📚 README.md : Solveur Modulaire du Problème du Voyageur de Commerce (TSP)

ℹ️ Vue d'Ensemble du Projet

Ce projet implémente un solveur polyvalent pour le **Problème du Voyageur de Commerce (TSP)** en langage **C**.

Le cœur du programme utilise un design modulaire basé sur des **pointeurs de fonctions génériques** (`types.h`) pour permettre l'échange dynamique des algorithmes de résolution et des fonctions de calcul de distance.

🔑 Stratégies de Résolution

Le solveur offre quatre stratégies de résolution distinctes :

1. **Force Brute (FB) :** Solution **optimale** pour N <= 12.
2. **Nearest Neighbor (NN) :** Heuristique constructive **rapide** (O(N^2)).
3. **Random Walk (RW) :** Heuristique constructive **aléatoire** (O(N)).
4. **Algorithme Génétique (GA) :** Méta-heuristique **performante** pour N > 50.

👨‍💻 Équipe et Rôles

Rôle	Membre(s)
Responsable Projet (RP)	Diogo CRUZ
Responsable Test (RT)	Diogo CRUZ
Responsable Développeur 3 (RD3)	Diogo CRUZ
Responsable Git (RG)	Camille MAGNE
Responsable Développeur 1 (RD1)	Camille MAGNE
Responsable Canevas (RCA)	Esso-Ylow-Noyou Trésor PANA
Responsable Développeur 2 (RD2)	Esso-Ylow-Noyou Trésor PANA

🛠️ Instructions de Compilation

Prérequis

- Compilateur C (GCC ou Clang) compatible avec C11.
- Librairie mathématique standard (`-lm`).

Commandes `make`

Pour compiler l'exécutable `main` :
`make`

Pour supprimer l'exécutable et les fichiers objets :

`make clean`

🚀 Scripts et Exécution

📝 Format du Fichier d'Entrée

Les fichiers doivent être au format **TSPLIB** (e.g., `NAME`, `DIMENSION`, `NODE_COORD_SECTION`).

Les fichiers tsp doivent être placés dans le sous-répertoire `tests/`.

Pour les test python il faut que le fichier soit dans le répertoire tests

Pour le main avec -f il faut passer le chemin relatif(par rapport au exécutable) ou absolue

Syntaxe de la Ligne de Commande

L'exécution est pilotée par les options suivantes :

Flags	Description	Valeurs
`-f`	**Fichier d'entrée TSPLIB** (Obligatoire).	`<file.tsp>`
`-m`	**Méthode de résolution**.	`bf` / `bfm` / `nn` / `rw` / `2optnn` / `2optrw` / `ga` / `gadpx`
`-d`	Type de **Distance** (surcharge l'auto-détection).	`{eucl2d | att | geo}`
`-o`	Fichier de sortie pour les résultats.	`<output.txt>`
`-c`	Afficher la longueur de la **tournée canonique**.	*(Aucune valeur)*

Flags	Description	Valeurs
`-h`	**Affiche l'aide** et quitte le programme.	*(Aucune valeur)*

Paramètres GA (`-m ga` ou `-m gadpx`)

Si la méthode GA est choisie, les paramètres suivants doivent suivre l'option `-m` :

-m ga <population> <taille_tournoi> <taux_mutation>

🚨 Interruption Contrôlée (Ctrl+C)

Les méthodes **`bf`** et **`bfm`** supportent le gestionnaire de signal. En cas de `Ctrl+C`, le programme affiche la **meilleure tournée trouvée jusqu'à présent** et demande si l'utilisateur souhaite continuer ou s'arrêter.

📊 Algorithmes, Performances et Limitations

Mesure des Performances

Le temps d'exécution réel (en secondes) est mesuré pour chaque méthode de résolution (`run()` dans `exec.c`) et est inclus dans la sortie formatée.

1. Méthodes de Résolution

Algorithme	Option(s)	Type / Complexité	Cas conseillé (N)	Notes
Force Brute	`bf`, `bfm`	Optimal / O(N!)	<=13	`bfm` utilise une demi-matrice pour l'optimisation.
Heuristiques	`2optnn`, `2optrw`	Amélioration locale / O(N^3)	Grande	Le **2-opt** utilise `reverse_segment` pour minimiser les croisements.
Algorithme Génétique	`ga`, `gadpx`	O(N^3)	>=50	`gadpx` intègre le **2-opt** pour affiner les descendants.

2. Types de Distance Supportés

Le solveur gère les normes de distance :

- **Euclidienne 2D (`eucl2d`)**
- **Pseudo-Euclidienne ATT (`att`)**
- **Géographique GEO (`geo`)**

La distance est **détectée automatiquement** à partir du fichier TSPLIB, mais peut être forcée par l'option `-d`.

Instances de Test

Instance	N	Type de Distance	Objectif de Test	Méthode Recommandée
`att10`	10	**ATT**	Tester l'**Optimum** et la fonction `dist_att`.	`bfm`
`burma14`	14	**GEO**	Tester la limite de `bfm` et la fonction **`dist_geo`**.	`bfm` (long) ou `2optnn`
`ali535`	535	**GEO**	Tester les performances de `gadpx` sur une très grande instance **GEO**.	`2optnn`
`att15`	15	**ATT**	Tester les performances sur des petit instances	`gadpx`
`att48`	48	**ATT**	Tester les performances sur des petit instances	`gadpx`
`fnl4461`	4461	**EUC_2D**	Tester les performances sur des grands instances	`2optnn`

📘 Documentation API (Doxygen)

La documentation complète de l'API (fonctions, structures, graphes d'appel/inclusion) peut être générée en HTML :

- **Commande :** `doxygen Doxyfile, make docs, ou juste make.`
- **Accès :** Ouvrez `docs/html/index.html` après génération.
    - **Astuce :** Une fois dans le dossier html, utilisez Ctrl f pour trouver index.html.