tour.c, 50
 tour.h, 30
Town, 11
town.c
 town_display, 51
 town_new, 51
town.h
 town_display, 31
 town_new, 31
town_display
 town.c, 51
 town.h, 31
town_new
 town.c, 51
 town.h, 31
util.c
 util_deleteArrayValue, 53
 util_displayArray, 53
 util_rand, 53
 util_reverseArray, 53

- [util_searchFirstOccurenceInArray](#), [53](#)
 - [util_sousTabExist](#), [54](#)
 - [util_sum](#), [54](#)
 - [util_swap](#), [54](#)
- [util.h](#)
 - [util_deleteArrayValue](#), [32](#)
 - [util_displayArray](#), [33](#)
 - [util_rand](#), [33](#)
 - [util_reverseArray](#), [33](#)
 - [util_searchFirstOccurenceInArray](#), [33](#)
 - [util_sousTabExist](#), [33](#)
 - [util_sum](#), [34](#)
 - [util_swap](#), [34](#)
- [util_deleteArrayValue](#)
 - [util.c](#), [53](#)
 - [util.h](#), [32](#)
- [util_displayArray](#)
 - [util.c](#), [53](#)
 - [util.h](#), [33](#)
- [util_rand](#)
 - [util.c](#), [53](#)
 - [util.h](#), [33](#)
- [util_reverseArray](#)
 - [util.c](#), [53](#)
 - [util.h](#), [33](#)
- [util_searchFirstOccurenceInArray](#)
 - [util.c](#), [53](#)
 - [util.h](#), [33](#)
- [util_sousTabExist](#)
 - [util.c](#), [54](#)
 - [util.h](#), [33](#)
- [util_sum](#)
 - [util.c](#), [54](#)
 - [util.h](#), [34](#)
- [util_swap](#)
 - [util.c](#), [54](#)
 - [util.h](#), [34](#)