

ECOLE POLYTECHNIQUE DE L'UNIVERSITÉ FRANÇOIS RABELAIS DE TOURS

Département Informatique

64 avenue Jean Portalis

37200 Tours, France

Tél. +33 (0)2 47 36 14 14

polytech.univ-tours.fr

Projet Recherche & Développement 2017-2018

Framework interactif pour tester les algorithmes de Graph Matching

**POLYTECH[®]**
TOURS

Tuteurs académiques

Mostafa DARWICHE

Romain RAVEAUX

Donatello CONTE

Vincent T'KINDT

Étudiant

Kai ZENG (DI5)



Liste des intervenants

| Nom | Email | Qualité |
|------------------|--|--|
| Kai ZENG | kai.zeng@etu.univ-tours.fr | Étudiant DI5 |
| Mostafa DARWICHE | mostafa.darwiche@univ-tours.fr | Tuteur académique, Département Informatique |
| Romain RAVEAUX | romain.raveaux@univ-tours.fr | Tuteur académique, Département Informatique |
| Donatello CONTE | donatello.conte@univ-tours.fr | Tuteur académique, Département Informatique |
| Vincent T'KINDT | tkindt@univ-tours.fr | Tuteur académique, Département Informatique |



Avertissement

Ce document a été rédigé par Kai ZENG susnommé l'auteur.

L'Ecole Polytechnique de l'Université François Rabelais de Tours est représentée par Mostafa Darwiche, Romain Raveaux, Donatello Conte et Vincent T'kindt susnommés les tuteurs académiques.

Par l'utilisation de ce modèle de document, l'ensemble des intervenants du projet acceptent les conditions définies ci-après.

L'auteur reconnaît assumer l'entière responsabilité du contenu du document ainsi que toutes suites judiciaires qui pourraient en découler du fait du non respect des lois ou des droits d'auteur.

L'auteur atteste que les propos du document sont sincères et assument l'entière responsabilité de la véracité des propos.

L'auteur atteste ne pas s'approprier le travail d'autrui et que le document ne contient aucun plagiat.

L'auteur atteste que le document ne contient aucun propos diffamatoire ou condamnable devant la loi.

L'auteur reconnaît qu'il ne peut diffuser ce document en partie ou en intégralité sous quelque forme que ce soit sans l'accord préalable des tuteurs académiques et de l'entreprise.

L'auteur autorise l'école polytechnique de l'université François Rabelais de Tours à diffuser tout ou partie de ce document, sous quelque forme que ce soit, y compris après transformation en citant la source. Cette diffusion devra se faire gracieusement et être accompagnée du présent avertissement.



Pour citer ce document

Kai ZENG, *Framework interactif pour tester les algorithmes de Graph Matching*, Projet Recherche & Développement, Ecole Polytechnique de l'Université François Rabelais de Tours, Tours, France, 2017-2018.

```
@mastersthesis{
  author={ZENG, Kai},
  title={Framework interactif pour tester les algorithmes de Graph Matching},
  type={Projet Recherche \& Développement},
  school={Ecole Polytechnique de l'Université François Rabelais de Tours},
  address={Tours, France},
  year={2017-2018}
}
```

Table des matières

| | |
|---|----------|
| Liste des intervenants | a |
| Avertissement | b |
| Pour citer ce document | c |
| Table des matières | i |
| Table des figures | vi |
| 1 Introduction | 1 |
| 1 Les acteurs du projet | 1 |
| 2 Contexte de la réalisation | 2 |
| 2.1 Contexte | 2 |
| 2.2 Description du problème | 2 |
| 2.3 Existant | 2 |
| 3 Objectifs | 3 |
| 4 Bases méthodologiques | 3 |
| 2 Description générale | 5 |
| 1 Environnement du projet | 5 |
| 2 Caractéristiques des utilisateurs | 5 |
| 3 Fonctionnalités du système | 6 |
| 4 Structure générale du système | 7 |
| 4.1 Le diagramme d'architecture générale du système | 7 |
| 4.2 Le diagramme de la structure de base de données | 8 |

| | | |
|----------|--|-----------|
| 3 | Etat de l'art/veille | 10 |
| 1 | Graphe | 10 |
| 2 | Graph Matching..... | 10 |
| 3 | Graph Edit Distance | 11 |
| 3.1 | L'opération d'édition..... | 11 |
| 3.2 | Le coût..... | 12 |
| 4 | Les Méthodes de GED | 12 |
| 4.1 | Les méthodes exactes..... | 12 |
| 4.2 | Les méthodes heuristiques | 13 |
| 4.2.1 | BP(Bipartite graph matching) | 13 |
| 4.2.2 | Beam-search..... | 13 |
| 4.2.3 | SBPBeam | 13 |
| 4.2.4 | IPFP/GNCCP | 14 |
| 5 | Notre proposition : LocBra | 14 |
| 6 | L'Expérimentation et les comparaisons | 15 |
| 6.1 | Choix de base de données..... | 15 |
| 6.2 | Comparaison de LocBra avec l'approche exacte | 16 |
| 6.3 | Comparaison de LocBra avec l'heuristique de la littérature | 16 |
| 4 | Analyse et conception | 18 |
| 5 | Mise en oeuvre | 22 |
| 1 | Les outils et librairies utilisées..... | 22 |
| 2 | Limites et risques..... | 22 |
| 3 | Design pattern | 23 |
| 4 | Framework –Hibernate..... | 24 |
| 5 | Les outils de gestion du projet | 24 |
| 6 | Mise en place des tests..... | 24 |
| 7 | Qualité du code | 25 |
| 8 | Résultat et performance | 25 |
| 6 | Bilan et conclusion | 26 |
| 1 | Point sur Fait et Reste à faire/retards S9..... | 26 |
| 2 | Plan de développement S10 | 27 |
| 3 | Point sur Fait et Reste à faire/retards S10..... | 27 |
| 4 | Bilan sur la qualité | 28 |
| 5 | Bilan auto-critique sur la gestion du projet | 28 |
| 6 | Conclusion | 28 |

| | |
|---|-----------|
| Annexes | 29 |
| A Description des interfaces externes du logiciel | 30 |
| 1 Interfaces matériel/logiciel | 30 |
| 2 Interfaces logiciel/logiciel | 30 |
| 3 Interfaces homme/machine | 30 |
| B Spécifications fonctionnelles | 36 |
| 1 Ajout d'une méthode | 36 |
| 2 Ajout d'un Dataset/Subset | 36 |
| 3 Ajout d'un Test | 37 |
| 4 Ajout d'une exécution | 37 |
| 5 Liste de Méthode/Dataset/Test/Exécution | 37 |
| 6 Recherche de Méthode/Dataset/Test/Exécution dans la liste | 37 |
| 7 Sauvegardes des résultats | 37 |
| 8 Affiche des résultats dans le tableau ou dans le diagramme | 38 |
| 9 Visualisation | 38 |
| C Spécifications non fonctionnelles | 39 |
| 1 Contraintes de développement et conception | 39 |
| 2 Contraintes de fonctionnement et d'exploitation | 39 |
| 2.1 Performances | 39 |
| 2.2 Capacités | 39 |
| 2.3 Contrôlabilité | 40 |
| 2.4 Sécurité | 40 |
| 3 Maintenance et évolution du système | 40 |
| D Analyse et cahier du développeur | 41 |
| 1 Le structure du projet | 41 |
| 1.1 Modèles | 41 |
| 1.2 Contrôleurs | 42 |
| 1.3 Vues | 42 |
| 2 Optimisation de structure de la base de données | 43 |
| 3 Implémenter des heuristiques en les intégrant dans le système | 44 |
| 3.1 Module de « Méthode » | 45 |
| 3.2 Module de « Test » | 46 |
| 3.3 Module de « Exécution » | 48 |
| 4 Evaluer les méthodes différentes sur des bases d'images | 49 |
| 5 Afficher les résultats de comparaison dans des tableaux ou diagrammes | 50 |
| 6 Réaliser la visualisation de Matching trouvé entre deux graphes | 50 |

| | | |
|----------|--|-----------|
| 6.1 | La couche « Model » lit le contenu en texte brut du fichier « .dat » | 51 |
| 6.2 | Lire la table correspondante des nœuds et des arcs générés | 51 |
| 6.3 | Diviser le contenu du texte brut du fichier « .dat » | 51 |
| 6.4 | Lire la collection graphique sélectionnée par l'utilisateur | 51 |
| 6.5 | Pré-traitement des données | 51 |
| 6.6 | Visualisation | 51 |
| 6.6.1 | Pas de coordonnées claires : MUTA et PAH | 52 |
| 6.6.2 | Avoir des coordonnées claires : AUCUNNOM | 52 |
| 6.7 | Interaction de l'utilisateur | 52 |
| E | Gestion de projet | 54 |
| 1 | Méthode de suivi de projet | 54 |
| 2 | Découpage du projet en tâches | 54 |
| 2.1 | Tâche 1 : L'étude du besoin | 55 |
| 2.2 | Tâche 2 : Documentation et lecture | 55 |
| 2.3 | Tâche 3 : Etat de l'art sur la Graph Edit Distance | 56 |
| 2.4 | Tâche 4 : Implémentation et Analyse du projet existant | 56 |
| 2.5 | Tâche 5 : Analyse et Modélisation | 56 |
| 2.6 | Tâche 6 : Préparation du rapport S9 et soutenance S9 | 56 |
| 2.7 | Tâche 7 : Réalisation des fonctionnalités | 56 |
| 2.8 | Tâche 8 : Test de la programmation et reprise du projet | 57 |
| 2.9 | Tâche 9 : Préparation du rapport S10 et soutenance S10 | 57 |
| 3 | Planning prévisionnel S9 et S10 | 57 |
| 4 | Planning réel S9 et analyse | 58 |
| 5 | Planning plus détaillé de développement S10 | 59 |
| 6 | Planning réel S10 et analyse | 60 |
| F | Doc d'installation/déploiement | 62 |
| 1 | Installation locale | 62 |
| 2 | Déploiement sur le serveur | 63 |
| G | Doc d'utilisation | 66 |
| H | Dossiers de tests | 76 |
| 1 | Plan de test | 76 |
| 1.1 | Test unitaire | 76 |
| 1.2 | Test d'intégration | 76 |
| 1.3 | Test fonctionnels | 76 |
| 2 | Tests unitaires avec Junit | 76 |
| 3 | Tests d'intégration | 77 |
| 4 | Tests fonctionnnels | 77 |

| | |
|---------------|----|
| Webographie | 80 |
| Bibliographie | 81 |

Table des figures

1 Introduction

| | | |
|---|---|---|
| 1 | L'exemple de graphes modélisant des objets..... | 3 |
|---|---|---|

2 Description générale

| | | |
|---|---|---|
| 1 | Le diagramme de processus du système | 6 |
| 2 | Le diagramme d'utilisation | 7 |
| 3 | L'architecture générale du système | 8 |
| 4 | Le diagramme MCD de la structure de base de données | 9 |
| 5 | Le diagramme MLD de la structure de base de données..... | 9 |

3 Etat de l'art/veille

| | | |
|---|--|----|
| 1 | Graphiques dans divers domaines | 11 |
| 2 | Une exemple d'une opération d'édition..... | 12 |
| 3 | Diagramme d'algorithme | 15 |

4 Analyse et conception

| | | |
|---|---|----|
| 1 | Le diagramme de classe DAO & Model..... | 19 |
| 2 | La relation entre la classe « Test » et la classe « Execution » | 20 |
| 3 | Le diagramme de classe Servlet | 21 |

5 Mise en oeuvre

| | | |
|---|-------------------------------|----|
| 1 | Versionning avec Github | 24 |
|---|-------------------------------|----|

A Description des interfaces externes du logiciel

| | | |
|----|---|----|
| 1 | La page d'accueil | 31 |
| 2 | La page de Liste des Méthodes..... | 31 |
| 3 | La page de l'ajout d'une méthode exacte | 32 |
| 4 | La page de l'ajout d'une méthode heuristique..... | 32 |
| 5 | La page de Liste des Datasets..... | 33 |
| 6 | La page de l'ajout d'un Dataset | 33 |
| 7 | La page de Liste des Tests..... | 34 |
| 8 | La page de l'ajout d'un test | 34 |
| 9 | L'affiche de détail d'une exécution | 35 |
| 10 | L'interface du résultat..... | 35 |

D Analyse et cahier du développeur

| | | |
|----|--|----|
| 1 | Convention de nommage et de commentaire | 41 |
| 2 | L'architecture générale du système | 42 |
| 3 | La structure des différentes modèles | 43 |
| 4 | Le diagramme de Model & DAO | 44 |
| 5 | Le contenu des différentes contrôleurs | 45 |
| 6 | Une exemple pour expliquer le servlet comme contrôleur | 45 |
| 7 | La structure des différentes vues | 46 |
| 8 | Le diagramme MCD pour la partie ajoutée&modifiée | 46 |
| 9 | Le diagramme MLD pour la partie ajoutée&modifiée | 47 |
| 10 | Les mappings pour les classes ajoutées | 47 |
| 11 | La couche d'accès aux données utilisée..... | 47 |
| 12 | Les fonctions importantes dans la classe de « Execution »..... | 49 |
| 13 | Ajouter d'autres types de collections graphiques | 53 |
| 14 | Comparer plusieurs graphiques dans le futur système(peut-être) | 53 |

E Gestion de projet

| | | |
|---|---|----|
| 1 | Capture d'écran de mon Trello..... | 54 |
| 2 | Découpage du projet en tâches | 55 |
| 3 | Le diagramme de Gantt prévisionnel au début | 58 |
| 4 | Le diagramme de Gantt réel S9..... | 58 |
| 5 | Le diagramme de Gantt réel S10..... | 60 |

F Doc d'installation/déploiement

| | | |
|---|---|----|
| 1 | Ajouter Apache Tomcat v9.0 comme environnement d'exécution du serveur | 63 |
| 2 | Démarrer le serveur..... | 63 |

| | | |
|---|---|----|
| 3 | Exporter le package de WAR..... | 64 |
| 4 | Le serveur a été démarré avec succès..... | 64 |
| 5 | Connecter à la base de données H2..... | 65 |

G Doc d'utilisation

| | | |
|----|---|----|
| 1 | La page d'accueil | 66 |
| 2 | Ajouter une méthode exacte..... | 67 |
| 3 | Ajouter une méthode heuristique | 67 |
| 4 | La liste de méthode..... | 68 |
| 5 | Modification de la méthode | 68 |
| 6 | Modification de la méthode non plus | 69 |
| 7 | Ajouter un DataSet/SubSet | 69 |
| 8 | La liste de DataSet/SubSet | 69 |
| 9 | Les choix de algos de type différent ne sont valides..... | 70 |
| 10 | Ajouter un test pour comparer les algos exactes | 70 |
| 11 | Ajouter un test pour comparer les algos heuristiques..... | 71 |
| 12 | La boîte de paramètres | 71 |
| 13 | La liste de Test | 72 |
| 14 | Ajouter une Execution..... | 72 |
| 15 | Exécuter un test | 73 |
| 16 | « Général » | 73 |
| 17 | « Temps CPU »..... | 73 |
| 18 | « Instances traitées »..... | 74 |
| 19 | « Déviation » | 74 |
| 20 | « Appareillement » | 74 |
| 21 | « molecule_3601 » et « molecule_3875 » | 75 |
| 22 | La visualisation de Graph Matching..... | 75 |

H Dossiers de tests

| | | |
|---|---|----|
| 1 | Le test unitaire de classe Parametre..... | 77 |
| 2 | Les tests unitaires de « Model » | 77 |

1

Introduction

Le Projet de Recherche & Développement (PR&D), réalisé en 5ème année, s'inscrit dans la formation dispensée à l'École Polytechnique de Tours et constitue une réelle expérience en terme de conduite de projet. Le PR&D se déroule du 20 septembre 2017 au 30 mars 2018 à raison de 2 jours par semaine à temps plein. Il donne lieu à la rédaction d'un rapport avec le cahier de spécification et de deux présentations du travail effectué lors de deux soutenances.

Le PR&D est orienté vers une étude théorique pour comprendre les algorithmes implémentés et une conception logicielle permettant la mise en œuvre de ces algorithmes et tester leurs performances en les comparant sur un ensemble d'instances. La MOA est représentée par Mostafa Darwiche, Romain Raveaux, Donatello Conte, Vincent T'kindt, membres du Laboratoire d'Informatique de Tours. Le projet s'inscrit dans un cadre théorique et appliqué qui fait l'objet d'une étude bibliographique, de l'existant et de veille technologique en amont, pour aboutir au développement d'une application.

Ce document est donc le rapport de système du projet "Framework interactif pour tester les algorithmes de Graph Matching". Il définit les besoins, l'environnement du projet, les objectifs à réaliser, l'analyse et la modélisation du système, et il contient l'état de l'art sur le problème à rechercher. Ce rapport présente aussi les différentes tâches à effectuer et le planning prévisionnel.

1 Les acteurs du projet

- La maîtrise d'œuvre (MOE) : Kai ZENG, étudiante en 5ème année de l'Ecole d'Ingénieur Polytech'Tours au sein du département Informatique, dans le cadre du PR&D.
- La maîtrise d'ouvrage (MOA) : Les encadrants :
 - Mostafa Darwiche, doctorant au Laboratoire d'Informatique de Tours
 - Romain Raveaux, maître de conférences de l'équipe de recherche RFAI du Laboratoire Informatique de Tours
 - Donatello Conte, maître de conférences de l'équipe de recherche RFAI du Laboratoire Informatique de Tours
 - Vincent T'kindt, professeur des universités de l'équipe de recherche ROOT ERL-CNRS 6305 du Laboratoire Informatique de Tours

Ce document a été rédigé par Kai ZENG et sera validé par les différents acteurs du projet.

2 Contexte de la réalisation

2.1 Contexte

Les problèmes de Graph Matching ou appariement de graphes sont des problèmes connus, intéressants et difficile à résoudre. Ils permettent de comparer des graphes entre eux et mesurer la similarité ou dis-similarité. Étant donné que les graphes sont considérés comme une bonne manière pour représenter des objets (ex. objet dans une image, une molécule chimique), la tâche de comparer ces graphes est donc un outil pour comparer les objets qu'ils représentent. On rencontre les problèmes de Graph Matching dans des applications comme : la reconnaissance de l'écriture manuscrite, la détection des objets dans les images, la détection des logiciels malveillants, etc.

Parmi les problèmes de Graph Matching, un des plus abordés est le Graph Edit Distance (GED), où on s'intéresse à trouver la distance minimale entre deux graphes ou le coût minimum des opérations pour transformer un graphe à un autre graphe.

GED est un problème NP-difficile, donc trouver un algorithme qui résout le problème et donne les meilleures solutions dans un temps acceptable n'est pas une tâche facile. Ce problème est traité par beaucoup des méthodes de type heuristiques ou exact comme la programmation linéaire en nombres entiers. Vu le nombre de méthodes croissant et la performance de méthode différente, il manque un système qui permet de les comparer entre eux, afin d'étudier leurs performances et précision.

C'est donc dans ce contexte que le chercheur propose de mettre en œuvre un système pour l'évaluation des modèles mathématiques et des méthodes heuristiques qui résolvent les problèmes de Graph Matching (comme le GED), soit des modèles linéaires, non-linéaires ou des heuristiques.

2.2 Description du problème

Nous pouvons prendre quelques exemples dans quelques domaines célèbres pour expliquer nos problèmes. Par exemple, dans le domaine de la reconnaissance de formes, les graphiques sont utilisés pour représenter des objets dans des images ou des vidéos et aussi pour exploiter les relations entre eux. Dans le domaine de la chimie et précisément en considérant les molécules chimiques, les graphes forment une représentation naturelle de la structure de liaison atomique des molécules. Chaque sommet du graphe représente un atome, tandis qu'un bord représente une liaison moléculaire.

La **Figure 1** montre un exemple de graphes modélisant des objets dans des images et des molécules chimiques. À gauche, il s'agit d'un exemple de graphes modélisant des objets dans des images, et à droite, c'est une molécule chimique avec son graphe associé. Après avoir modélisé les objets à l'aide de graphiques, une tâche importante consiste à comparer les graphiques entre eux. Dans l'espace graphique, la comparaison de graphique peut être réalisée en résolvant les problèmes de correspondance graphique. La solution d'un problème GM permet de comparer deux graphes et de trouver des similitudes entre les objets. Nous allons donc trouver la meilleure solution en utilisant différents algorithmes pour faire correspondre les graphes afin d'obtenir un résultat plus clair de la visualisation et de comparer leurs performances.

2.3 Existant

✓ Logiciel

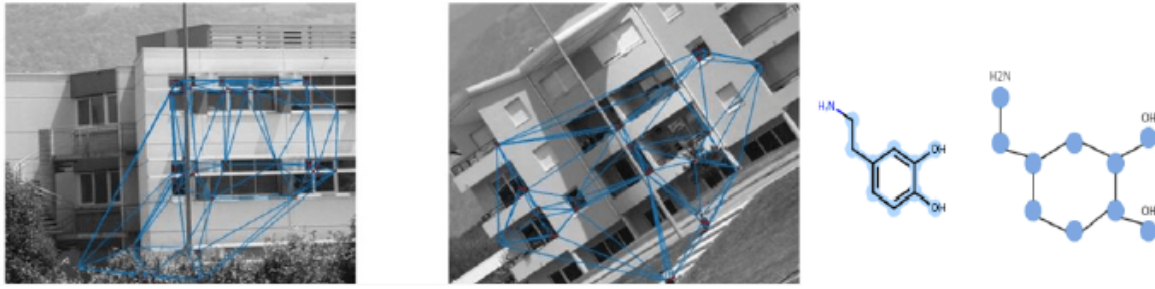


Figure 1 – *L'exemple de graphes modélisant des objets*

Le projet a été lancé l'année dernière, donc une partie est déjà réalisée : il y a l'interface graphique pour définir les modèles, les bases de données et la réalisation des tests. Le projet est implémenté en Java, JSP, JavaScript intégrant plusieurs librairies ex. Google Charts.

✓ **Algorithmique**

De nombreux algorithmes qui traitent du problème GED peuvent être trouvés. Ils peuvent être divisés en deux catégories : les méthodes exactes et heuristiques. Ces deux méthodes ont leurs propres avantages et inconvénients. Nous proposons également l'heuristique Local Branching (LocBra) pour traiter le problème GED^{EnA} qui se concentre principalement sur la recherche locale dans un voisinage spécifique pour une solution améliorée. Cette partie sera plus détaillée dans le **Chapitre 3** Etat de l'art/veille.

3 Objectifs

Le but de ce projet est d'améliorer l'interface développée l'année dernière permettant à l'utilisateur d'effectuer un test pour faire la comparaison de chaque algorithme de Graph Edit Distance à partir d'une méthode et d'une base de données d'images, et de visualiser les résultats. Cette interface sera modifiée afin de pouvoir intégrer de nouvelles fonctionnalités incluent :

- Modifier et effacer les erreurs rencontrées du système existant
- Améliorer les fonctionnalités de comparer les modèles mathématiques
- Étudier et Implémenter des heuristiques en les intégrant dans le système
- Évaluer ces méthodes différentes sur des bases d'images
 - L'utilisateur peut choisir quelles données et variables à afficher sous différents critères (ex. Moyen de temps d'exécution, déviation, nombre des solutions optimales ...)
- Présenter et afficher les résultats dans des tableaux ou diagrammes
- Réaliser la visualisation de Matching trouvé entre deux graphes

4 Bases méthodologiques

Durant ce projet, l'ensemble de la documentation est réalisé sous le format Latex.

Pour la modélisation du système, j'ai choisi d'utiliser le langage UML (Unified Modeling Language). Cela permet d'avoir une modélisation formelle et normée compréhensible par l'ensemble des intervenants sur le projet. Ce langage est utilisé dans ce projet afin de dessiner les diagrammes des cas d'utilisations ainsi que les diagrammes de classe.

Afin de conduire et gérer ce projet, l'outil GanttProject sera utilisé pour réaliser la planification des tâches et le suivi de celle-ci.

Dans le cas du développement d'une plateforme web, les différents langages liés à ce type d'application (JSP, CSS, Javascript, Java,...) sont utilisés.

La méthode de gestion de projet est la méthode Agile[[WWW2](#)], constituée de plusieurs sprints avec un enchaînement logique des tâches. Les sources du logiciel seront versionnées grâce à Github et le projet sera suivi grâce à Trello qui nous permet de gérer l'avancement du projet (Documentation, Planning, Rendez-vous, suivi de bugs,...).

Afin d'avoir un feedback régulier et de pouvoir fournir des livrables fréquemment, je vais mettre en place une soumise hebdomadaire comme compte-rendu.

2

Description générale

1 Environnement du projet

Le système permet aux utilisateurs d'ajouter des tests et de les exécuter pour comparer les méthodes. Voici un schéma **Figure 1** permettant de montrer le processus lors de l'utilisation de ce système. Avant d'ajouter un test, l'utilisateur doit importer la méthode d'calcul la distance d'édition de graphique(GED) et la base de données graphique et entrer les paramètres nécessaires. Ensuite, l'utilisateur peut effectuer ce test et les résultats de l'exécution seront affichés sur la page Web sous la forme d'un graphique. En outre, le document texte est automatiquement généré dans le dossier du projet pour enregistrer le résultat de l'exécution. Enfin, l'utilisateur peut cliquer sur le bouton pour visualiser la correspondance des deux graphiques.

Lors de la réalisation de ce projet, nous disposerons déjà d'éléments utilisables : les méthodes d'calcul de GED avec le format « .exe », les bases de données MUTA/PAH. Pour les méthodes, elles sont en essence les formulations qui s'appellent Programme linéaire en nombre entiers(MILP). Les formulations MILP sont très efficaces pour modéliser les problèmes d'optimisation combinatoire. Souvent, la solution de ces formulations est effectuée en utilisant un solveur de boîte noire, par exemple CPLEX. Donc on a besoin d'IBM®ILOG®CPLEX®Optimization Studio 12.6.0 pour développer et déployer des modèles d'optimisation.

Ces éléments utilisables seront supposés entièrement fonctionnels, intégrables et utilisables au sein du système. Comme ce PR&D est une amélioration de précédent projet, des contraintes de développement et d'environnement devront être respectées. Ainsi les tests et le développement seront réalisés en Java, sous une machine Windows/Mac, avec l'IDE Eclipse ou IntelliJ. Plusieurs bibliothèques externes devront être utilisées pour faire fonctionner les outils correctement. De plus, nous choisissons Apache Tomcat comme serveur pour publier JSP et Java.

De plus, ce système pourra être utilisé en interne, ou par un nombre d'utilisateurs en externe.

2 Caractéristiques des utilisateurs

Ce système est destinée à tous types d'utilisateurs bien qu'il ait été conçu à l'origine pour aider les chercheurs à mieux comparer les modèles mathématiques exacts et les méthodes heuristiques, donc il va disposer d'une interface intuitive simple, adaptée aux utilisateurs qui

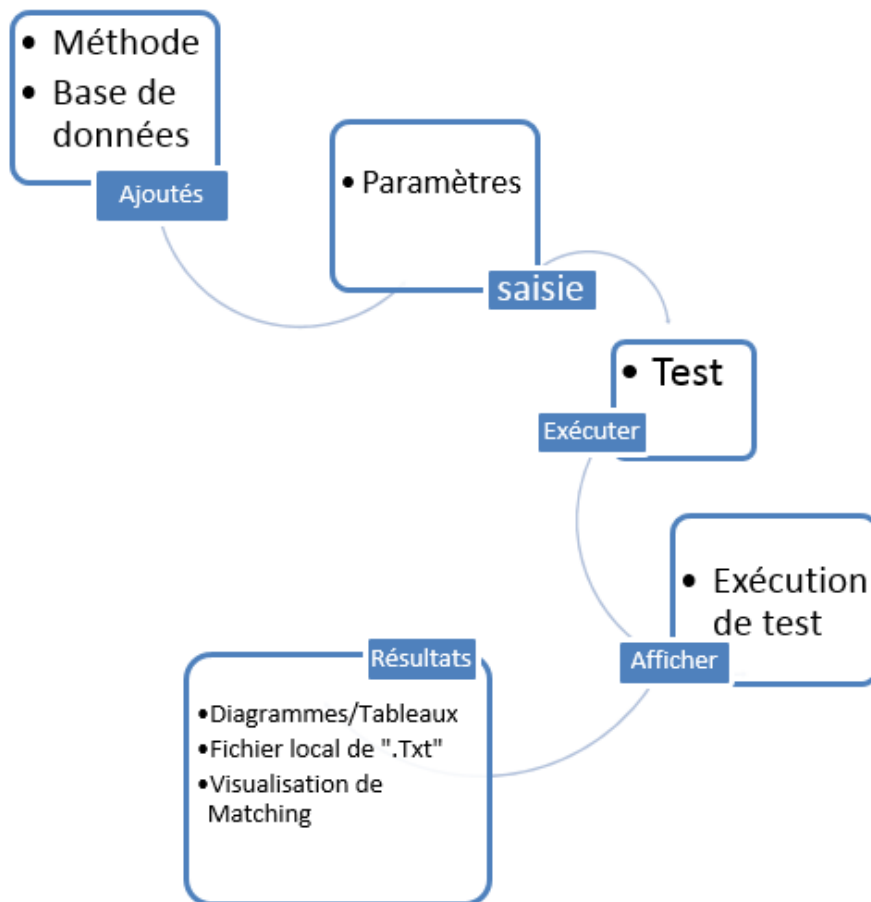


Figure 1 – Le diagramme de processus du système

n'ont pas de compétences pratiques en informatique. Tous les utilisateurs ont accès aux mêmes services donc aucun droits d'accès ne doit être géré ici.

3 Fonctionnalités du système

L'interface existante permet déjà de réaliser un certain nombre de fonctionnalités. Du coup maintenant l'utilisateur a ainsi la possibilité d'ajouter les bases d'images ainsi que les modèles. Aussi toutes les fonctionnalités liées à la comparaison de modèle exacte mathématique sont disponibles, à savoir le test et l'exécution de la formulation MILP à comparer. En même temps, l'utilisateur peut parcourir la liste de chaque module y compris le modèle, la base de données, le test et l'exécution.

Dans l'avenir, le système doit ajouter plusieurs méthodes heuristiques de calcul de la distance d'édition des graphes pour les utilisateurs et comparer l'efficacité de ces méthodes. Donc il est très important pour nous de modifier les fonctionnalités dans la partie de « Modèle ». En plus, une nouvelle fonctionnalité importante est la visualisation de Graph Matching trouvé entre deux graphes, cela permet à l'utilisateur d'évaluer visuellement les résultats des méthodes. Selon les besoins de l'utilisateur, toutes les fonctionnalités à modifier et toutes les fonctionnalités à ajouter suivant sont présentées dans le diagramme de cas d'utilisations et marquées des couleurs différentes pour montrer la distinction comme **Figure 2**.

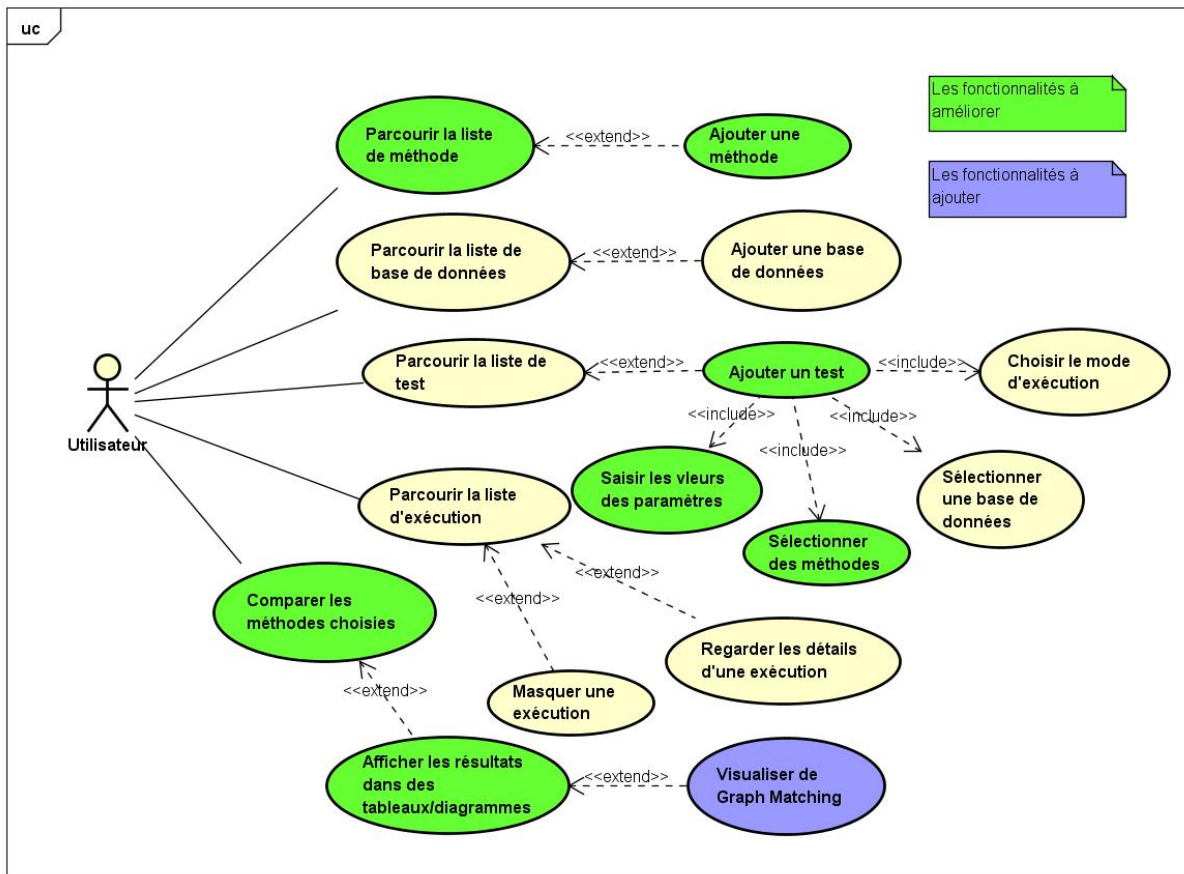


Figure 2 – Le diagramme d'utilisation

4 Structure générale du système

4.1 Le diagramme d'architecture générale du système

Voici le schéma **Figure 3** permettant de montrer l'architecture générale du système. Les utilisateurs importent des méthodes et des bases de données d'images pour calculer les GED sur des pages Web afin de tester et d'afficher les résultats d'exécution. Lorsque le serveur reçoit la requête de l'utilisateur, il accède à la page spécifiée et appelle les fonctions dans le back-end pour répondre à la demande de l'utilisateur. Le résultat de l'exécution est affiché sur la page Web et stocké dans le document « .txt » local. Dans le même temps, le back-end fera l'opération également la base de données via la couche DAO.

Comme une partie de la base de données a été complétée, la partie principale du projet pour cette année sera la partie de la zone rouge de la figure.

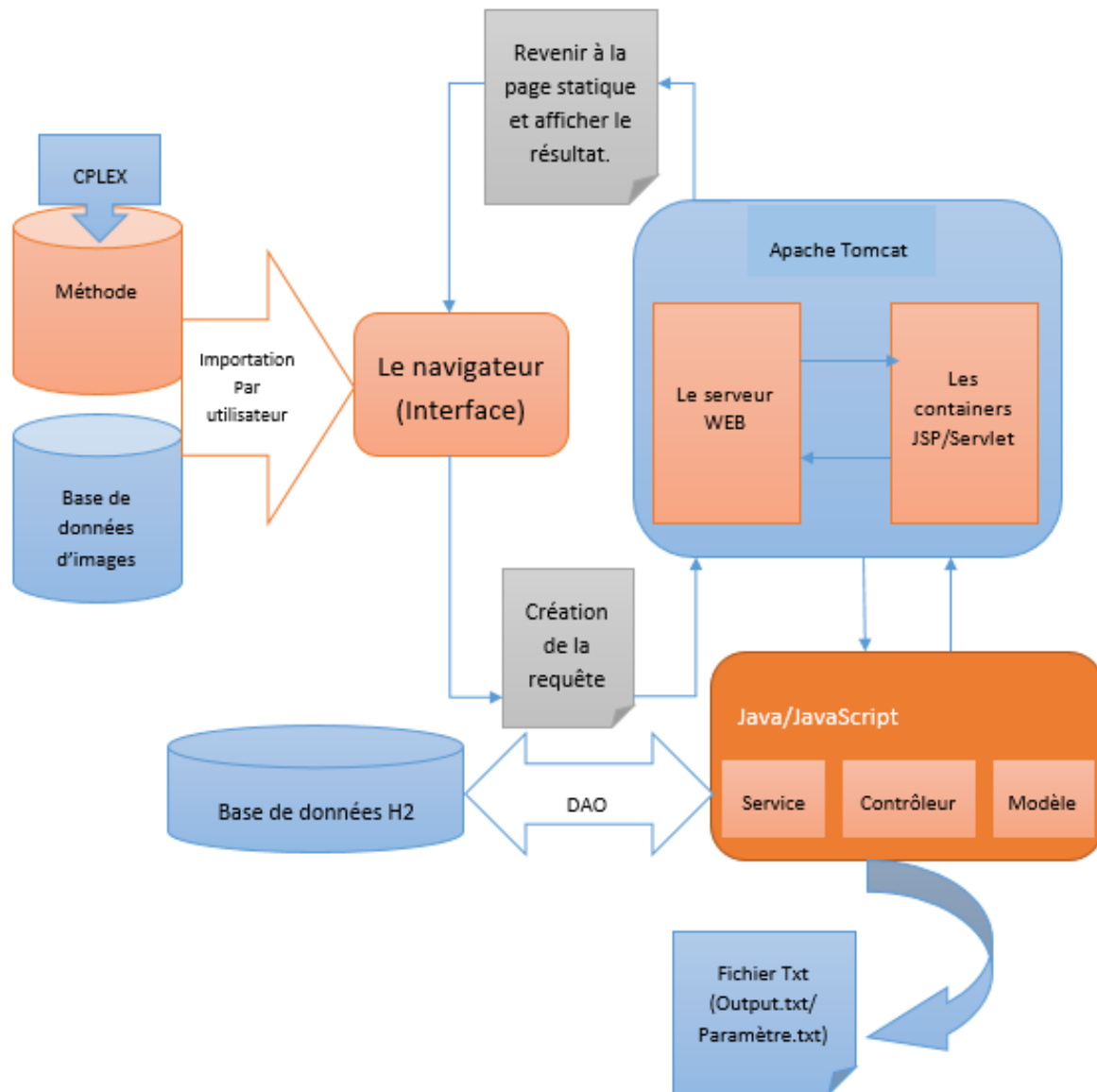


Figure 3 – L'architecture générale du système

4.2 Le diagramme de la structure de base de données

Étant donné que la fonctionnalité nouvellement ajoutée nécessite uniquement l'ajout d'une implémentation de la partie algorithme heuristique, nous n'avons pas besoin de modifier la structure de données précédente. Dans la modèle MCD, il y a 7 entités correspondant à tables, les relations entre les entités sont comme la diagramme [Figure 4](#). Parmi eux, l'entité « Test » et l'entité « Methode », l'entité « Test » et l'entité « Dataset » sont des relations plusieurs-à-plusieurs, et l'entité « Exécution » est la même. Cependant, la relation entre l'entité « Test » et l'entité « Exécution » est qu'une exécution ne peut correspondre qu'à un test. En outre, il existe les entités « Params » correspondant pour les entités « Test » et « Exécution ». Deux tables relationnelles sont nécessaires entre l'entité « Dataset » et l'entité « Subset » : la première table relationnelle indique la relation plusieurs-à-plusieurs entre les deux et la deuxième table relationnelle est utilisée pour montrer à quel Dataset appartient le Subset sélectionné par l'utilisateur. Le diagramme MLD est présenté dans [Figure 5](#).

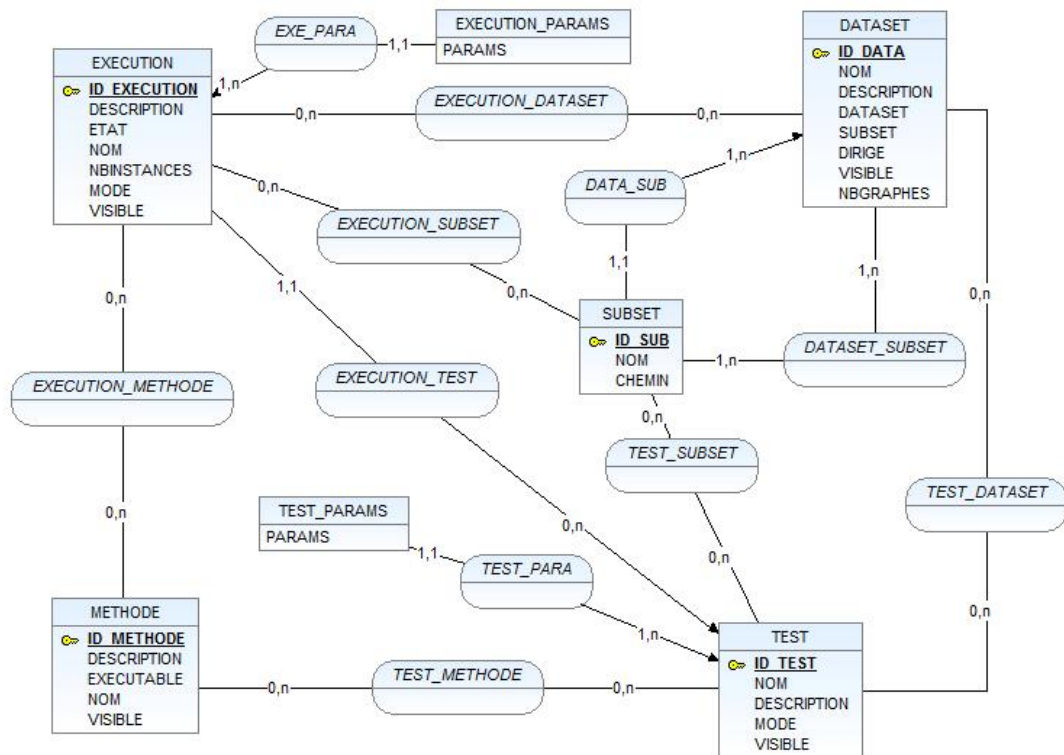


Figure 4 – Le diagramme MCD de la structure de base de données

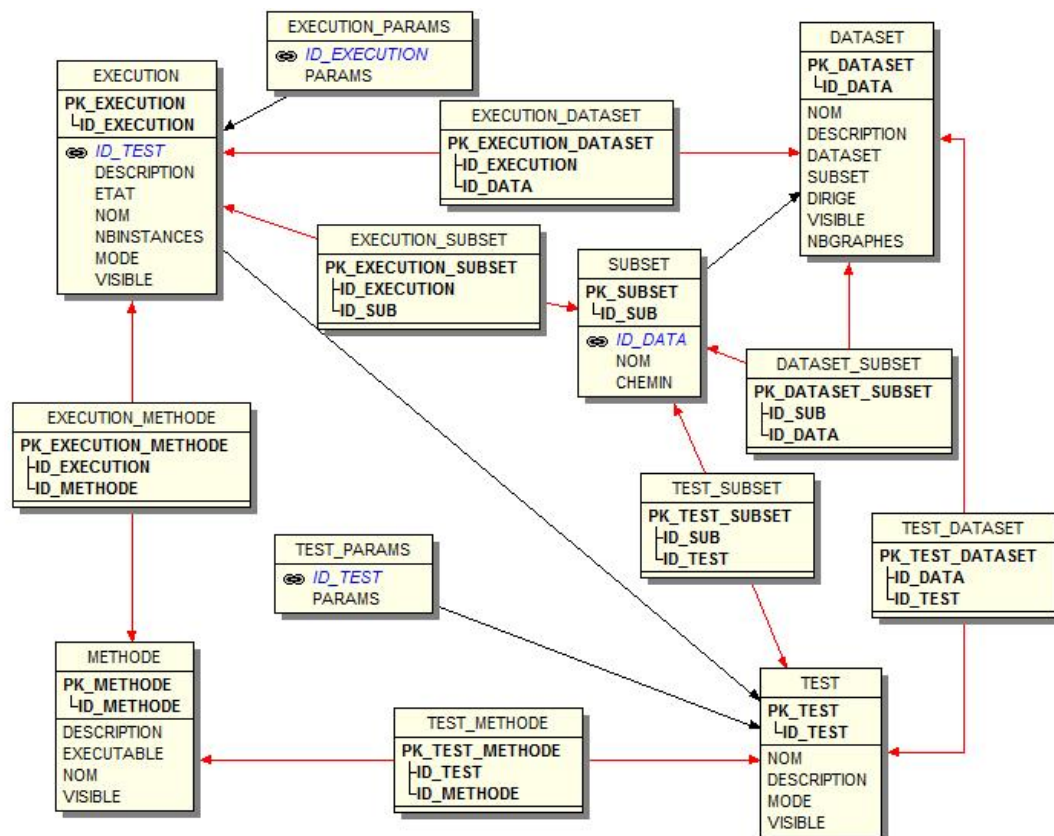


Figure 5 – Le diagramme MLD de la structure de base de données

3

Etat de l'art/veille

Dans cette partie je vais vous présenter toutes les recherches que j'ai pu effectuer lors de la première moitié de ce PR&D. Celles-ci ont été guidées par Monsieur Mostafa Darwiche, Monsieur Romain Raveaux, Monsieur Donatello Conte, Monsieur Vincent T'kindt. Le but de ces recherches est de pouvoir voir des méthodes et techniques différentes, essentiellement au niveau de la solution optimale et du temps, pour ensuite prendre une décision sur laquelle ou lesquels il serait pertinent de mettre en place, afin de répondre aux objectifs définis pour ce PR&D.

1 Graphe

Les graphes sont une structure de données générale et puissante pour la représentation d'objets et de concepts. Dans une représentation de graphe, les nœuds représentent généralement des objets ou des parties d'objets, tandis que les arcs décrivent les relations entre objets ou parties d'objet. Pour les graphes attribués, les deux sommets et les arcs peuvent être caractérisés par des attributs qui peuvent varier d'étiquettes nominales à des descriptions plus complexes telles que des chaînes ou des vecteurs de caractéristiques.

Dans des applications comme affichées dans la [Figure 1](#) telles que la reconnaissance de formes et la vision par ordinateur, la similitude des objets est un problème important. Compte tenu d'une base de données d'objets connus et d'une requête, la tâche est de récupérer un ou plusieurs objets de la base de données qui sont similaires à la requête. Si les graphes sont utilisés pour la représentation d'objets, ce problème se transforme en déterminant la similitude des graphes, généralement appelée correspondance graphique (Graph Matching).

2 Graph Matching

Graph Matching est un problème fondamental dans Computer Vision et Machine Learning. De nombreux problèmes d'intérêt pour Computer Vision et Machine Learning peuvent être formulés comme un problème de correspondance : trouver un mappage entre un ensemble de nœuds et un autre ensemble de nœuds. Parce que ces ensembles de nœuds peuvent avoir une structure interne importante, ils sont souvent considérés non seulement comme les ensembles de nœuds, mais comme deux graphes distincts. En conséquence, le problème de correspondance

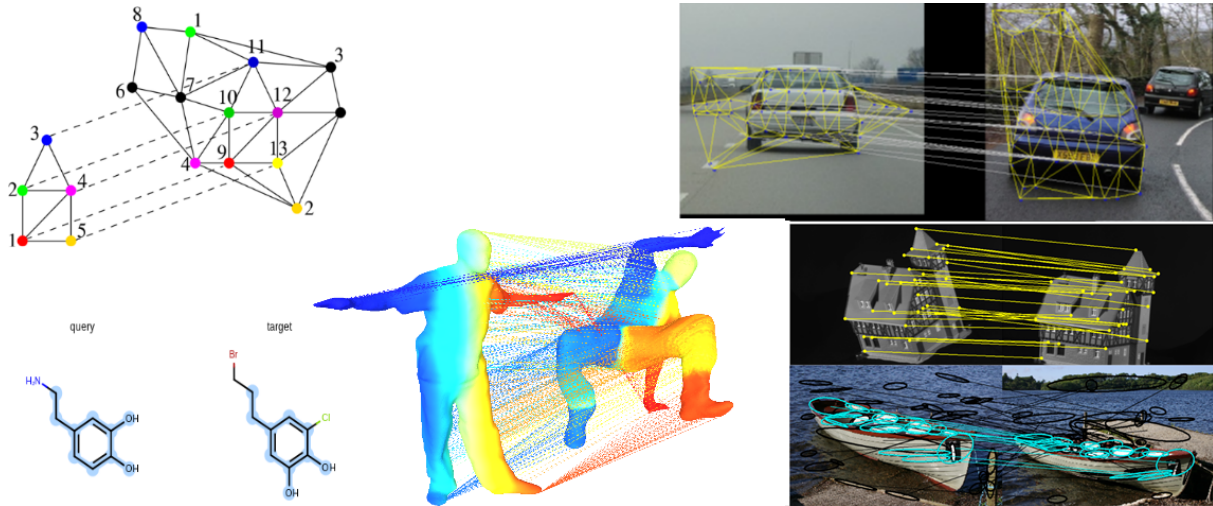


Figure 1 – Graphiques dans divers domaines

graphique est souvent appelé Graph Matching. Dans ce paramètre, les nœuds représentent les nœuds caractéristiques extraits de chaque instance et les arcs représentent les relations entre les nœuds caractéristiques. Donc le problème de Graph Matching consiste à trouver un mappage entre les deux ensembles de nœuds qui préservent le plus possible les relations entre les nœuds. [4]

3 Graph Edit Distance

Les problèmes de Graph Matching (GM) sont séparés en deux catégories principales : l'isomorphisme exact et Graph Matching tolérant aux erreurs.[5]

Le problème GM tolérant aux erreurs est plus important, en particulier dans la tâche de trouver des dissemblances entre des objets inconnus et connus. Il est destiné à trouver la meilleure correspondance entre deux graphes, même s'ils diffèrent dans leur structure et/ou leurs attributs. Un problème bien connu qui appartient à la catégorie des problèmes GM tolérants aux erreurs est le problème de la distance d'édition (Graph Edit Distance, GED). Résoudre ce problème implique de minimiser une mesure de dissemblance qui représente le coût nécessaire pour transformer un graphe en un autre par une série d'opérations d'édition.

Pour savoir Graph Edit Distance, on doit tout d'abord faire la connaissance de ces deux définitions : l'opération d'édition et le coût.

3.1 L'opération d'édition

Dans la forme la plus générale, une opération d'édition est soit une suppression, une insertion ou une substitution (c'est-à-dire une modification d'étiquette). Les opérations d'édition peuvent être appliquées aux nœuds ainsi qu'aux arcs. La distance d'édition de deux graphes, \underline{G} et \underline{G}' , est définie comme la plus petite séquence d'opérations d'édition qui transforme \underline{G} en \underline{G}' . De toute évidence, cette séquence est plus courte, les deux graphes sont plus similaires. Ainsi, la distance d'édition est appropriée pour mesurer la similitude des graphes. La séquence des opérations d'édition qui transforme \underline{G} en \underline{G}' implique un mappage de correction d'erreur des nœuds de \underline{G} aux nœuds de \underline{G}' . Pour le Figure 2, $u_i (i \in [1, 3])$ sont les trois nœuds de \underline{G} , $v_j (j \in [a, b])$ sont les deux nœuds de \underline{G}' , $e_{(i,i)} (i \in [1, 3])$ représente l'arc entre chaque nœud de \underline{G} , et $f_{(j,j)} (j \in [a, b])$

représente l'arc entre chaque nœud de $\underline{G'}$. Donc on va exécuter $u_1 \rightarrow v_a$, $u_3 \rightarrow v_b$, $e_{(1,3)} \rightarrow f_{(a,b)}$ et la suppression de v_2 et $e_{(1,2)}$.

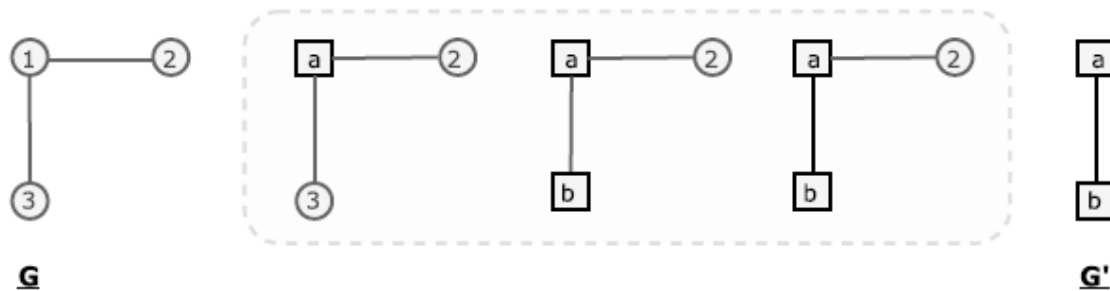


Figure 2 – Une exemple d'une opération d'édition

3.2 Le coût

Dans les applications pratiques, certaines opérations d'édition peuvent avoir plus d'importance que d'autres. Par conséquent, très souvent, les coûts sont affectés aux opérations d'édition individuelles. Généralement, une opération d'édition est plus susceptible de se produire, son coût est plus faible. Une affectation des coûts aux différentes opérations d'édition est souvent appelée fonction de coût.

Compte tenu d'un ensemble d'opérations d'édition associées à leurs coûts, le calcul de la distance d'édition (GED) dans sa forme la plus générale permet de trouver une séquence d'opérations d'édition qui transforme, avec un coût minimal, l'un des graphes donnés dans l'autre. GED^{EnA} (Edges no Attributes) est un sous-problème du problème GED qui traite d'un type particulier de graphes où les arcs ne portent pas d'attributs, ces deux problèmes appartiennent à NP-hard.

4 Les Méthodes de GED

De nombreux algorithmes qui traitent du problème GED et GED^{EnA} peuvent être trouvés. Ils peuvent être divisés en deux catégories : les méthodes exactes et heuristiques.

4.1 Les méthodes exactes

Quelques formulations sont proposées pour résoudre les problèmes GED ou GED^{EnA} si nous voulons utiliser les méthodes exactes.

Deux formules de programmation linéaire binaire (BLP) pour calculer le GED exact entre deux graphes sont proposées dans un article par Lerouge et al.[8], on peut s'appeler F1 et F2. F1 est une formulation directe du problème puisqu'elle consiste à minimiser le somme de coût d'adaptation des sommets et coût de l'alignement des arcs. La formulation F2 a été dérivée de la formulation F1, elle réduit le nombre de variables et le nombre de contraintes.

JH est aussi une formulation générale de programmation linéaire binaire de la distance d'édition du graphe, qui est proposée par D. Justice et A. Hero, mais elle est définie pour les graphes

non-pondérés et non dirigés avec des attributs de sommet, c'est-à-dire la formulation MILP^{JH} est utilisée uniquement pour le problème GED^{EnA}.

Le problème de résoudre un BLP/ILP/MILP est NP-hard, donc explorer l'ensemble de l'arbre de la solution n'est pas une bonne option puisqu'il prendrait un temps exponentiel. Cependant, un solveur qui consiste à mettre en œuvre ce modèle a été développé pour réduire le nombre de solutions explorées et le temps de résolution.

En général, l'approche exacte consiste à résoudre la formulation MILP à l'aide d'un solveur CPLEX sans limite de temps et de ressources, afin d'obtenir la solution optimale, de sorte qu'elle s'appelle CPLEX- α .

4.2 Les méthodes heuristiques

Dans la littérature passée, nous avons pu obtenir beaucoup d'heuristiques.

4.2.1 BP(Bipartite graph matching)

BP est une approche pour la computation efficace de la distance d'édition basée sur l'appariement de graphes bipartis au moyen de l'algorithme de Munkres[WWW1], parfois appelé l'algorithme Hungarian. Cet algorithme s'exécute en temps polynomial, mais ne fournit que des résultats de distance d'édition sous-optimaux. La raison de sa sous-optimalité est que les opérations d'arc impliquées sont prises en compte mais de manière indépendante des autres affectations lors de la recherche de l'affectation optimale des nœuds.[10] C'est-à-dire BP ne considère que la structure locale autour des sommets, plutôt que la structure globale.

4.2.2 Beam-search

Cette méthode[9] est basée sur la recherche de faisceau. Au lieu d'étendre tous les nœuds successeurs dans l'arbre de recherche, seul un nombre fixe de nœuds à traiter est conservé dans l'ensemble OPEN à tout moment, par conséquent, nous définissons généralement α comme un paramètre pour représenter un nombre fixe de nœuds. Cela veut dire que l'espace de recherche n'est pas complètement exploré, mais seulement les nœuds associés aux plus prometteuses des correspondances partielles sont développés. Cette méthode calcule la distance d'édition d'une manière plus rapide mais elle ne renvoie généralement pas le chemin d'édition optimal, mais seulement un chemin sous-optimal.

4.2.3 SBPBeam

Il combine deux heuristiques : l'appariement des graphes bipartis (BP) et les heuristiques de recherche des faisceaux (Beam-search).[7] L'heuristique SBPBeam utilise une approche de recherche de faisceau comme une recherche locale pour améliorer la solution initiale obtenue par l'heuristique BP. Chaque niveau de l'arbre de recherche correspond à tous les échanges possibles des composants (sommets ou arêtes) de deux opérations d'édition de la solution initiale. À chaque nœud, le coût du chemin d'édition est calculé et la meilleure solution trouvée jusqu'à présent est mise à jour. Un nombre sélectionné de nœuds sont conservés à chaque niveau de l'arbre de recherche, en fonction de la valeur de la taille du faisceau. De même, nous définissons α comme un paramètre pour représenter un nombre fixe de nœuds.

4.2.4 IPFP/GNCCP

Les opérations d'édition sur les nœuds et les arcs ne sont pas traitées simultanément en utilisant la méthode GED bipartite, ce qui limite la précision de l'approximation. Pour surmonter cette limitation, S. Bougleux et al.[2] ont proposé d'étendre le modèle d'assignation linéaire à un modèle quadratique. Ceci est réalisé grâce à la définition d'une famille de chemins d'édition induite par les affectations entre les nœuds. Il a été prouvé que GED, restreint aux chemins de cette famille, est équivalent à un problème d'affectation quadratique. Comme ce problème de GED est NP-difficile, ils proposent de calculer une solution approximative en adaptant deux algorithmes : la méthode IPFP (Integer Projected Fixed Point) et la méthode GNCCP (Graduation Non Convexity et Concavity Procedure).[3] Les deux utilisent la formulation QAP du problème GED pour calculer une solution initiale, puis tentent de l'améliorer de manière itérative en utilisant des transformations mathématiques et des méthodes de projection. Les expériences montrent que l'approche proposée est généralement capable d'atteindre une approximation plus précise du GED exact que le GED bipartite, avec un coût de calcul qui est encore abordable pour des graphiques de tailles non triviales. En utilisant ces deux approches, nous devons généralement définir le nombre maximal d'itérations pour IPFP, et pour GNCCP, il est important de définir la quantité à déduire de la variable ζ à chaque itération, où ζ est la variable qui contrôle la concavité et la convexité de la fonction objective du modèle QAP.

5 Notre proposition : LocBra

Nous choisissons l'heuristique Local Branching (LocBra[6]) pour traiter le problème GED^{EnA} . LocBra a été initialement introduit par Fischetti et Lodi, en tant que méta heuristique général basé sur les formulations de MILP. En général, une heuristique locale de branchement est un algorithme de recherche local qui améliore une solution initiale en explorant une série de voisinages définis via la solution de formulations MILP. Le but est de fournir une heuristique de branchement locale efficace et dédiée pour le problème GED^{EnA} , qui surpasse fortement les heuristiques de la littérature disponible.

Pour le choix de MILP, nous choisissons la formulation de $MILP^{JH}$ (c'est-à-dire nous choisissons JH comme le modèle mathématique) qui est la formulation de base utilisée dans l'heuristique de ramification locale pour résoudre le problème de GED^{EnA} . Cette formulation s'est montrée la plus efficace pour le problème GED^{EnA} parmi les autres formulations trouvées dans la littérature.

Semblablement, nous pouvons utiliser le solveur CPLEX pour résoudre cette formulation $MILP^{JH}$, dans ce cas d'une instance du problème, ce solveur CPLEX explore l'arbre des solutions avec l'algorithme de branchement et de liaison, et trouve la meilleure solution réalisable, en termes d'optimisation de la fonction objective.

Tout d'abord, LocBra commence par une solution initiale x^0 , et définit son voisinage k -opt, on peut le signaler $N(x^0, k)$. Le voisinage défini contient les solutions qui se trouvent à une distance ne dépassant pas k à partir de x^0 .

$$\Delta(x, x^0) \leq k \quad (1)$$

Maintenant, LocBra commence à rechercher la meilleure solution dans $N(x^0, k)$, correspondant au nœud 2 de la Figure 3. Si une nouvelle solution x^1 est trouvée, la contrainte est remplacée par $\Delta(x, x^0) \geq k + 1$ et la branche droite émanant du nœud 1 est explorée. Cela garantit qu'un espace de solution déjà exploré ne sera plus visité. Ensuite, une branche gauche est créée mais en utilisant la solution x^1 pour la contrainte $\Delta(x, x^1) \leq k$, correspondant au nœud 4 de la Figure 3.

Donc le processus est répété jusqu'à ce qu'un critère d'arrêt soit atteint, par exemple, une limite de temps totale (*total time limit*) ou une limite de temps de nœud (*node time limit*).

Parfois $\Delta(x, x^i) \leq k$ ne conduit pas à une solution réalisable dans le délai limité, par exemple le nœud 6 de la Figure 3. En ce moment, une étape d'intensification complémentaire est appliquée, en remplaçant la dernière contrainte par $\Delta(x, x^i) \leq k/2$. Si aucune solution réalisable n'est trouvée encore (nœud 7 sur la Figure 3-a), une étape de diversification est appliquée pour sauter à un autre point de l'espace de la solution. Cette procédure de diversification peut aider l'algorithme à échapper aux optimaux locaux. La Figure 3-b montre l'évolution de la recherche de solution et des voisinages.

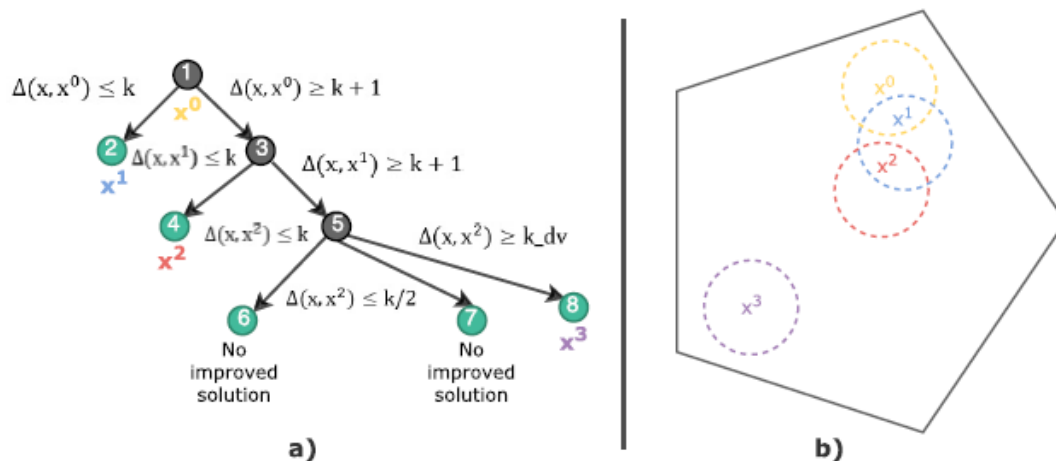


Figure 3 – Diagramme d'algorithme

En plus de k , la limite de temps totale et la limite de temps de nœud, LocBra nécessite les paramètres suivants :

- ✓ k_{dv} : les changements minimum de variables binaires pour garantir que la prochaine solution soit trouvée dans l'actuelle.
- ✓ l_{max} : pour éviter de passer beaucoup de temps à chercher dans une région où aucune solution d'amélioration n'est trouvée.
- ✓ dv_{max} : le nombre maximum d'étapes de diversification autorisée pendant l'exécution de LocBra.
- ✓ dv_{cons_max} : un critère d'arrêt, lorsque des étapes de diversification consécutive ont renvoyé des solutions avec la même valeur de la fonction objective, alors l'heuristique s'arrête.

6 L'Expérimentation et les comparaisons

6.1 Choix de base de données

Comme indiqué précédemment, Graph Matching a de nombreuses applications dans la reconnaissance de formes. Cela a conduit au fil des années à la création de bases de données pertinentes d'instances liées à des applications particulières. Ils servent également de base reconnue pour comparer les algorithmes d'appariement. Les bases de données MUTA et PAH

[1] sont choisies après avoir passé en revue les jeux de données publics, qui sont généralement utilisés pour évaluer les méthodes GED. Ces deux bases de données contiennent des graphiques représentant des molécules chimiques. MUTA est divisé en 8 sous-ensembles, les 7 premiers sous-ensembles contiennent 10 graphes du même nombre de sommets. Le dernier sous-ensemble comporte 10 graphes de tailles mixtes. Cette base de données est intéressante car elle a de grands graphes, et ils sont connus pour être difficiles à faire correspondre les algorithmes. La base de données PAH comprend 94 graphes de tailles diverses et de petite taille. Chaque paire de graphes est considérée comme une instance du problème GED^{EnA}. Par conséquent, la base de données MUTA contient un total de 800 instances ($8 * 100$) et PAH contient un total de 8836 ($94 * 94$) instances.

Deux expérimentations sont faites pour chaque base de données. La première consiste à évaluer la qualité des solutions obtenues par LocBra. La seconde consiste à étudier l'efficacité de LocBra par rapport aux heuristiques de la littérature.

6.2 Comparaison de LocBra avec l'approche exacte

Pour la base de données PAH, les solutions optimales sont calculées pour toutes les 8836 instances par CPLEX- α , et LocBra a trouvé les solutions optimales pour 8702 instances. Donc LocBra offre un bon résultat puisque la déviation moyenne est petite. D'ailleurs, CPLEX- α est en moyenne plus rapide que LocBra mais dans le pire des cas, CPLEX- α devient grand en calcul (le temps maximal arrive jusqu'à 278.20s), alors que l'heuristique reste au maximum inférieure à 13s.

Pour la base de données MUTA, il y a deux catégories : les instances simples et les instances difficiles. Sur toutes les instances simples, la déviation moyenne de LocBra est égale zéro (Sauf une seule instance), pour les instances difficiles, la déviation moyenne est toujours inférieure à 1%. Quand on utilise un LocBra avec 4 threads, la déviation moyenne est parfois inférieure à 0%, c'est-à-dire que CPLEX- α n'a pas pu trouver la solution optimale mais LocBra a fourni une meilleure solution dans certains cas. Compte tenu du temps moyen, il a beaucoup augmenté pour CPLEX- α et atteint plus de 3,000 secondes, alors que LocBra a un maximum de 751s, ainsi que de déviations moyennes très faibles obtenues.

Le résumé pour la comparaison de LocBra avec l'approche exacte : dans le cas d'instances simples, CPLEX- α est très efficace dans la résolution de MILP^{JH} et l'obtention de solutions optimales. Il a également été prouvé que LocBra a également l'avantage d'obtenir des solutions optimales approchées. Un avantage pour LocBra est qu'il est plus rapide dans le pire des cas en raison de la limite de temps imposée. En ce qui concerne les instances difficiles, où CPLEX- α n'est pas capable de calculer toutes les solutions optimales, mais LocBra fonctionne mieux que CPLEX et fournit de meilleures solutions pour certaines instances dans le temps limité.

6.3 Comparaison de LocBra avec l'heuristique de la littérature

L'heuristique LocBra est testée par rapport aux heuristiques suivantes : CPLEX-t, CPLEX-LocBra-t, BeamSearch- α , SBPBeam- α , IPFP-it et GNCCP-d. Les choix des paramètres pour chaque heuristiques soit suivent les expériences préliminaires soit sont tirés de l'article qui a présenté à l'origine les méthodes.

Pour la base de données PAH, bien que la déviation moyenne de LocBra est supérieur à celle de CPLEX-t, mais LocBra fournit les solutions approchées aux meilleurs dans le pire des cas. En termes de temps, l'heuristique basée sur BeamSearch- α fonctionne très rapidement (le temps moyen < 1s), alors que l'heuristique proposée est la plus lente avec le temps moyen 3.03s.

Pour la base de données MUTA/PAH, sur les instances simples, les heuristiques fondées sur la méthode MILP ont donné les meilleures solutions pour presque toutes les instances et ils surpassent fortement les deux heuristiques à base de BeamSearch. Sur les instances difficiles, LocBra proposé fonctionne le mieux et a la plus petite déviation moyenne qui est inférieure à 0.6%.

Le résumé pour la comparaison de LocBra avec l'heuristique de la littérature : LocBra proposé est plus efficace que le solveur et sa mise en œuvre générique de branchement local, et il améliore l'heuristique de la littérature et fournit des solutions approchées.

4

Analyse et conception

Étant donné que le projet est optimisé pour installer des algorithmes heuristiques, le diagramme de classe prévisionnel ci-dessous **Figure 1** permet de présenter les principaux composants du Model et la couche DAO ainsi que leurs relations, les parties en jaune foncé sont mes tâches principales à améliorer et à ajouter.

Dans la package de « Model », il y a huit classes principales « Methode », « Dataset », « Subset », « Test », « Execution », « Parametre », « TestMethode » et « TestMethodeParametre » correspondant aux huit classes contenues dans la couche DAO pour les opérations de la base de données. Parmi eux, la classe « Test » et la classe « Execution » associent les classes « Methode », « Dataset » et « Subset » parce qu'il faut obtenir et utiliser ces données avant de l'ajout d'un test ou d'une exécution, donc « Methode », « Dataset » et « Subset » sont des attributs de la classe « Test » et de la classe « Execution ».

D'ailleurs, compte tenu de l'utilisation de la méthode est divisée en deux cas : « Execte » et « Heuristique », ma proposition est de créer un nouvel attribut dans la classe de « Methode » qui s'appelle « type ». La grande différence entre les deux types est que les méthodes heuristiques nécessitent des paramètres spécifiques afin d'ajouter des contraintes à la méthode pour obtenir un effet heuristique. Comme il existe deux types de méthodes, l'algorithme exact n'a pas de paramètres, c'est pourquoi nous ne pouvons pas ajouter directement des paramètres à la classe de méthode en tant qu'attributs. Donc j'ai créé une nouvelle classe « Parametre » pour indiquer les informations de chaque paramètre heuristique.

Dans le même temps, compte tenu des exigences de l'utilisateur, le nom du paramètre et le type de paramètre de chaque méthode sont fixes, mais la valeur du paramètre peut être entrée de manière aléatoire chaque fois que l'utilisateur ajoute le test. C'est-à-dire que nous ne pouvons pas ajouter directement la valeur d'un paramètre à la classe de « Parametre » en tant qu'attribut. Donc j'ai créé une nouvelle classe « TestMethode » pour stocker une instance qui contient d'une méthode et d'un test, grâce à cette classe (ou table), nous pouvons connaître les informations de correspondance entre toutes les méthodes ajoutées au test et son propre test. De même, nous avons également besoin d'une classe « TestMethodeParametre » qui stocke ces informations : Pour une méthode ajoutée au test, un des paramètres qu'elle contient et la valeur de ce paramètre. L'avantage de cette conception est que nous pouvons trouver n'importe quelle valeur de n'importe quel paramètre de n'importe quelle méthode de n'importe quel type de test. Ce couplage fort assure également la synchronisation des données dans la base de données.

Pour la référence de base de données, un ensemble de données (Dataset) contient un ou plusieurs

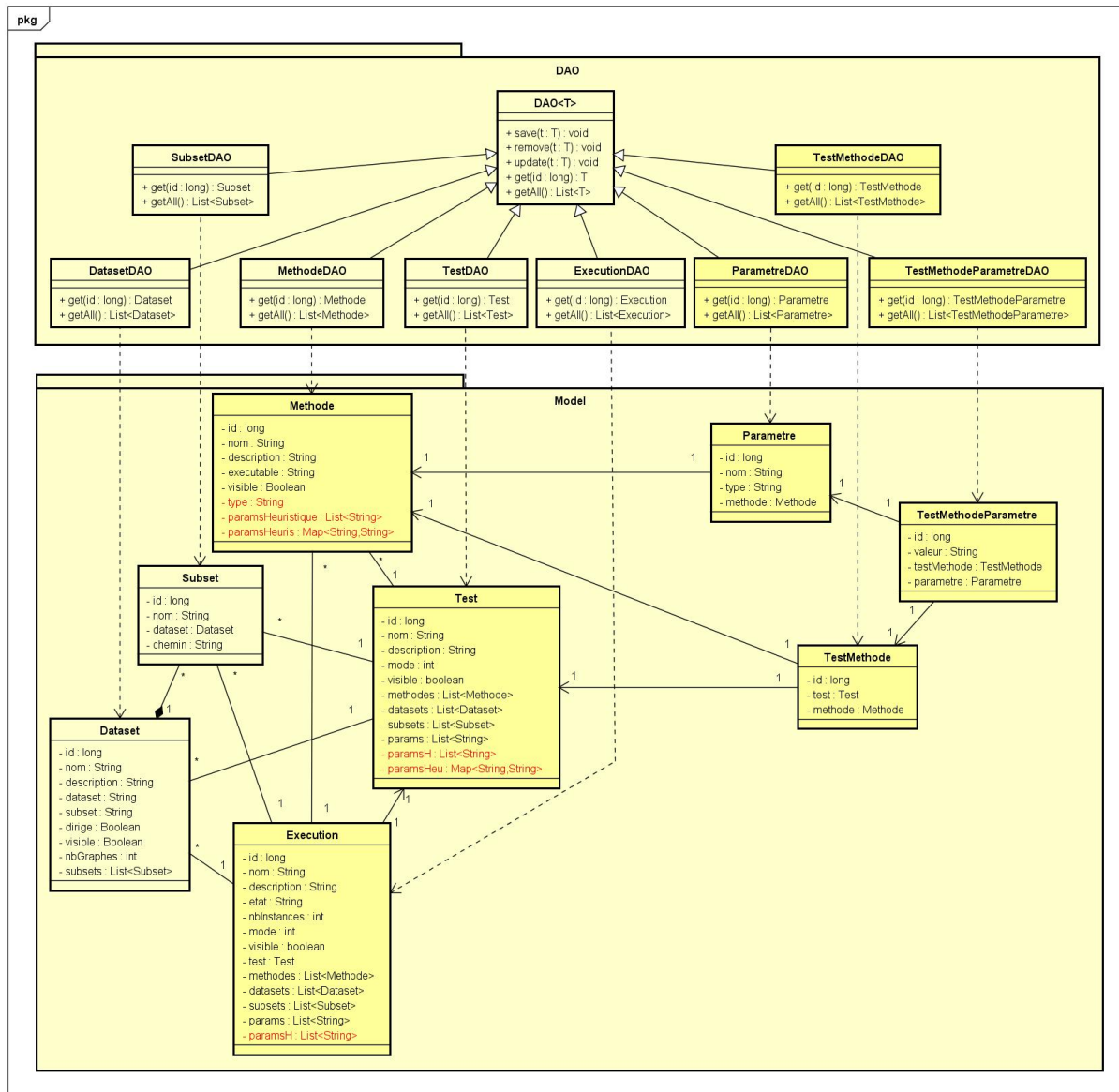


Figure 1 – Le diagramme de classe DAO & Model

sous-ensembles (Subset), de sorte que la relation entre eux est une combinaison.

Ces classes incluent des opérations simples telles que `get()` et `set()` pour obtenir les valeurs de l'attribut ou affecter des valeurs aux attributs.

En outre, la classe « Test » contient certaines fonctions nécessaires que je dois modifier ou améliorer pour ajouter un test, par exemple, ajouter des paramètres heuristiques ou obtenir les paramètres heuristiques que les utilisateurs ont saisis :

- ✓ « `addParamHeu(nom : String, valeur : String)` » : Ajouter un paramètre à la liste pour les méthodes heuristiques
- ✓ « `fichierParam(id : long)` » : Créer le fichier contenant les paramètres
- ✓ « `getParametresHeuristique()` » : Renvoyer les paramètres heuristiques correspondant les méthodes heuristiques sous la forme d'un HashMap
- ✓ « `getEachParams(id : long)` » : Retourner la liste des paramètres heuristiques sous forme « nom = valeur » pour chaque méthode quand on va exécute un test qui contient quelques méthodes heuristiques

La classe « Execution » contient également certaines fonctions nécessaires que je dois modifier ou améliorer pour obtenir les résultats de l'exécution, par exemple, calculer le nombre des instances, lire des fichiers de sortie pour calculer la déviation, extraire l'appareillement des sommets et des arcs entre deux graphes pour une méthode, lire les informations ligne par ligne de chaque graphe sur le fichier « .dat » pour visualiser les graphes sur Web et faire l'exécution et mettre à jour l'état de l'Exécution, etc.

La relation entre eux et certaines fonctions importantes sont illustrées dans la [Figure 2](#). Et pour plus d'informations avec les détails et compléments, veuillez-vous référer à l'[Annexe D](#) Analyse et cahier du développeur.

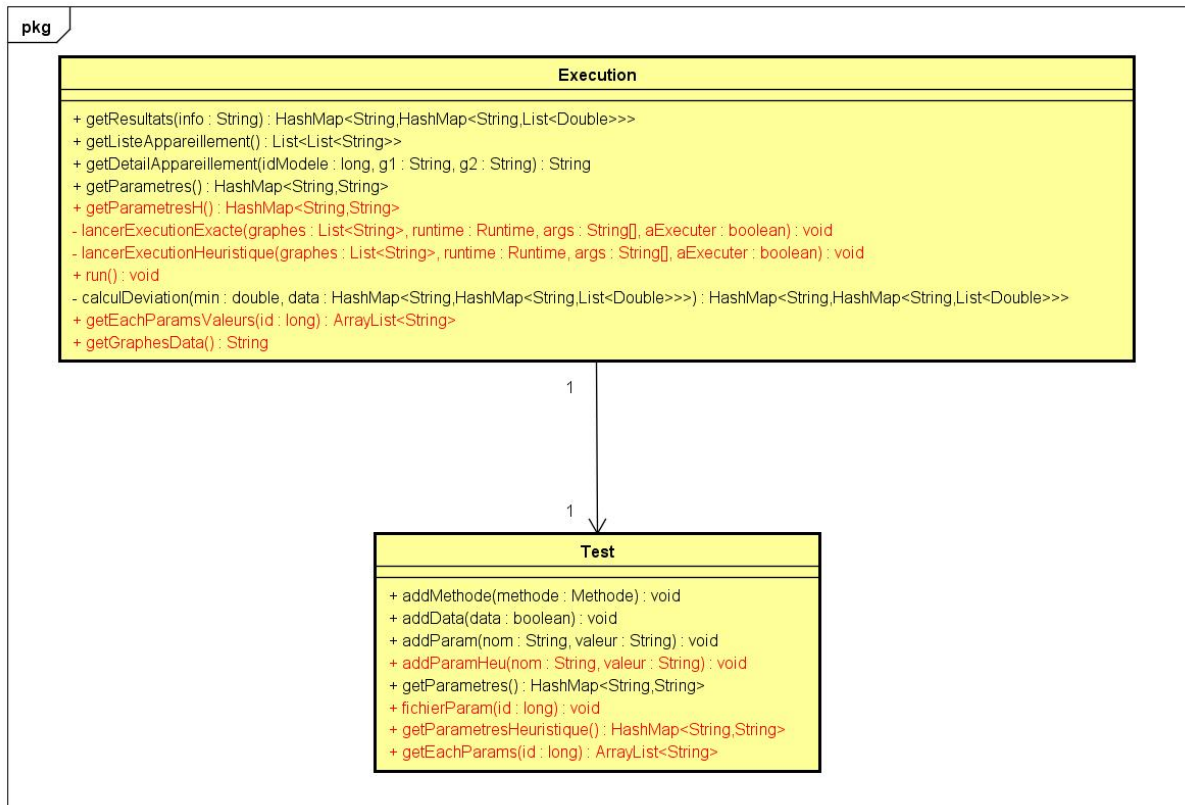


Figure 2 – La relation entre la classe « Test » et la classe « Execution »

La couche de servlet, équivalente au contrôleur en mode MVC, est principalement responsable de la transmission des données front-end et back-end. Les trois packages y compris « Servlets.Methodes », « Servlets.Datasets » et « Servlets.Tests » sont principalement pour le contrôle des méthodes et des collections graphiques. La [Figure 3](#) ci-dessous montre la structure prévisionnelle d'un module de Servlet pour visualiser et modifier des données et générer du contenu Web dynamique, les parties en jaune foncé sont mes tâches principales à améliorer. Il n'y a pas beaucoup d'interaction dans cette partie, en utilisant essentiellement doGet() et doPost(), mais chaque Servlet dépend des couches DAO et Model montrées ci-dessus. Parmi eux, « FicheTestServlet » a été modifié le plus pendant cette année, lequel récupère et retourne des informations sur un test selon le choix de l'utilisateur.

Pour les deux autres packages y compris « Servlets.Execution » et « Servlets.Resultat », « ExecExecution » gère l'exécution des instances de l'exécution échouée, « InfosExecutionServlet » gère l'affichage des informations sur l'exécution, « NewExecutionServlet » gère la création et l'exécution, « FicheResultatServlet » récupère et retourne des informations sur le résultat d'une exécution selon le choix de l'utilisateur, « DetailAppareillement » renvoie les informations sur

les appareilllements des nœuds et des arcs sous forme d'un tableau HTML, et également renvoie les informations sur les appareilllements des nœuds et des arcs sous forme d'une correspondance de visualisation entre deux graphes.

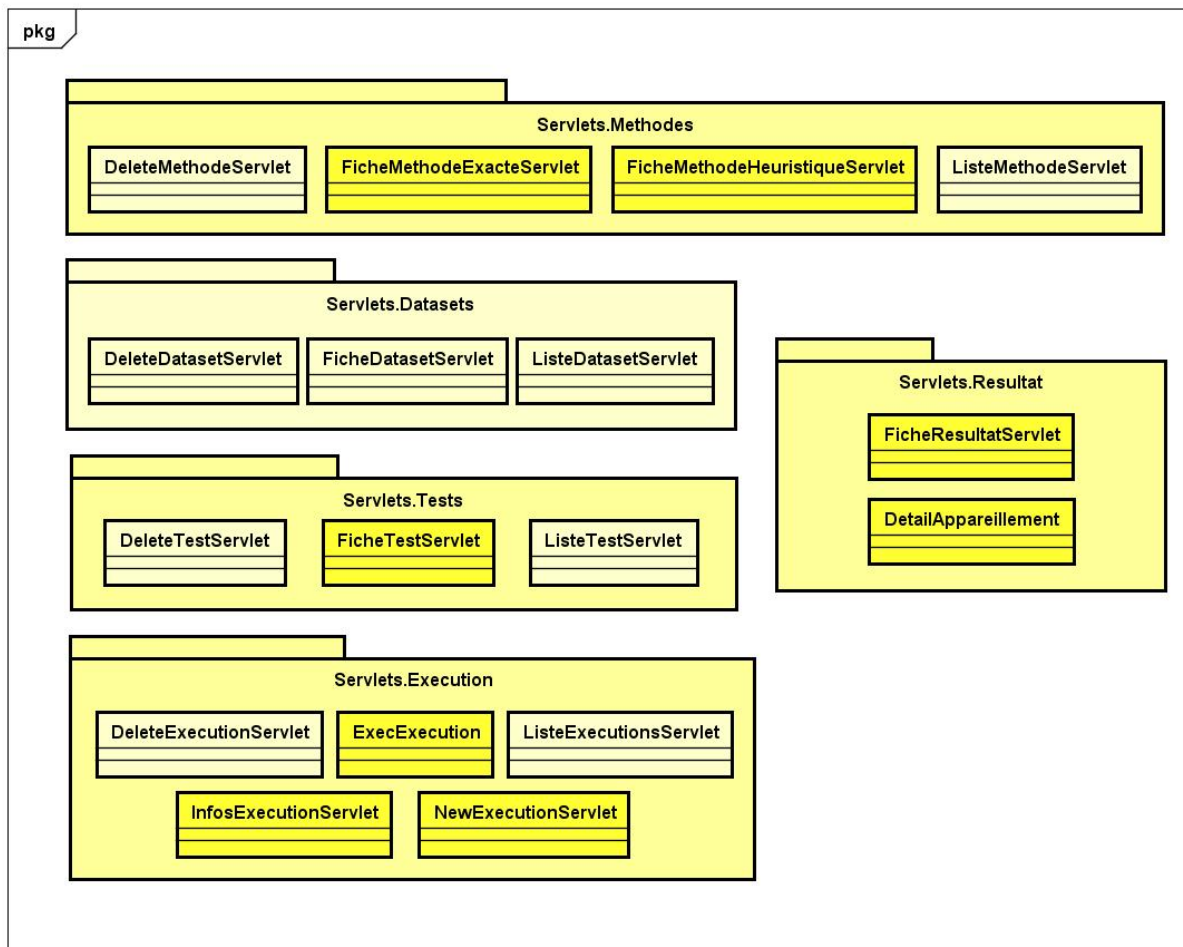


Figure 3 – Le diagramme de classe Servlet

La visualisation des deux correspondances graphiques est une partie difficile du projet. Ma proposition est de mettre en place trois toiles avec deux petites tailles de toile et une grande taille de toile. Les deux petites toiles permettant d'afficher les deux graphes à comparer, et la grande toile est pour présenter le résultat de appareillement. Le contenu en texte brut de graphe est obtenu en lisant de la collection choisie par utilisateur dans l'exécution courante, ce processus nécessite à appeler les fonctions de back-end, ensuite, à travers une série de traitement de données, qui sont réalisée par la fonction de front-end, on va compléter la visualisation de l'image. Ce processus de traitement de front-end est très compliqué : lecture de la table, découpage du texte, lecture de l'image sélectionnée, prétraitement des données, appel du plugin « echarts » pour dessiner l'image, affichage de la connexion des deux points correspondants et amélioration d'interaction de l'utilisateur pour l'importation ultérieure de plusieurs collection graphique. Pour plus d'informations, veuillez-vous référer à l'[Annexe D](#) Analyse et cahier du développeur.

5

Mise en oeuvre

1 Les outils et librairies utilisées

J'ai choisi Apache Tomcat comme Serveur d'applications Web, qui est un petit serveur d'applications léger avec une technologie de pointe, une performance stable. Il est couramment utilisé dans les systèmes de petite et moyenne taille et dans les systèmes qui n'ont pas trop de utilisateurs d'accès concurrents. Il est le premier choix pour le développement et le débogage des programmes JSP. Lors de l'exécution, il occupe peu de ressources système, il a une bonne évolutivité et il prend en charge les fonctions couramment utilisées dans le développement Web.

Pour l'utilisation des librairies, j'ai utilisé deux librairies graphiques visuelles, Google Charts et ECharts, qui peuvent être utilisées comme outils de conception graphique. Le projet de l'année dernière a pris Google Charts, mais pour les nouvelles fonctionnalités qui doivent être ajoutées cette année, ECharts, une bibliothèque de graphiques Javascript pure, qui est compatible avec la plupart des navigateurs actuels (IE8/9/10/11, Chrome, Firefox, Safari, etc.), peut être recalculé par glisser-déposer et peut implémenter des fonctions d'analyse de données multidimensionnelles. Il peut fournir des graphiques de visualisation de données intuitifs, vifs, interactifs et hautement personnalisables. Par conséquent, j'ai choisi ECharts pour la réalisation de la fonction de visualisation de correspondance graphique. Veuillez vous référer à l'[Annexe D](#) Analyse et cahier du développeur pour une analyse détaillée.

En outre, on utilise également la JSTL (JSP Standard Tag Library), qui est un ensemble de balises JSP qui encapsulent la fonctionnalité de base commune des applications JSP. Il a les avantages suivants :

1. Fournir une interface cohérente entre les serveurs d'applications pour optimiser la migration des applications Web entre les serveurs d'applications.
2. Simplifier le développement des applications JSP et WEB.

2 Limites et risques

Les principales conditions et restrictions pour l'installation de ce projet sont que la structure de la base de données doit être claire et rigoureuse, avec une logique forte et un développement durable. Actuellement, la structure de base de données du système est fortement couplée.

Du point de vue de l'analyse des risques prévisible, le logiciel peut entraîner dans certains cas des risques techniques dus à des spécifications et à des technologies en cours de conception, de mise en œuvre et de maintenance, il est donc nécessaire de maîtriser strictement la structure existante. Dans le même temps, conformément aux exigences réelles du projet, on doit sélectionner des technologies adaptées et matures pour la implémentation et l'amélioration du projet. Manuel d'installation et manuel d'utilisation veuillez vous référer à l'[Annexe F](#) et l'[Annexe G](#).

Puisque ce système est utilisé comme un système de contrôle d'algorithme pour les chercheurs et n'est pas utilisé pour les ventes, il n'y a pas de risque commercial.

Parallèlement, la gestion de projets des logiciels assure non seulement un contrôle efficace conformément aux processus de gestion établis, mais aussi une gestion standardisée des documents à chaque étape afin de garantir l'intégrité et la standardisation des documents, de cette façon on peut réduire les risques de développement logiciel.

3 Design pattern

Le développement du système suit le modèle de conception MVC (modèle vue contrôleur). Dans l'architecture J2EE, la couche de présentation « Vue » fait référence à la couche du navigateur et est utilisée pour afficher graphiquement les résultats de la requête. La couche du « Contrôleur » fait référence à la couche du serveur Web. La couche de données « Modèle » fait référence à l'implémentation de la logique applicative et à la persistance des données. Dans le programme JSP traditionnel précédent, les pages Web et l'accès aux données étaient mélangés, mais dans MVC, la couche de présentation « Vue » et la couche de données « Modèle » étaient séparées forcément et le « Contrôleur » peut être utilisé pour connecter différents modèles et vues afin de répondre aux besoins de l'utilisateur.

Selon le concept de ce design pattern, nous avons mis au point un modèle de développement « JSP + Servlet + JavaBean », c'est-à-dire :

- JSP est utilisé pour afficher des vues et représenter des pages, avec des scripts « JavaScript » et des styles « CSS » comme aide ;
- Servlet est le contrôleur, utilisé pour contrôler le flux du programme et appeler des services de traitement ;
- JavaBean encapsule la logique et le métier, utilisé pour enregistrer les entités de données, correspondant généralement aux tables de la base de données.

Son défaut est : Il n'y a pas de cadre de développement unifié, ce qui entraîne un long cycle de développement. Cependant, les frameworks Spring et Struts augmentent souvent la complexité du système, perturbant les habitudes d'écriture des pages web du programmeur. Par conséquent, pour le développement simple du système, le mode de conception ci-dessus peut être adopté et plus pratique.

En outre, j'ai également utilisé la couche DAO, DAO (Data Access Object) est un type de design pattern d'objet d'opération de base de données, il est dans la couche de « Model » de MVC dans le développement Java EE pour accéder à la couche de base de données. L'avantage de l'utilisation de la couche DAO est que le niveau du structure du système est clair, la division du travail est claire, elle n'effectue que l'accès aux données et elle est pratique pour les tests unitaires. En même temps, si le langage ou le framework de la base de données change, nous avons seulement besoin de modifier le code pertinent dans la couche DAO, réduisant ainsi l'impact sur les autres couches de gestion.

4 Framework –Hibernate

Puisque ce projet est optimisé en tant que projet hérité de l'année dernière, le cadre de développement du programme est continuellement respecté. C'est-à-dire, afin de compléter toutes les nouvelles fonctionnalités demandées cette année, je continue d'utiliser le framework « Hibernate » dans la structure de code d'origine.

Pendant l'utilisation, j'ai découvert les avantages d'Hibernate :

Pour faire un mapping une base de données à une classe Java, il suffit de manipuler la classe Java pour implémenter l'opération de base de données. Donc je n'ai pas besoin d'écrire des instructions SQL pour ajouter, modifier, supprimer et rechercher dans la base de données.

Hibernate peut mettre la base de données fréquemment consultée dans le cache, réduisant l'accès à la base de données.

Léger, flexible, prenant en charge une variété de bases de données relationnelles, allant de la relation un-à-un aux relations complexes plusieurs-à-plusieurs.

Vous trouverez des instructions détaillées sur le travail de configuration et le développement de « Hibernate » dans l'[Annexe D](#) Analyse et cahier du développeur.

5 Les outils de gestion du projet

La gestion de versions et la gestion des codes sources consistent à maintenir l'ensemble des versions d'un ou plusieurs fichiers. Cette technique « Github » a été très utilisée pendant le déroulement de développement. Son utilité pour mon projet a permis de sauvegarder mes codes et documentation de configuration dans un serveur([Figure 1](#)).

| | | | | | | | |
|-----------------|---|--------------------------------------|--|------------------------------------|--------------|---------------|-------------------|
| 24 commits | | 1 branch | | 0 releases | | 1 contributor | |
| Branch: master | | New pull request | | Create new file | Upload files | Find file | Clone or download |
| littlecherisher | | Modify some erreurs found by JavaDoc | | Latest commit 6b86193 13 hours ago | | | |
| .settings | settings | | | 14 days ago | | | |
| WebContent | complete all the necessary comments and modify some pages | | | 13 hours ago | | | |
| build/classes | Modify some erreurs found by JavaDoc | | | 13 hours ago | | | |
| src | Modify some erreurs found by JavaDoc | | | 13 hours ago | | | |
| .classpath | update configuration files | | | 14 days ago | | | |
| .project | Finish reading (heuristic) parameters in database | | | 2 months ago | | | |
| ParseXML.exe | add ParseXML.exe | | | 14 days ago | | | |
| cplex1260.dll | update configuration files | | | 14 days ago | | | |

Figure 1 – Versionning avec Github

6 Mise en place des tests

En général, les tests unitaires peuvent être utilisés pour vérifier qu'une petite fonction bien définie du code testé est correcte. Dans les grands projets de développement, il est d'abord nécessaire de s'assurer que chaque unité fonctionne avec succès afin de s'assurer que la fonction

prévue est atteinte. Les tests unitaires peuvent détecter les bogues le plus tôt possible dans les premières phases du développement, ce qui permet aux développeur d'économiser plus de temps et de réduire les risques logiciels.

Par conséquent, dans ce projet, après avoir terminé la conception et le développement de chaque classe de JavaBeans au niveau du « Model », j'ai d'abord effectué des tests unitaires pour s'assurer que les fonctions de back-end puissent être exécutées avec succès et garantir l'extraction des données à front-end.

J'ai choisi « Junit » comme l'outil de test unitaire, ce qui est un framework de test de Java, et il permet de simplifier le travail de test unitaire et l'efficacité de développement s'élève.

Puis, des tests d'intégrations vérifient l'interaction entre les modules pour assurer le transfert de données entre back-end et front-end.

Ensuite, des tests fonctionnels sont nécessaires pour vérifier les fonctions du système. J'ai testé les éléments un à un en fonction des cas de test fonctionnels pour vérifier si le système répond aux fonctions demandées par l'utilisateur.

Veuillez vous référer à l'[Annexe H](#) pour les dossiers de test.

7 Qualité du code

- Haute lisibilité, le nom de la fonction et le nom du paramètre peuvent être facilement compris.
- Selon la spécification de construction « JavaDoc », les commentaires sont nommés strictement et de manière standard.
- Lire le code de l'année dernière pour réduire autant que possible le taux de répétition du code.
- La structure du code correspond strictement au diagramme UML.
- Utiliser les frameworks « Hibernate » et les plug-ins pour réduire les SQL inutiles et le code de faible qualité.
- Il est portable (appliqué à plusieurs systèmes d'exploitation et IDE), maintenable (documentations complètes, et il a des interfaces/entrée pour les besoins potentiels futurs), une grande flexibilité.

8 Résultat et performance

Tous les résultats du système et les fonctions réalisées sont disponibles en [Annexe G](#) Doc d'utilisation.

Mesurée du point de vue des utilisateurs, ce système :

- Il a répondu à des besoins des clients.
- Son opération est simple, facile à utiliser, et la page est intuitive.
- Convient pour plusieurs navigateurs.
- Ecrire plus de code de traitement de données de page dans « JavaScript » au lieu de back-end « Java », ce qui réduit l'occupation des serveurs et améliore la vitesse de navigation.

6

Bilan et conclusion

1 Point sur Fait et Reste à faire/retards S9

Plusieurs choses ont été faites durant ce premier semestre. Cette partie m'a permis d'acquérir et de posséder les connaissances dans le domaine de la recherche, la lecture des documents scientifiques et de pouvoir rapporter correctement ma recherche à mes professeurs hebdomadairement. Dans la section de recherche théorique, je me familiarise d'abord avec l'article qui m'a été confié par mon tuteur et j'apprends les concepts de correspondance graphique et de distance d'édition graphique. Lors du calcul de la distance d'édition des graphiques, je me suis concentré sur la recherche et la réflexion sur les avantages et les inconvénients des méthodes exactes et des méthodes heuristiques, et j'ai analysé les résultats expérimentaux. En même temps, j'ai étudié cinq algorithmes précis qui seront ajoutés aux systèmes existants dans le futur. Pendant cette période, j'ai pu aussi faire l'étude de l'application existante. J'ai eu le temps de me familiariser avec ce système, afin de comprendre comment elle était faite, ou trouver les désavantages existantes du système, et proposer mes propres idées. Ensuite, j'ai déjà la possibilité d'avoir une idée précise de la structure de ce système et de faire l'analyse et la modélisation pour les fonctionnalités ajoutées afin de faciliter l'optimisation de ma deuxième partie.

Il y a quelques retards au début du projet. Tout d'abord, la correspondance graphique et l'heuristique sont nouvelles pour moi et n'ont pas été étudiées, donc la lecture des documents prend plus de temps que prévu. Bien sûr, les barrières linguistiques sont aussi l'une des raisons. La seconde est une analyse des besoins du projet, au départ, je n'avais pas une idée claire des livrables. Je remercie mes tuteurs, leurs explications patients et conseils utiles, qui m'ont permis d'en savoir plus sur le contexte et les objectifs du projet. Pour plus d'informations, veuillez-vous référer à l'[Annexe E](#) Gestion de projet.

Les principales tâches de ce semestre ont été essentiellement achevées. Mais les trois langages de développement de JavaScript / JSP / CSS me sont encore inconnus à l'heure actuelle, la familiarité et l'application de ces trois langues seront la Reste à faire de ce semestre et la première tâche du prochain semestre.

De manière générale, cette première partie du projet m'a permis de découvrir les importances d'analyse et de recherche d'un projet et de réfléchir comment mettre en pratique les connaissances acquises. J'ai également eu la possibilité de distribuer les tâches et le temps par soi-même, ce qui m'a permis d'améliorer mon sens de la gestion du projet.

2 Plan de développement S10

Les principales tâches et les plans pour le prochain semestre sont de mettre en œuvre les principales fonctionnalités à ajouter et à optimiser dans l'analyse ci-dessus, et de tester la qualité du code. Pendant cette période, selon la méthode Agile de gestion de projet, je continuerai à consulter le MOA pour répondre à leurs besoins, mais aussi pour faciliter la prochaine étape du plan plus clairement. Ce processus va durer environ deux mois. Enfin, je vais générer une documentation Javadoc basée sur le code du projet, écrire des manuels d'utilisation et préparer des rapports et la maintenance. Pour le planning de développement plus détaillé, c'est possible de voir le diagramme de Gantt et les Sprints dans l'[Section 5](#) (Annexe E) de Gestion de projet.

3 Point sur Fait et Reste à faire/retards S10

Pour le travail de développement S10, j'ai accompli beaucoup de tâches. Premièrement, les fichiers exécutables de l'algorithme sont générés par le code langage C, puis ces heuristiques sont correctement installées dans le système, et l'algorithme heuristique a besoin des caractéristiques des paramètres d'entrée par utilisateur, donc j'ai optimisé toutes les pages qui se rapportent à l'algorithme heuristique, notamment pour la conception en profondeur et l'optimisation de l'affichage des types de méthodes sur la page, la boîte d'entrée de paramètre et le placement des données historiques. D'ailleurs le traitement différent de deux types différents de méthodes par le back-end Java est effectué différemment. Puis, comme l'algorithme exact précédent, j'ai utilisé le fichier de sortie pour calculer, comparer et afficher les résultats dans les tableaux sur la page Web. Enfin, chaque image est restaurée et les résultats de l'exécution de l'algorithme sont visualisés sur la page Web, plutôt que de toujours voir les résultats correspondants sous la forme d'une table. L'utilisateur peut maintenant voir complètement la correspondance entre les deux nœuds en regardant la ligne entre les deux images.

De plus, pour la visualisation de graphes, il existe deux types selon qu'il y a des coordonnées (x, y) dans « .dat ». Parmi eux, un système de gravité est utilisé pour empêcher l'empilement des nœuds quand il n'y a pas de coordonnée (x, y) dans « .dat ».

Le nombre de collection de graphiques pouvant être utilisés par le système est actuellement limité et il ne faut pas assez de temps pour tester plus de collection de graphiques. Le travail principal à l'avenir devrait se concentrer sur la recherche de normes plus générales pour optimiser les conditions de jugement, afin que le système puisse afficher des résultats correspondants pour plus de formats d'ensembles graphiques.

Contrairement au plan, j'ai seulement installé trois algorithmes heuristiques, car les exécutables des deux autres algorithmes heuristiques ne sont pas adaptés à ce système. Nous devons passer du temps à intégrer le code source du langage C de l'algorithme pour générer un fichier exécutable. Dans le futur travail de développement, il est possible de générer ces fichiers exécutables, mais pour l'instant, je n'ai pas le temps de plonger dans le langage C.

En bref, cette deuxième partie de ce projet a plus de pratique, elle m'a permis de découvrir beaucoup de nouvelles techniques de développement, d'approfondir mes connaissances en Java/JSP/JavaScript. Grâce à ce projet, j'ai eu une manière différente de développer en pensant toujours à avoir une meilleure qualité de code, respect des conventions de codage, documentation, tests unitaires, tests fonctionnels, etc.

4 Bilan sur la qualité

Dans cette section, je présenterai plusieurs points de suivi de la qualité du projet que j'ai résumés durant le développement de ce projet, afin d'essayer d'éviter certains risques du projet et d'améliorer la qualité du développement :

- Rédiger « cahier de spécifications » détaillé pour éviter le risque de livraison.
- Contrôler le temps de conception de page et améliorer l'efficacité de production de page Web.
- Faire du bon travail dans les bibliothèques publiques, clarifier les spécifications de développement, augmenter la réutilisation et réaliser un développement agile et rapide.
- Effectuer des tests unitaires et des tests d'intégration en temps opportun pour assurer que les petits modules ne comportent aucun bug.
- Communiquer souvent avec les clients, clarifier les exigences du projet, informer les risques le plus rapidement possible et augmenter la satisfaction.
- Soumettre le compte-rendu hebdomadaire pour éviter les retards du projet.

5 Bilan auto-critique sur la gestion du projet

Il y a quelques différences entre le diagramme de Gantt prévisionnel et le diagramme de Gantt réel, cela montre que je suis encore un peu inexpérimenté dans l'allocation du temps et des ressources. Mais dans l'ensemble, les travaux dans les deux phases de recherche et développement sont strictement conformes au diagramme de Gantt. Une fois qu'il y a un certain retard dans la progression du projet, je vais essayer d'améliorer mon efficacité au travail autant que possible, et les progrès globaux seront récupérés avant le prochain jalon.

Du point de vue des exigences du projet, j'ai rempli toutes les fonctions du cahier des charges du projet. Selon la méthode de développement « Agile », chaque phase de Sprint est terminée, je vais confirmer avec le client et commencer le prochain Sprint. Pendant cette période, je voudrais remercier mon tuteur pour mes conseils et suggestions, non seulement il peut corriger mes lacunes de conception, mais il peut également me donner des conseils du point de vue du client.

6 Conclusion

Selon le plan de gestion du projet, j'ai accompli la tâche donnée, pendant laquelle j'ai rencontré des difficultés. Cependant, les défis sont des opportunités, les difficultés m'ont appris à m'adapter, à résoudre et à résumer.

Ce projet m'a donc apporté de nombreux points positifs et m'a permis d'en apprendre plus sur moi-même. Les technologies que j'ai travaillées pourront être utiles dans le futur, étant des technologies très utilisées de nos jours.

J'ai eu la chance d'obtenir dans ce projet la compétence la plus importante comme un ingénieur, la recherche, la possibilité de réfléchir et les concevoir, la capacité de réaliser un grand projet de manière indépendante.

Enfin, je remercie les conseils et les aides de mes encadrants, M. Mostafa Darwiche, M. Romain Raveaux, M. Donatello Conte et M. Vincent T'kindt.

Annexes

A

Description des interfaces externes du logiciel

1 Interfaces matériel/logiciel

Dans ce projet, nous allons développer une interface web. Pour cela, l'utilisateur devra être muni d'un ordinateur et plus précisément d'un navigateur et d'un service d'accès Internet afin de pouvoir accéder au système.

2 Interfaces logiciel/logiciel

Afin de calculer l'algorithme et le modèle, nous allons utiliser CPLEX Optimizer qui permet le développement rapide et le déploiement de modèles d'optimisation de décision.

Afin d'interagir avec la base de donnée le site web fera appel à un serveur de base de données embarqué H2 puisqu'il fournit une console Web très pratique pour l'exploitation et la gestion du contenu de la base de données, et il est API JDBC très rapide. Afin de faciliter le développement de la base de données, nous utilisons le Framework Hibernate.

Nous avons choisi Apache Tomcat comme WEB serveur pour fournir des services de navigation d'informations en ligne, et Tomcat est un conteneur JSP/servlet pour publier JSP et Java.

3 Interfaces homme/machine

Interfaces homme/machine d'origine sont dessinées par « moqups » avant mettre en oeuvre.

Dans la page d'accueil existant [Figure 1](#), il y a quatre options y compris « Modèles », « Datasets », « Tests » et « Exécution ». Ces quatre options contiennent essentiellement la fonction principale du système, en cliquant sur le bouton de ces quatre options à réaliser. Pour améliorer ce système, nous allons transformer « Modèles » en « Méthodes ». De cette manière, le système permet à l'utilisateur de résoudre le problème de distance d'édition le plus court grâce à un modèle mathématique exact, et des heuristiques peuvent également être utilisées pour résoudre le problème. Dans l'interface présentée de suivi, nous allons aussi changer l'interface associée au « Modèles » en « Méthodes ».

Chaque option a deux/trois choix pour afficher la liste de méthode [Figure 2](#), la liste de dataset [Figure 5](#), la liste de test [Figure 7](#), la liste d'exécution [Figure 9](#) ou pour ajouter une nouvelle

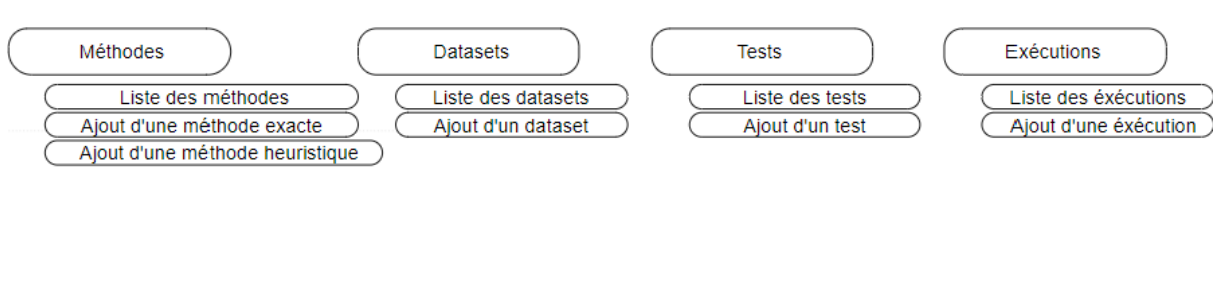


Figure 1 – La page d'accueil

méthode exacte [Figure 3](#), ajouter une nouvelle méthode heuristique [Figure 4](#), ajouter un nouveau dataset [Figure 6](#), ajouter un nouveau test [Figure 8](#).

Rechercher :

Par nom / description :

Show entries

| ▼ Nom | ▼ Description | ▼ Chemin exécutable |
|-------|---------------|--------------------------------------|
| 1 | D1 | ProjetPRD\Methodes\1\IPPrepro_JH.exe |
| 2 | D2 | ProjetPRD\Methodes\2\IPPrepro_F1.exe |
| 3 | D3 | ProjetPRD\Methodes\3\IPPrepro_F2.exe |

Previous 1 Next

Showing 1 to 3 of 3 entries

[+ ajouter une méthode exacte](#)

[+ ajouter une méthode heuristique](#)

Figure 2 – La page de Liste des Méthodes

Ces pages sont similaires sauf la page de « Ajout d'un test » [Figure 8](#) parce qu'elle est plus compliquée et il y a plus de choses à choisir et à remplir. Dans l'optimisation du projet de suivi, nous continuerons probablement à modifier cette page selon la fonction à ajouter. Par exemple, la partie la plus importante est les champs optionnels de paramètre puisque les méthodes différentes ont des paramètres différents qui sont nécessaire à saisir.

Après que les utilisateurs font une exécution, c'est possible pour lui de regarder les détails de cette exécution comme la [Figure 9](#) ci-dessous, et lorsque une clique de bouton « Résultats » marche, tous les résultats sur ce test sont présentés.

Voici l'interface du résultat [Figure 10](#), l'utilisateur peut choisir quelles données et variables à afficher sous différents critères (ex. Nœuds explorés, Moyen de temps d'exécution, déviation, nombre des solutions optimales, etc.).

Méthodes
Datasets
Tests
Exécutions

Ajout d'une méthode exacte : _____

Nom :

Executable :
 N'a choisi aucun fichier

Description :

Figure 3 – La page de l'ajout d'une méthode exacte

Méthodes
Datasets
Tests
Exécutions

Ajout d'une méthode heuristique : _____

Nom :

Executable :
 N'a choisi aucun fichier

Description :

Paramètres nécessaires :

| | | |
|--|--|--|
| Nom : <input style="width: 100px;" type="text"/> | Type : <input type="text" value="int"/> | <input type="button" value="- supprimer"/> |
| Nom : <input style="width: 100px;" type="text"/> | Type : <input type="text" value="float"/> | <input type="button" value="- supprimer"/> |
| Nom : <input style="width: 100px;" type="text"/> | Type : <input type="text" value="double"/> | <input type="button" value="- supprimer"/> |

Figure 4 – La page de l'ajout d'une méthode heuristique

Méthodes Datasets Tests Exécutions

Liste des collections de graphes

Rechercher :
 Par nom / description : Par type de graphe **Non dirigé** ▼

Show ▼ entries

| ▼ Nom | ▼ Description | ▼ Dirigé | ▼ Nb graphes | ▼ Dataset | ▼ Subset |
|-------|---------------|------------|--------------|------------------------------|-----------------------------|
| 1 | D1 | Non dirigé | 4337 | ProjetPRD\Datasets\1\Dataset | ProjetPRD\Datasets\1\Subset |
| 2 | D2 | Non dirigé | 1100 | ProjetPRD\Datasets\2\Dataset | ProjetPRD\Datasets\2\Subset |

Showing 1 to 2 of 2 entries

Previous Next

[+ ajouter une collection](#)

Figure 5 – La page de Liste des Datasets

Méthodes Datasets Tests Exécutions

Ajout d'une collection de graphes :

Nom :

Dataset :

N'a choisi aucun fichier

Subset :

N'a choisi aucun fichier

Description :

Figure 6 – La page de l'ajout d'un Dataset

Méthodes
Datasets
Tests
Exécutions

Liste des tests

Rechercher :

Par nom / description : Par méthode Par dataset

Show entries

| ▼ Nom | ▼ Description | ▼ Mode | ▼ Nb executions | ▼ Méthodes | ▼ Datasets |
|-------|---------------|--------|-----------------|------------|---|
| 1 | D1 | IP | 0 | 1, 1, 1, 2 | train10 (1), train10 (1), train10 (2), train10 (1), train10 (2) |

Showing 1 to 1 of 1 entries

Previous Next

Nouvelle exécution du test 1

Nom :

Description :

Exécuter le test

+ ajouter un test

Figure 7 – La page de Liste des Tests

Méthodes
Datasets
Tests
Exécutions

Ajout d'un test :

☐ Méthodes

| | |
|-------------------------------------|---|
| <input checked="" type="checkbox"/> | 1 |
| <input checked="" type="checkbox"/> | 2 |
| <input type="checkbox"/> | 3 |

☐ Datasets

| Datasets | Subsets |
|-------------------------------------|--------------|
| <input type="checkbox"/> | 1 |
| <input type="checkbox"/> | mixed-graphs |
| <input checked="" type="checkbox"/> | train10 |
| <input type="checkbox"/> | train20 |
| <input type="checkbox"/> | train30 |

Test

Nom :

Description :

Mode d'exécution :

Showing 1 to 3 of 3 entries

Showing 1 to 15 of 15 entries

Paramètres pour les exactes :

Mémoire limite (en Mo) :

Temps limite (en sec) :

Temps limite heuristique (en sec) :

Tolérance d'écart :

Nombre de threads :

Paramètres supplémentaires :

+ Ajouter

Paramètres pour 1 :

| Nom | Type | Valeur |
|-------------|--------|----------------------|
| Paramètre 1 | String | <input type="text"/> |
| Paramètre 2 | int | <input type="text"/> |
| Paramètre 3 | String | <input type="text"/> |

Enregistrer

+ ajouter un test

Liste des tests

Figure 8 – La page de l'ajout d'un test

Méthodes Datasets Tests Exécutions

Liste des exécutions

Rechercher :
 Par nom / description : Par test : 1 ▼ Par état : En cours ▼

Show 10 ▼ entries

| ▼ Nom | ▼ Description | ▼ Test | ▼ Nb instances | ▼ Etat |
|-------|---------------|--------|----------------|----------|
| 1 | D1 | 1 | 300/400 | En cours |

Showing 1 to 2 of 2 entries (filtered from 3 total entries)

Previous 1 Next

Détails de l'exécution 1 :

Exécutions : 300 / 400 75.0 %
 État : En cours
 0 echec(s)

Résultats
 Masquer cette exécution

+ ajouter une exécution

Figure 9 – L’affiche de détail d’une exécution

Méthodes Datasets Tests Exécutions

Résultats de l'exécution 1 :

| Général | Temps CPU | Noeuds explorés | Solutions optimales | Instances traitées | Déviations | Appareillements | Visualisation de Matching |
|---|-----------|-----------------|---------------------|--------------------|------------|-----------------|---------------------------|
| <ul style="list-style-type: none"> Mode : IP 300 / 400 instances Méthodes : 1, 2 Datasets : train10 (1) ,train10 (2) Paramètres : <ul style="list-style-type: none"> memoire : 1000 thread : 1 tempsHeur : 900 tolerance : 0.1 temps : 900 | | | | | | | |

Liste des exécutions

Figure 10 – L’interface du résultat

B

Spécifications fonctionnelles

Dans cette partie, nous allons décrire les différentes fonctionnalités à mettre en place ou à modifier. Ainsi chacune d'entre elles sera précisément décrite dans le cas où il s'agit d'une amélioration/modification de l'existant.

1 Ajout d'une méthode

Nom de la fonction : Ajout d'une méthode

Priorité : primordiale, c'est obligatoire d'ajouter quelques méthodes pour exécuter un test.

Au début, il n'existe pas de méthode affichée dans le système. Il est donc nécessaire d'ajouter cette fonctionnalité au site. Le système va enregistrer et afficher la méthode qui est choisie par l'utilisateur et elle est ajoutée à la base de données. En ce moment-là, quand nous ajouterons un test, notre interface prend en entrée une méthode qu'elle doit être en mesure d'afficher afin de permettre à l'utilisateur d'effectuer une sélection sur celles-ci.

Entrées : Méthode (.exe)

Sorties : Ajout de la méthode à la base de données

Précondition : Méthode valide

2 Ajout d'un Dataset/Subset

Nom de la fonction : Ajout d'une collection de graphes

Priorité : primordiale, c'est obligatoire d'ajouter quelques collections d'images pour exécuter un test.

Au début, il n'existe pas de Dataset/Subset (la base de données d'images/les collections des images). Il est donc nécessaire d'ajouter cette fonctionnalité au site. Le système va enregistrer et afficher la base de données d'images qui est choisi par l'utilisateur et il est ajouté à la base de données. Du coup quand nous ajouterons un test, notre interface prend en entrée une base de données qu'elle doit être en mesure d'afficher afin de permettre à l'utilisateur d'effectuer une sélection sur celles-ci.

Entrées : Dataset/Subset (.zip)

Sorties : Ajout de Dataset/Subset à la base de données

Précondition : Dataset/Subset valide, Dataset et Subset correspondant

3 Ajout d'un Test

Nom de la fonction : Ajout d'un test

Après que l'utilisateur ajout quelques méthodes et quelques collections d'images, il peut ajouter un nouveau test. Cette fonctionnalité permet à l'utilisateur de choisir les méthodes à comparer avec une collection d'image particulière, de saisir les paramètres essentiels, de sélectionner le mode d'exécution, d'ajouter des commentaires nécessaires.

Entrées : Entrée de l'utilisateur et l'élément sélectionné sur le site

Sorties : Ajout de test à la base de données

Précondition : L'option requise ne peut pas être vide

4 Ajout d'une exécution

Nom de la fonction : Ajout d'une exécution

Après que l'utilisateur ajout un test, il peut ajouter une nouvelle exécution. Cette fonctionnalité permet à l'utilisateur de regarder le processus d'exécution pour un test, de regarder l'état de l'exécution, de regarder le nombre d'instances y compris le nombre d'instances réussi et le nombre d'échecs, de comparer les résultats et d'examiner la visualisation.

Entrées : Entrée de l'utilisateur sur le site

Sorties : Ajout d'exécution à la base de données

Précondition : Test valide

5 Liste de Méthode/Dataset/Test/Exécution

Nom de la fonction : Afficher la liste

Cette fonctionnalité permet à l'utilisateur de regarder la liste de Méthode/Dataset/Test/Exécution par cliquer le bouton optionnel.

6 Recherche de Méthode/Dataset/Test/Exécution dans la liste

Nom de la fonction : Recherche d'un objet la liste

Cette fonctionnalité permet à l'utilisateur de rechercher un objet particulier dans la liste de Méthode/Dataset/Test/Exécution. Il peut rechercher par nom, par état, par description, etc.

7 Sauvegardes des résultats

Nom de la fonction : Sauvegardes des résultats

Cette fonctionnalité permet de sauvegarder localement les résultats du calcul donc les utilisateurs peuvent regarder les sorties dans le fichier de « .txt ».

Entrées : Les résultats d'une exécution

Sorties : Le fichier « .txt »

Précondition : Exécution valide

8 Affiche des résultats dans le tableau ou dans le diagramme

Nom de la fonction : Affiche des résultats

Cette fonctionnalité permet de la lecture des fichiers de sortie pour extraire les minimums, moyennes et maximums correspondant au critère, donc les utilisateurs peuvent regarder ces résultats dans le tableau ou dans le diagramme.

Entrées : Les fichiers de sortie

Sorties : Le tableau/le diagramme dans le site

Précondition : Le plugin Google Charts valide

9 Visualisation

Nom de la fonction : Visualisation

Cette fonctionnalité permet de réaliser la visualisation de Matching trouvé entre deux graphes. Par conséquent, l'utilisateur peut intuitivement voir la correspondance des nœuds et des arêtes dans deux images.

Entrées : Les fichiers de sortie, les images, les coordonnées

Sorties : Le résultat de visualisation dans le site

C

Spécifications non fonctionnelles

1 Contraintes de développement et conception

Le projet existant induit quelques contraintes concernant les points suivants :

- Langage de programmation imposé : Java, JSP, Javascript, CSS
- Logiciels de base à utiliser pour le développement : Eclipse/IntelliJ
- Environnements nécessaires : Windows/Mac
- Bibliothèques de programmes imposées : IBM®ILOG®CPLEX®Optimization Studio 12.6.0, Google Charts, Echarts
- Apache Tomcat, navigateurs

De plus, le délai de réalisation est 30/03/2018.

2 Contraintes de fonctionnement et d'exploitation

2.1 Performances

Du point de vue de l'utilisateur, le temps de réponse devra être raisonnable mais il dépend de sa connexion internet et des performances globales de l'ordinateur. Lors de l'exécution d'une opération de saut de page, ce temps ne devra pas excéder 5 secondes pour assurer un confort d'utilisation.

2.2 Capacités

Théoriquement, tant que l'utilisateur a installé le système avec succès, le système peut supporter la charge nécessaire. Donc les limites ne sont pas fixées, de plus elles dépendront de l'efficacité de nos algorithmes.

2.3 Contrôlabilité

Une barre de défilement permet d'indiquer le déroulement lorsqu'un test est en cours d'exécution.

Une alerte est affichée pour prévenir l'utilisateur lorsqu'une erreur ou un manque d'information. Afin de suivre l'exécution du programme, des messages et des résultats seront affichés en console.

2.4 Sécurité

Aucune demande particulière n'a été faite de la part de la MOA en ce qui concerne la sécurité.

3 Maintenance et évolution du système

Afin de pouvoir maintenir ce projet et faciliter la reprise de celui-ci, la documentation cahier des charges va être rédigée ainsi que des documents d'analyses, y compris doc d'installation/déploiement, doc d'utilisation, cahier du développeur et cahier de test. Aussi pour permettre un suivi précis des développements et des différents livrables, les outils de gestion de version Github et Trello seront utilisés. De cette façon, le code source et les documents rédigés seront accessibles par tous les acteurs du projet. En plus, le code source doit être clair et compréhensible pour le futur développement et de sa maintenabilité.

D

Analyse et cahier du développeur

Dans cette section, je vais décrire en détail la méthode de développement, la structure du code et la réalisation de la conception de toutes les nouvelles fonctionnalités qui doivent être mises en œuvre cette année.

En outre, la documentation du code est générée par JavaDoc, donc tous les noms des fonctions et tous des commentaires suivent les spécifications indiquées dans la **Figure 1** ci-dessous.

```
/**
 * Retourne la liste des valeurs des paramètres heuristiques pour chaque méthode
 * quand on va exécute un test qui contient deux méthodes heuristiques
 * @param id identifiant de la méthode heuristique
 * @return la liste des valeurs des paramètres heuristiques
 */
public ArrayList<String> getEachParamsValeurs(long id) {
```

Figure 1 – Convention de nommage et de commentaire

1 Le structure du projet

Voici la structure détaillée de ce système, **Figure 2**, les utilisateurs importent des méthodes et des bases de données d'images pour calculer les GED sur des pages Web afin de tester et d'afficher les résultats d'exécution. Lorsque le serveur reçoit la requête de l'utilisateur, il accède à la page spécifiée et appelle les fonctions dans le back-end pour répondre à la demande de l'utilisateur. Le résultat de l'exécution est affiché sur la page Web et stocké dans le document « .txt » local. Dans le même temps, le back-end fera l'opération également la base de données via la couche DAO.

Le développement du système suit le modèle de conception MVC.

1.1 Modèles

JavaBean encapsule la logique et le métier, utilisé pour enregistrer les entités de données, correspondant généralement aux tables de la base de données, nous allons voir la structure des différentes modèles dans la **Figure 3** et leur relation est commue le diagramme **Figure 4**.

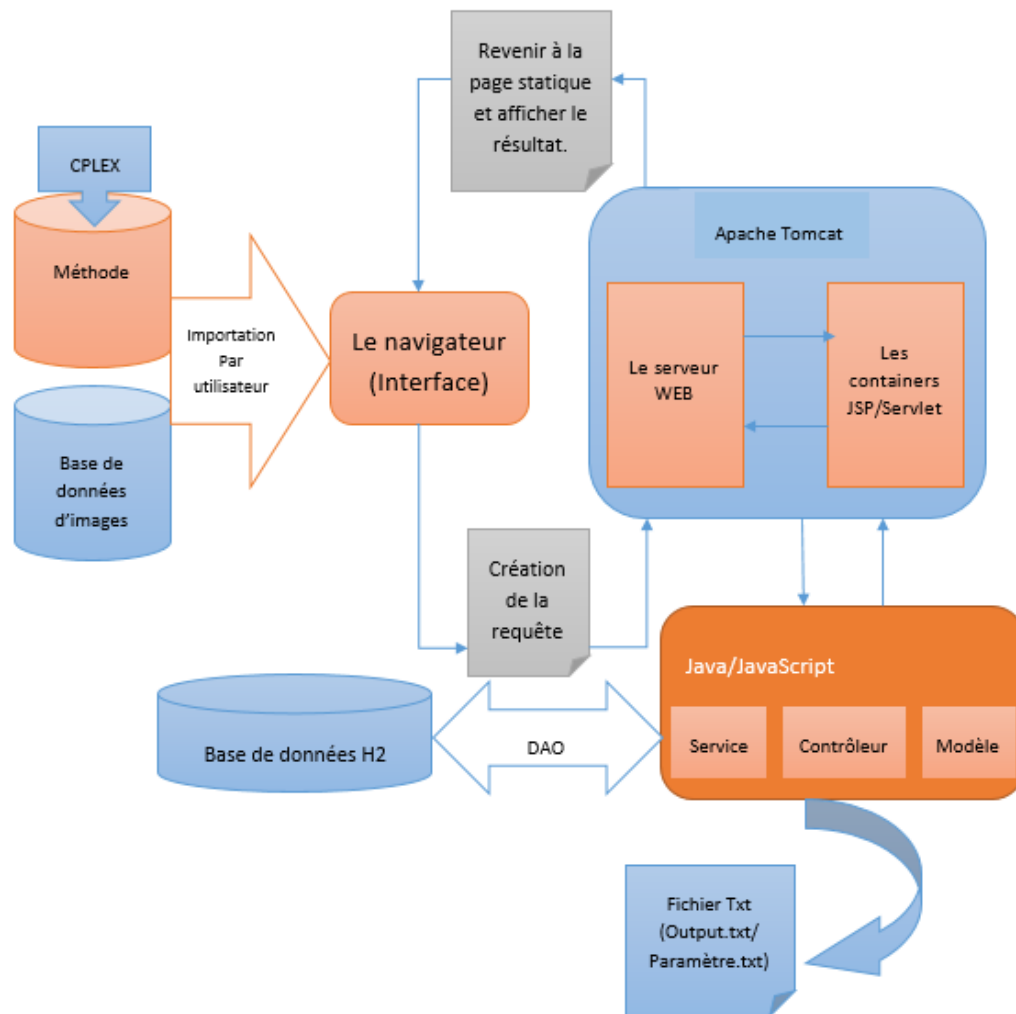


Figure 2 – L'architecture générale du système

1.2 Contrôleurs

Servlet est le contrôleur, utilisé pour contrôler le flux du programme et appeler des services de traitement. Ces contrôleurs (comme présente dans la Figure 5) permettant de faire le lien entre les données et les vues et ils permettent dans la majeure partie des cas de faire appel à un modèle qui lui-même exécute une requête à la base de données et le contrôleur envoie ces données à la vue qui se charge de les afficher.

Nous allons prendre un exemple dans la Figure 6 pour voir l'utilisation d'un contrôleur.

La méthode de « `doGet(HttpServletRequest, HttpServletResponse)` » est pour afficher la fiche d'un Methode ou une fiche vide pour une création renvoie sur « `ficheMethodeHeuristique.jsp` », c'est-à-dire obtenir des données du serveur / back-end.

La méthode de « `doPost(HttpServletRequest, HttpServletResponse)` » est pour créer ou modifier un objet Methode renvoie sur « `ficheMethodeHeuristique.jsp` », c'est-à-dire obtenir récupérer les données de front-end / server vers Model.

1.3 Vues

Les vues de notre système sont dans le dossier « WebContent » comme la Figure 7. Il contient trois types de fichiers : CSS, JavaScript et JSP. Parmi eux, CSS est de contrôler le style et la

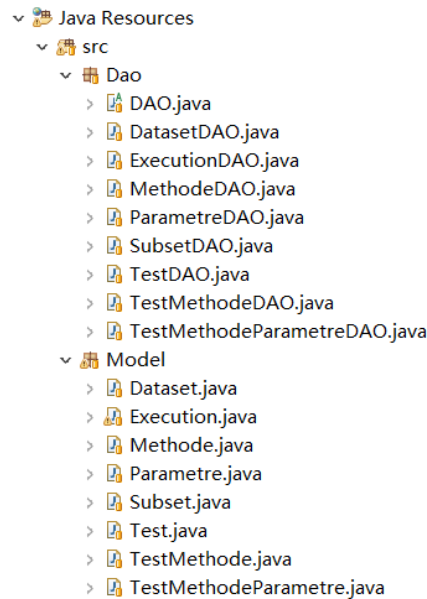


Figure 3 – La structure des différents modèles

disposition des pages Web, JavaScript est un langage de script qui appartient au web, il est couramment utilisé pour ajouter diverses fonctions dynamiques pour les pages Web, offrant aux utilisateurs des effets de navigation plus doux et plus beaux. Dans ce système, nous l'utilisons également pour fournir un traitement de données de base et des jugements conditionnels pour les pages Web portant le même nom que JSP afin de réduire l'interaction avec le back-end et économiser les ressources du serveur. JSP est notre vraie vue de serveur, il contient beaucoup de bibliothèques de balises pour notre utilisation, par exemple « JSTL (JSP Standard Tag Library) ».

2 Optimisation de structure de la base de données

Sur la base de la structure de base de données d'origine, deux tables d'entités « parametre », « test_methode » ont été ajoutées. La table « parametre » sert à stocker les informations sur les paramètres de l'algorithme heuristique.

La table « test_methode » est destinée à stocker toutes les méthodes ajoutées au test et leurs tests correspondants, c'est-à-dire les identifiants de toutes les méthodes contenues dans chaque test. De cette manière, la table de « test » et la table de « methode » sont en réalité dans une relation plusieurs-à-plusieurs, et seule une table relationnelle générée automatiquement entre elles est suffisante. Cependant, les exigences du système sont les suivantes : L'entrée de la valeur du paramètre d'algorithme heuristique n'est pas fixée par la méthode (la méthode ne peut que déterminer le nom et le type du paramètre), l'utilisateur peut entrer librement la valeur du paramètre lors de l'ajout du test (test détermine la valeur du paramètre). Par conséquent, nous avons besoin d'une table relationnelle avec un attribut de valeur, et nous pouvons connaître les informations de paramètre, les informations de méthode et les informations de test à travers la clé étrangère de cette table « test_methode_params ». Nous avons donc besoin de « test_methode » comme table d'entités pour établir toutes les relations sur les valeurs des paramètres. Enfin, nous pouvons trouver des informations via le diagramme de structure de la base de données. Le diagramme MCD est présenté dans Figure 8 et le diagramme MLD est présenté dans Figure 9.

Après avoir terminé la conception de la structure de la base de données, nous avons commencé à établir une relation de mappage dans la classe Java. Comme mentionné précédemment, nous utilisons le framework Hibernate pour connecter et configurer la base de données dont la

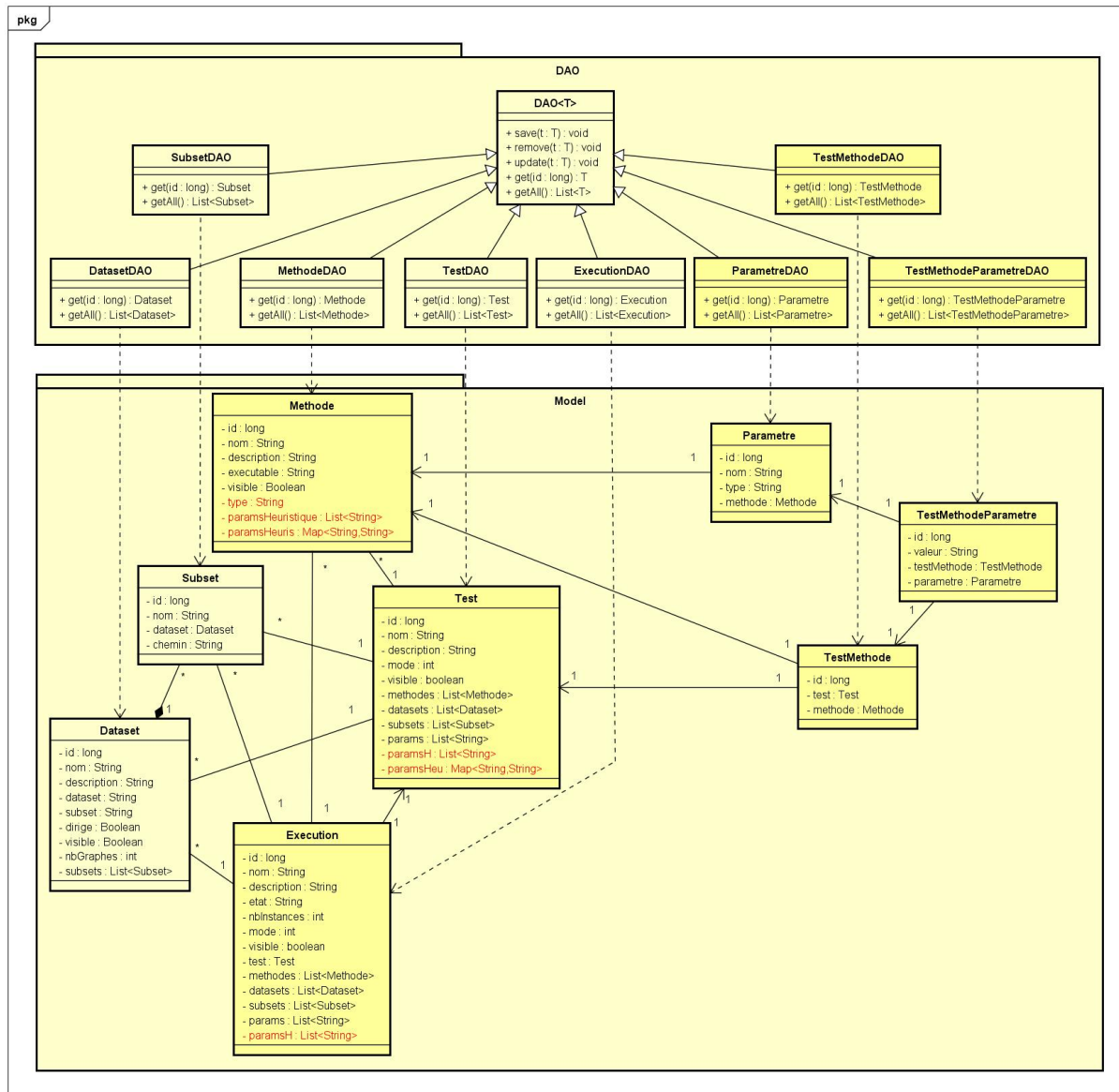


Figure 4 – Le diagramme de Model & DAO

réalisation concrète est programmée dans le fichier de « DatabaseConnexion.java », et nous utilisons la couche Dao pour encapsuler les opérations de la base de données.

Par conséquent, nous avons ajouté d'abord les informations de mapping représentées dans la Figure 10 suivante au fichier de configuration « hibernate.cfg.xml ».

Ensuite, nous devons créer trois classes JavaBean « Parametre.java », « TestMethode » et « TestMethodeParametre » dans la couche « Model ». Chaque classe déclare la table d'entité, les informations d'ID et les relations avec d'autres tables (@Entity – Table, @Id – Clé primaire, @OneToMany, @ManyToOne, @ManyToMany – Clé étrangère). En même temps, trois fichiers d'accès aux données correspondant au même nom JavaBean class sont créés dans la couche Dao.

3 Implémenter des heuristiques en les intégrant dans le système

Cette section présente principalement l'implémentation et l'exécution d'algorithmes heuristiques, y compris tous les travaux de développement front-end et back-end.























- ▼  Servlets.Datasets
 - >  DeleteDatasetServlet.java
 - >  FicheDatasetServlet.java
 - >  ListeDatasetsServlet.java
- ▼  Servlets.Execution
 - >  DeleteExecutionServlet.java
 - >  ExecExecution.java
 - >  InfosExecutionServlet.java
 - >  ListeExecutionsServlet.java
 - >  NewExecutionServlet.java
- ▼  Servlets.Methode
 - >  DeleteMethodeServlet.java
 - >  FicheMethodeExacteServlet.java
 - >  FicheMethodeHeuristiqueServlet.java
 - >  ListeMethodeServlet.java
- ▼  Servlets.Resultat
 - >  DetailAppareillement.java
 - >  FicheResultatServlet.java
- ▼  Servlets.Test
 - >  DeleteTestServlet.java
 - >  FicheTestServlet.java
 - >  ListeTestServlet.java

Figure 5 – Le contenu des différentes contrôleurs

```

/**
 * Affiche la fiche d'un Methode ou une fiche vide pour une création
 * renvoie sur ficheMethodeHeuristique.jsp
 * @param req requete contenant éventuellement l'identifiant de la Methode à afficher (dataset)
 * @param resp reponse
 * @throws ServletException
 * @throws IOException
 */
protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {}

/**
 * Créé ou modifie un objet Methode
 * renvoie sur ficheMethodeHeuristique.jsp
 * @param req requete contenant les informations saisies dans le formulaire
 * @param resp reponse
 * @throws ServletException
 * @throws IOException
 */
protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {}

```

Figure 6 – Une exemple pour expliquer le servlet comme contrôleur

3.1 Module de « Méthode »

D'abord créer « ficheMethodeHeuristique.jsp » pour afficher la page d'ajouter une méthode heuristique et créer « FicheMethodeHeuristiqueServlet.java » pour répondre la requête de « ficheMethodeHeuristique.jsp ». Dans le même temps, séparer le traitement de la page pour les algorithmes exactes et heuristiques. Un fichier « ficheMethodeHeuristique.js » supplémentaire contenant la fonction « addParamHeuristique() » est ajouté. Cette fonction retourne un « div » pour stocker le nom et le type d'un paramètre dans la méthode ajoutée par l'utilisateur.

Dans la fichier de « FicheMethodeHeuristiqueServlet.java », il reçoit les données envoyées depuis le front-end et les transmet en back-end : "Récupère les données du formulaire, stocke les données de type chaîne, reprend les éléments du formulaire et appelle une fonction de back-end pour ajouter les données à la classe correspondante de « Model »."

Afin de mieux traiter les données envoyées depuis le front-end, deux attributs « paramsHeuristique : List<String> » et « paramsHeuris : Map<String, String> » sont ajoutés dans « Methode.java », leur relation est « @ElementCollection », c'est un composant qui ne peut pas être séparé de l'entité « Methode ». En conséquence, la fonction « add », la fonction « get » et la fonction « clear » sont créées pour modifier les données en fonction de la opération de l'utilisateur.

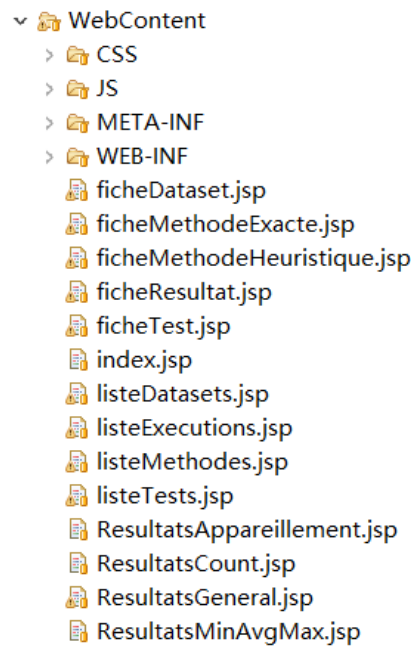


Figure 7 – La structure des différentes vues

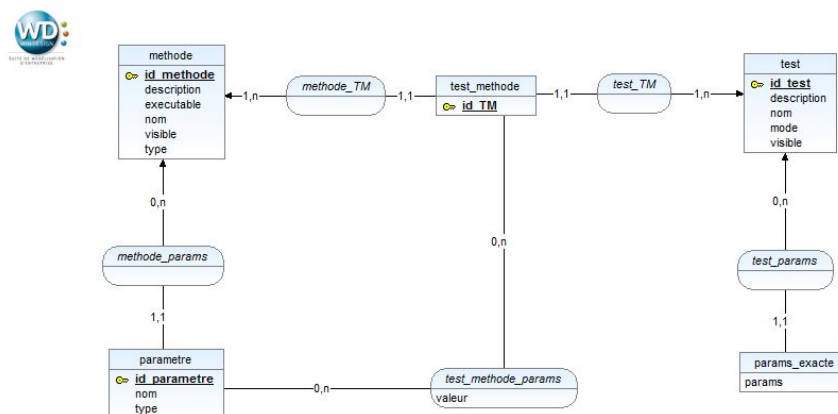


Figure 8 – Le diagramme MCD pour la partie ajoutée & modifiée

En outre, une fonction « `dejaTest()` » est ajoutée dans « `Methode.java` » pour déterminer si l'heuristique actuelle a été ajoutée au test. Dans ce cas, cette méthode ne peut pas être modifiée. La méthode de traitement de front-end est la suivante : si la méthode heuristique actuelle est en test, toutes les parties modifiables de la page sont grisées, et les parties de « `input` » et « `select` » pour la saisie de « `Nom de paramètre` » et le choix de « `type de paramètre` » utilisent respectivement « `readOnly` » et « `disabled` ». En même temps, puisque la méthode qui a été ajoutée au test ne peut pas être soumise, les boutons pour la méthode d'ajout, d'enregistrement, de masquage et de téléchargement sont annulés et une annonce est ajoutée dans la position d'origine.

3.2 Module de « Test »

Comme « Methode.java », afin de mieux traiter les données envoyées depuis le front-end, deux attributs « paramsH : List<String> » et « paramsHeu : Map<String, String> » sont ajoutés dans « Test.java », leur relation est aussi « @ElementCollection » qui ne peut pas être séparé de l'entité «

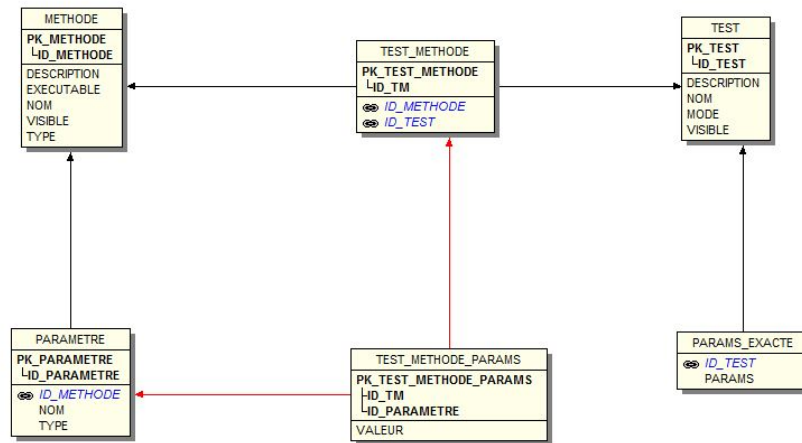


Figure 9 – Le diagramme MLD pour la partie ajoutée&modifiée

```

<mapping class="Model.Parametre"/>
<mapping class="Model.TestMethode"/>
<mapping class="Model.TestMethodeParametre"/>

```

Figure 10 – Les mappings pour les classes ajoutées

Test ». En conséquence, la fonction « add », la fonction « get » et la fonction « clear » sont créées pour modifier les données en fonction de la opération de l'utilisateur.

De plus, certaines fonctions du fichier original « Test.java » ont été modifiées pour séparer les opérations de l'algorithme exacte et de l'algorithme heuristique. L'une des plus importantes est la fonction « fichierParam() », dont le rôle est de stocker tous les paramètres saisis par l'utilisateur dans un fichier local.

Contrairement à « FicheMethodeHeuristiqueServlet.java », le traitement de « FicheTestServlet » devient plus compliqué car il a besoin de traiter plus d'informations et de les distribuer à différentes tables de données, qui sont présentées dans la Figure 11 ci-dessous. Donc lorsque nous traitons les données pour la module de Test dans ce servlet, nous devons considérer strictement la structure d'association de la base de données.

```

private final MethodeDAO methodeDAO = new MethodeDAO();
private final DatasetDAO datasetDAO = new DatasetDAO();
private final TestDAO testDAO = new TestDAO();
private final TestMethodeDAO testmethodeDAO = new TestMethodeDAO();
private final ParametreDAO parametreDAO = new ParametreDAO();
private final TestMethodeParametreDAO testmethodeparametreDAO = new TestMethodeParametreDAO();

```

Figure 11 – La couche d'accès aux données utilisée

En raison de la complexité des données traitées par ce module, outre le transfert nécessaire des données vers le back-end et les données provenant du back-end, la plupart des autres opérations sont effectuées dans les fichiers de « ficheTest.js » pour réduire les interactions entre le back-end et le front-end. Un autre avantage est que cela permet d'économiser la charge du serveur dans une certaine mesure, en rendant les ressources du serveur disponibles pour plus d'utilisateurs en même temps.

1. Sélectionner une méthode. Après que l'utilisateur a sélectionné une méthode heuristique, la page affiche automatiquement la boîte de saisie des paramètres ci-dessous, où le nom et le type du paramètre ont déjà été écrits, et l'utilisateur n'a besoin que d'entrer la valeur du paramètre. Idées :

- Générer itérativement des boîtes de saisie. Ajouter une fonction « getParams() » qui génère une boîte d'entrée de manière cyclique via « <c : forEach> ». Afin de garantir que les

éléments entre plusieurs boîtes d'entrée ne sont pas remplacés, l'identifiant de l'élément dans la boîte de saisie généré est défini sur « already ».

- Générer le même nombre de lignes que les paramètres. Les paramètres de l'algorithme heuristique sont lus par la fonction « getParams() » et le processus de segmentation de chaîne est exécuté. Ajouter ensuite une fonction « addParamHeuristique() » pour générer le même nombre de lignes.

2. Désélectionner une méthode. L'opération de suppression traditionnelle (définition de « div.style.display » sur « none ») génère un bug qui entraîne la même existence de « input » après la désélection d'une méthode heuristique, cela a causé que le formulaire ne soit pas soumis. J'ai donc utilisé la méthode « parentNode.removeChild » pour supprimer complètement le nœud.

3. Modifier et interroger dans la page de « Modification du test ». Les heuristiques qui ont été sélectionnées dans le test en cours et leurs boîtes de saisie de paramètres et les données historiques sont automatiquement reproduites.

- Reproduire automatiquement la boîte de saisie des paramètres. Comme la boîte de saisie des paramètres sur la page est chargée à la fin, les informations de la méthode sélectionnée dans le test en cours ne peuvent pas être lues en utilisant la méthode traditionnelle. Après avoir obtenu les informations de choix d'utilisateur, une fonction « getParams2() » qui est similaire à « getParams() » est ajoutée et elle génère une boîte de saisie de paramètre pour la lecture de la méthode déjà choisie.
- Entrer les données d'historique pour la boîte de saisie des paramètres. Ajouter une fonction « addParamHeuristique2() » qui est similaire à la fonction « addParamHeuristique() ». En plus du traitement des éléments « name » et « type » contenus dans la fonction « addParamHeuristique() », un traitement supplémentaire de l'élément « value » est ajouté.

4. Éviter les utilisateurs choisissant deux types de méthodes. Une fois que l'utilisateur a sélectionné une méthode heuristique, il ne peut plus sélectionner la méthode exacte et vice versa.

- L'utilisateur ne peut pas choisir sous différents types. Ajouter le paramètre « selectFlag » à la fonction « getParams() » de « ficheTest.js ». Quand une méthode de type différent est sélectionnée, le « flag » en cours (selectFlagNew) et « selectFlag » ne sont pas égaux, et une alerte apparaîtra. Dans le même temps, supprimer la boîte dynamique de saisie de paramètre et décocher la méthode automatiquement.
- Réinitialiser l'opération. Ajouter le paramètre « countFlag » à « getParams() » et juger si toutes les méthodes sont annulées, tout est restauré dans son état d'origine.

3.3 Module de « Exécution »

En plus de la nécessité de définir certains attributs clés comme « Test.java » et « Methode.java », la **Figure 12** ci-dessous montre toutes les fonctions nécessaires à l'installation de l'algorithme heuristique sans affecter l'exécution de l'algorithme exact. Afin de séparer l'algorithme heuristique de l'algorithme exact, ajouter les conditions de jugement nécessaires dans la fonction « run() » pour appeler respectivement deux fonctions « lancerExecutionExacte(List<String>, Runtime, String[], boolean) » et « lancerExecutionHeuristique(List<String>, Runtime, String[], boolean) ». Parmi eux :

- « getResultats(info : String) » : Lire des fichiers de sortie pour extraire les minimums, moyennes et maximums correspondant au critère info
- « getListeAppareillement() » : Lire des fichiers de sortie pour extraire les paires de graphes sur lesquelles on a exécuté les Méthode
- « getDetailAppareillement(idModele : long, g1 : String, g2 : String) » : Parcourir le fichier de sortie pour extraire l'appareillement des sommets et des arcs entre deux graphes pour

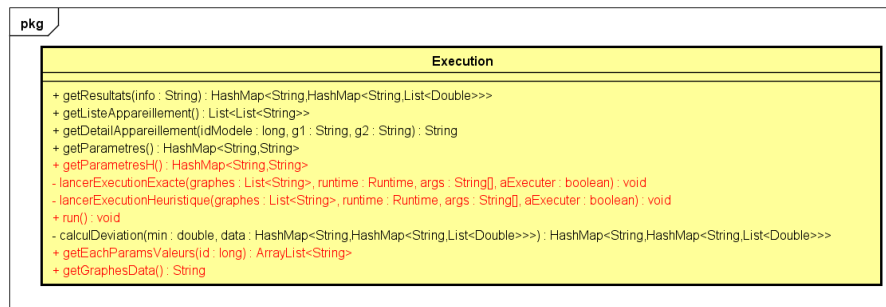


Figure 12 – Les fonctions importantes dans la classe de « Execution »

une Méthode

- « getParametresH() » : Retourner la liste des paramètres heuristiques associés à l'Execution sous forme de HashMap
- « lancerExecutionExacte (graphes : List<String>, runtime : Runtime, args : String[], aExecuter : boolean) » : Appel à l'exécutable du méthode exacte pour chaque paire de graphes
- « lancerExecutionHeuristique(graphes : List<String>, runtime : Runtime, args : String[], aExecuter : boolean) » : Appel à l'exécutable du méthode heuristique pour chaque paire de graphes
- « run() » : Faire l'exécution et mettre à jour l'état de l'Exécution
- « calculDeviation(min : double, data : HashMap<String,HashMap<String,List<Double>>>) » : Calculer la déviation de la fonction objectif
- « getEachParamsValeurs(id : long) » : Retourner la liste des valeurs des paramètres heuristiques pour chaque méthode quand on va exécute un test qui contient quelques méthodes heuristiques
- « getGraphesData() » : Lire les informations ligne par ligne de chaque graphe sur le fichier « .dat »

Les fonctions décrites au **Section 5** sont principalement implémentées en appelant ces fonctions ci-dessus.

4 Evaluer les méthodes différentes sur des bases d'images

Pour exécuter un algorithme heuristique, vous devez compiler le fichier de langage C de l'algorithme dans Visual Studio afin de générer un fichier exécutable au format « .exe ». En même temps, les paramètres du système respectent les paramètres entrants dans la langue C.

Puisque l'utilisateur peut créer une exécution à tout moment, nous devons créer un mécanisme de multithreading. Par conséquent, nous utilisons le framework de « Java Executor » et l'implémentation de l'interface « Runnable ».

Le processus spécifique est :

1. Définir la classe de « Execution » pour réaliser l'interface « Runnable ».
2. Remplacer la méthode de « run() » dans l'interface « Runnable », le thread pour exécuter le code est stocké dans la méthode de « run() ».
3. Etablir une classe « Executor » pour définir une méthode de « execute(Runnable command) » qui peut réaliser « Runnable » de la classe de « Execution ». Dans le même temps, « Executor-Service » définit des méthodes telles que l'arrêt, l'envoi de tâches et le suivi des résultats de la tâche.
4. « newSingleThreadExecutor » sérialise toutes les tâches qui lui sont soumises et gère la file d'attente, en veillant à ce qu'une seule tâche soit exécutée à tout moment, ce qui évite de devoir

traiter les problèmes de synchronisation.

5. Après que le servlet obtienne la requête de créer une nouvelle exécution, Executor va appeler la méthode « execute(Runnable command) » et puis démarrer le thread et appeler la méthode « run() » de la sous-classe d'interface « Runnable ».

L'Executor utilise un pool de threads pour gérer les threads. Il peut réutiliser les threads qui ont été créés au lieu de devoir créer de nouveaux threads à chaque fois, ce qui économise une partie de la surcharge. Le pool de threads peut également gérer facilement la taille du thread et le nombre de threads en cours d'exécution.

Par conséquent, nous pouvons gérer l'exécution de l'algorithme heuristique ou l'exécution de l'algorithme exact séparément dans la méthode « run() » de l'interface « Runnable » et appeler la fonction « lancerExecutionHeuristique() » ou « lancerExecutionExacte() » déjà décrite ci-dessus.

5 Afficher les résultats de comparaison dans des tableaux ou diagrammes

Afin de comparer les résultats d'exécution pour deux algorithmes, nous devons tout d'abord traiter le fichier de sortie « Output.txt » dans le back-end, qui est d'extraire des informations utiles. Ceux-ci comprennent des collections graphiques, des identifiants des graphes, des informations sur les nœuds, des informations sur les arcs, des solutions optimales, le nombre d'instances, le temps CPU, les déviations et les résultats de Matching. Tous les résultats de la comparaison seront affichés sur la page Web sous la forme d'un diagramme ou d'un tableau. La méthode du back-end pour mettre en œuvre les fonctions ci-dessus a été introduite dans la [Section 3.3](#) de Module de « Exécution » et ne sera pas décrite ici.

Pour afficher le diagramme ou le tableau sur le front-end, on utilise « Google Charts » qui fournit un moyen idéal de visualiser les données sur notre page Web, car la galerie de graphiques fournit un grand nombre de types de graphiques prêts à l'emploi.

La manière la plus courante d'utiliser « Google Charts » est d'intégrer JavaScript simple dans notre page Web. Nous chargeons certaines bibliothèques « Google Chart », listons les données à tracer, sélectionnons les options pour personnaliser notre graphique, et créons finalement un objet graphique avec un identifiant que nous choisissons. Ensuite, plus tard dans la page Web, nous créons un « <div> » avec cet identifiant pour afficher ce « Google Chart ».

6 Réaliser la visualisation de Matching trouvé entre deux graphes

Cette fonction est développée à l'aide du plug-in « Echarts », qui lit les informations de la collection de graphique en cours d'exécution, puis on peut compléter la visualisation du graphe et l'appariement des graphes par une série de traitements de chaînes et de prétraitement de données à la fonction du front-end.

ECharts, une bibliothèque de graphiques Javascript pure, qui est compatible avec la plupart des navigateurs actuels (IE8/9/10/11, Chrome, Firefox, Safari, etc.), peut être recalculé par glisser-déposer et peut implémenter des fonctions d'analyse de données multidimensionnelles. Il peut fournir des graphiques de visualisation de données intuitifs, vifs, interactifs et hautement personnalisables. Pour les collections graphiques, j'espère obtenir une visualisation tridimensionnelle plus intuitive et plus vivante afin que l'utilisateur puisse voir clairement la correspondance entre les deux graphes. Par conséquent, j'ai choisi « Echarts » pour la réalisation de la fonction de visualisation de correspondance graphique.

6.1 La couche « Model » lit le contenu en texte brut du fichier « .dat »

Ecrire une fonction « `getGraphesData()` », qui sert à lire les informations ligne par ligne de chaque graphe sur le fichier « .dat », et appeler « `$execution.getGraphesData()` » sur la page JSP pour l'enregistrer dans un « input » cachée.

6.2 Lire la table correspondante des nœuds et des arcs générés

Selon la table générée par le résultat de correspondance du graphe, un graphe sélectionné par l'utilisateur à visualiser est obtenu, en même temps, un tableau de « `getGraphNum[]` » est généré et un tableau à traiter « `data_VC0[]` » de deux nœuds correspondants est enregistré.

6.3 Diviser le contenu du texte brut du fichier « .dat »

Selon différents types de collections graphiques (MUTA/PAH ou AUCUNNOM, des molécules chimiques avec des liaisons chimiques, ou des diagrammes avec des coordonnées (x, y)), différentes méthodes de séparation sont automatiquement mises en correspondance. « `GraphesData_V_V` » et « `GraphsData_E_E` » sont respectivement des tableaux tridimensionnels de nœuds et d'arêtes. Par conséquent, dans « `GraphsData_V_V` » de MUTA/PAH, l'ID et la liaison chimique de chaque point de chaque graphe sont stockés, et dans « `GraphsData_E_E` » de MUTA/PAH, le point de départ, le point final et le nombre de ses liaisons chimiques sont stockés pour chaque arc de chaque graphe. Et dans « `GraphsData_V_V` » de AUCUNNOM, l'ID, la coordonné de X et la coordonné de Y de chaque point de chaque graphe sont stockés, et dans « `GraphsData_E_E` » de AUCUNNOM, l'ID du point de départ, l'ID du point final, le nombre de lignes et l'angle sont stockés pour chaque arc de chaque graphe.

6.4 Lire la collection graphique sélectionnée par l'utilisateur

Rejoindre « `GraphsData_V_V` », « `GraphesData_E_E` » et « `getGraphNum[]` » pour obtenir le contenu textuel de deux graphes à associer actuellement sélectionnées par l'utilisateur, puis placer les deux graphiques dans les quatre tableaux. Dans le même temps, selon différents types de collection graphique (MUTA/PAH ou AUCUNNOM), le système correspond automatiquement à différentes méthodes de placement de données.

6.5 Pré-traitement des données

Afin d'améliorer la réutilisabilité d'une partie du code d'image, les tableaux obtenus dans le processus ci-dessus sont combinés en un tableau tridimensionnel, tout comme la structure de « `GraphsData_V_V` » et « `Graphs_Data_E_E` ». L'avantage de cela est qu'avec les opérations de traitement appropriées, nous pouvons obtenir un tableau à deux dimensions de nœuds de correspondance d'image d'affichage final.

6.6 Visualisation

Appeler « `Echarts` » pour afficher les deux graphes à correspondre sur la page Web et afficher la connexion des deux nœuds de correspondance des deux graphes dans un grand toile. Afin

d'améliorer la réutilisabilité du code, le code visuel est exécuté en utilisant une structure de bloc de code qui est similaire à une fonction, et tout le code peut être exécuté de manière cyclique. Une fois la visualisation terminée, toutes les images peuvent être agrandies, amoindries, déplacées, téléchargées et réinitialisées. Chaque nœud du graphique peut afficher les informations spécifiques du nœud et d'autres nœuds connectés via « hover ». Comme le diagramme de liaison chimique adopte le modèle de gravité, l'utilisateur peut également faire glisser ou ajuster librement la position du nœud.

Il existe deux types de collections de graphiques : Pas de coordonnées claires dans « .dat » ou Avoir des coordonnées claires « .dat ».

6.6.1 Pas de coordonnées claires : MUTA et PAH

Il n'a pas de position de point fixe et pas d'angle fixe, donc j'ai utilisé un modèle de gravité. Dans l'image unique, le centre de gravité est le centre du graphique par défaut, le paramètre de force de centroïde est égal à 0,2, la répulsion entre les points est égale à 5000/point et la force de répulsion de bord est 0-40.

Dans le mode d'appariement deux cartes, la toile est ajustée sur un double cœur et les deux centres de gravité sont situés respectivement aux centres des côtés gauche et droit, afin que les deux images ne soient pas tirées vers les parties invisibles. À ce stade, le paramètre de force centripète est 0.2, la répulsion entre les points est 10000/le nombre total de points, la force de répulsion du bord est 0-800, le poids de la figure est 295-300 (ce paramètre correspond automatiquement au nombre de liaisons chimiques). Ces paramètres peuvent garantir que les deux images ne seront pas exclues de la toile en raison d'une inter-image répulsive trop forte, et que les liaisons chimiques ne seront pas tirées en arc difforme en raison d'une répulsion intra-image trop forte.

6.6.2 Avoir des coordonnées claires : AUCUNNOM

Il fournit une position de coordonnées fixe du point, donc j'utilise le modèle commun au lieu du modèle de gravité. Il est à noter que dans le mode d'appariement deux cartes, les coordonnées x de la deuxième image sont toutes décalées vers la droite de la moitié de la taille de la toile et que la taille de l'arc sera transformée en radians après traitement.

6.7 Interaction de l'utilisateur

Ajouter une fonction « draw() » pour déterminer le type de la collection graphique. Après le type de la collection graphique qui satisfait à la condition, la fonction correspondante est appelée pour réaliser la visualisation. De même, si nous avons besoin de visualiser d'autres types de collections graphiques dans le système à l'avenir, nous devons seulement ajouter des fonctions connexes et les appeler ici dans la **Figure 13**, ce qui aidera les développeurs suivants à continuer d'améliorer le jugement ou d'ajouter des collections graphiques utiles.

Pour MUTA, il a des informations moléculaires chimiques complètes. Donc, en faisant la visualisation MUTA, j'ai montré autant que possible les noms des molécules chimiques, des liaisons chimiques, de la valence chimique et des doubles liaisons. Cependant, comme PAH, il n'y a pas de coordonnées exactes (x, y), donc à proprement parler, elles appartiennent à un type.

En outre, comme le montre **Figure 14** suivante, si le futur système doit être capable de comparer plusieurs graphiques en même temps, j'ai déjà laissé une interface, et les développeurs suivants n'ont besoin que d'ajouter un « case » ici.


```

function draw(iflag) {
  if (iflag == 0) {
    alert("Veuillez mettre à jour le formulaire en premier");
  } else {
    var getDat = document.getElementById('getGraphesData').value;
    if (getDat.indexOf('chem:') >= 0) {
      // alert("MUTA");
      document.getElementById("graph").style.display = "";
      getGraphesData();
    } else if (getDat.indexOf('x:') >= 0 && getDat.indexOf('y:') >= 0) {
      // alert("AUCUNNOM");
      document.getElementById("graph").style.display = "";
      getSimpleData();
    } else if (getDat.indexOf('chem:') < 0 && getDat.indexOf('x:') < 0 && getDat.indexOf('y:') < 0
      && getDat.indexOf('V,id') >= 0 && getDat.indexOf('E,from,to') >= 0){
      document.getElementById("graph").style.display = "";
      getNewChemData();
      // alert("PAH/Autres");
    }
    else {
      alert("Ne supporte pas la visualisation de tels types de collection")
    }
  }
}

```

Figure 13 – Ajouter d'autres types de collections graphiques

```

case 0:
  var graph_cplex1 = echarts.init(document.getElementById('graph_cplex1'));
  var data_V_length = data_V1.length;
  var graphName = graphNum[0];
  break;
case 1:
  var graph_cplex2 = echarts.init(document.getElementById('graph_cplex2'));
  data_V_length = data_V2.length;
  graphName = graphNum[1];
  break;
default:
  alert("Le nombre d'images par défaut est 2, veuillez ajouter un case.");

```

Figure 14 – Comparer plusieurs graphiques dans le futur système(peut-être)

E

Gestion de projet

1 Méthode de suivi de projet

La méthode de gestion de projet est la méthode Agile[WWW2], qui permet des livraisons incrémentales de fonctionnalité au travers de Sprints, des tâches pourront être testées et vérifiées directement pour une amélioration plus rapide et stable du système. Les sources du logiciel seront versionnées grâce à Github et le projet sera suivi grâce à Trello **Figure 1** qui nous permet de gérer l'avancement du projet (Documentation, Planning, Rendez-vous, suivi de bugs,...).



Figure 1 – Capture d'écran de mon Trello

2 Découpage du projet en tâches

Voici le schéma **Figure 2** permettant de montrer le découpage du projet en tâches.

Tâches

| Nom | Date de début | Date de fin |
|--|---------------|-------------|
| Recherche et spécification | 20/09/17 | 06/11/17 |
| L'étude du besoin | 20/09/17 | 26/09/17 |
| Documentation et lecture | 27/09/17 | 06/10/17 |
| Etat de l'art sur la Graph Edit Distance | 09/10/17 | 20/10/17 |
| Implémentation et Analyse du projet existant | 05/10/17 | 20/10/17 |
| Cahier de spécification | 23/10/17 | 06/11/17 |
| Analyse et Modélisation | 07/11/17 | 01/12/17 |
| Préparation du rapport S9 | 04/12/17 | 08/12/17 |
| Préparation de la soutenance S9 | 11/12/17 | 15/12/17 |
| Développement | 18/12/17 | 19/03/18 |
| Réalisation des fonctionnalités | 18/12/17 | 19/02/18 |
| Test de la programmation | 20/02/18 | 05/03/18 |
| Reprise du projet | 06/03/18 | 19/03/18 |
| Préparation du rapport S10 | 20/03/18 | 26/03/18 |
| Préparation de la soutenance S10 | 27/03/18 | 02/04/18 |
| Jalons & livrables | 20/09/17 | 02/04/18 |
| Début du projet | 20/09/17 | 20/09/17 |
| Cahier des charges | 27/09/17 | 27/09/17 |
| Cahier de spécification | 07/11/17 | 07/11/17 |
| Diagramme UML | 04/12/17 | 04/12/17 |
| Rapport et support S9 | 11/12/17 | 11/12/17 |
| Validation | 20/03/18 | 20/03/18 |
| Rapport et support S10 | 27/03/18 | 27/03/18 |
| Fin du Projet | 03/04/18 | 03/04/18 |

Figure 2 – Découpage du projet en tâches

2.1 Tâche 1 : L'étude du besoin

Description de la tâche :

Comprendre les objectifs, le contexte, les contraintes du projet. Faire l'étude de l'environnement et savoir les besoins de MOA. Déterminer les outils de gestion de projet, et préparer la première version du diagramme de Gantt.

Livrables :

Cahier des charges

2.2 Tâche 2 : Documentation et lecture

Description de la tâche :

Faire la recherche qui consiste à : deux catégories de graphe matching, le problème de Distance de l'édition du graphique (GED), Local Branching (LocBra) heuristique et ses caractéristiques principales, quelques méthodes heuristiques utiles, principalement y compris Beam search qui utilise une recherche large-première pour construire son arbre de recherche, SBPBeam qui combine Bipartite Graph matching et Beam search heuristique. Et les instances (PAH/MUTA) pour évaluer et comparer les algorithmes.

Livrables :

Une présentation pour expliquer à mon tuteur ce que je comprends.

2.3 Tâche 3 : Etat de l'art sur la Graph Edit Distance

Description de la tâche :

Faire la recherche qui consiste à : les modèles mathématiques (F1/F2/JH), la différence entre l'exacte et l'heuristique, Les critères de comparaison : le temps moyen, le temps maximal, la déviation avec la solution optimal, le nombre de solutions optimales en total et les expérimentations. Après que tout la recherche, faire la rédaction de l'état de l'art sur la Graph Edit Distance.

Livrables :

Etat de l'art sur la Graph Edit Distance

2.4 Tâche 4 : Implémentation et Analyse du projet existant

Description de la tâche :

Installer, déboguer et modifier le code de projet existant. Comprendre la structure générale du projet, faire la compréhension initiale et l'apprentissage du langage de programmation à utiliser.

2.5 Tâche 5 : Analyse et Modélisation

Description de la tâche :

Analyser les objectifs et faire la modélisation que le projet va accomplir, décrire les fonctions spécifiques que le projet doit mettre en œuvre et identifier les liens entre les différentes fonctions et composantes. Justifier les éléments et choix retenus lors de l'analyse par rapport aux contraintes et objectifs du projet. Si possible, les risques liés à la solution retenue sont soulignés avec une évaluation de l'impact potentiel.

Livrables :

Cahier de spécification, les diagrammes UML

2.6 Tâche 6 : Préparation du rapport S9 et soutenance S9

Description de la tâche :

Rédiger le rapport et préparer la soutenance selon les besoins d'enseignement de Polytech.

Livrables :

Rapport S9

2.7 Tâche 7 : Réalisation des fonctionnalités

Description de la tâche :

Selon le plan, je vais progressivement réaliser les fonctions du projet dont nous avons parlé dans le chapitre précédent, et utiliser l'outil de gestion de projet « Github » pour le management de

la version.

Livrables :

Documentation d'installation/déploiement, Codage et JavaDoc

2.8 Tâche 8 : Test de la programmation et reprise du projet

Description de la tâche :

Tester et déboguer le code déjà écrit, effectuer de nombreux tests sur l'implémentation afin de vérifier la fonctionnalité, et si nécessaire, faire l'amélioration.

Livrables :

Documentation d'utilisation, Dossiers de tests

2.9 Tâche 9 : Préparation du rapport S10 et soutenance S10

Description de la tâche :

Rédiger le rapport et préparer la soutenance selon les besoins d'enseignement de Polytech.

Livrables :

Rapport S10

3 Planning prévisionnel S9 et S10

Voici le diagramme de Gantt prévisionnel **Figure 3** permettant de montrer le planning prévisionnel pour le semestre 9 et 10.

- Recherche et spécification
Cette partie comprend principalement l'analyse des besoins, la lecture de documents, l'installation et le débogage de projets existants.
- Analyse et conception
Ce cycle se concentre sur la compréhension et l'analyse de la structure du système, faire la modélisation des fonctionnalités à ajouter et à modifier et faire la préparation de programmation du prochain semestre.
- Préparation du rapport S9
- Préparation de la soutenance S9
- Développement
Ce cycle contient trois parties : « Réalisation des fonctionnalités », « Test de la programmation » et « Reprise du projet ». Pour la partie de « Réalisation des fonctionnalités », les choses à réaliser ont déjà été définies par la précédente période de recherche. La partie test consiste à vérifier la qualité du code. La partie reprise sert principalement à ajuster ou optimiser le code en fonction des résultats du test.
- Préparation du rapport S10
- Préparation de la soutenance S10

En plus, Chaque étape a un livrable correspondant.

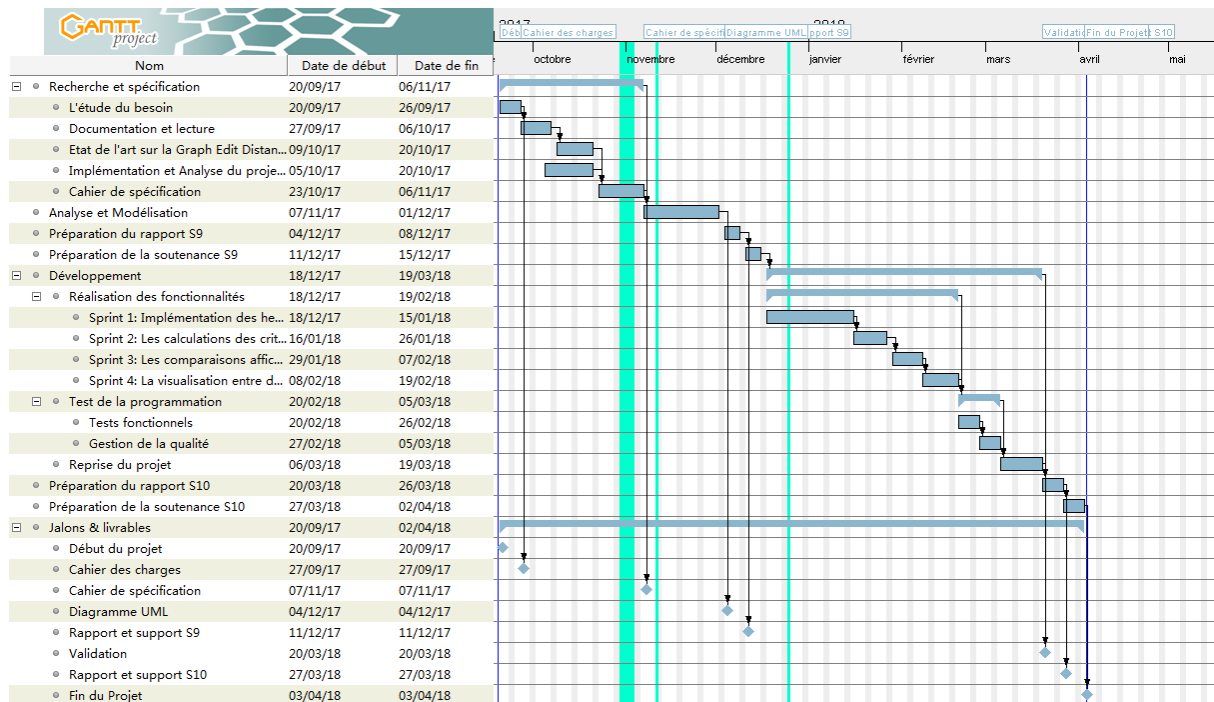


Figure 3 – Le diagramme de Gantt prévisionnel au début

4 Planning réel S9 et analyse

Voici le diagramme de Gantt réel S9 **Figure 4** permettant de montrer l'exécution de planning pour le S9.

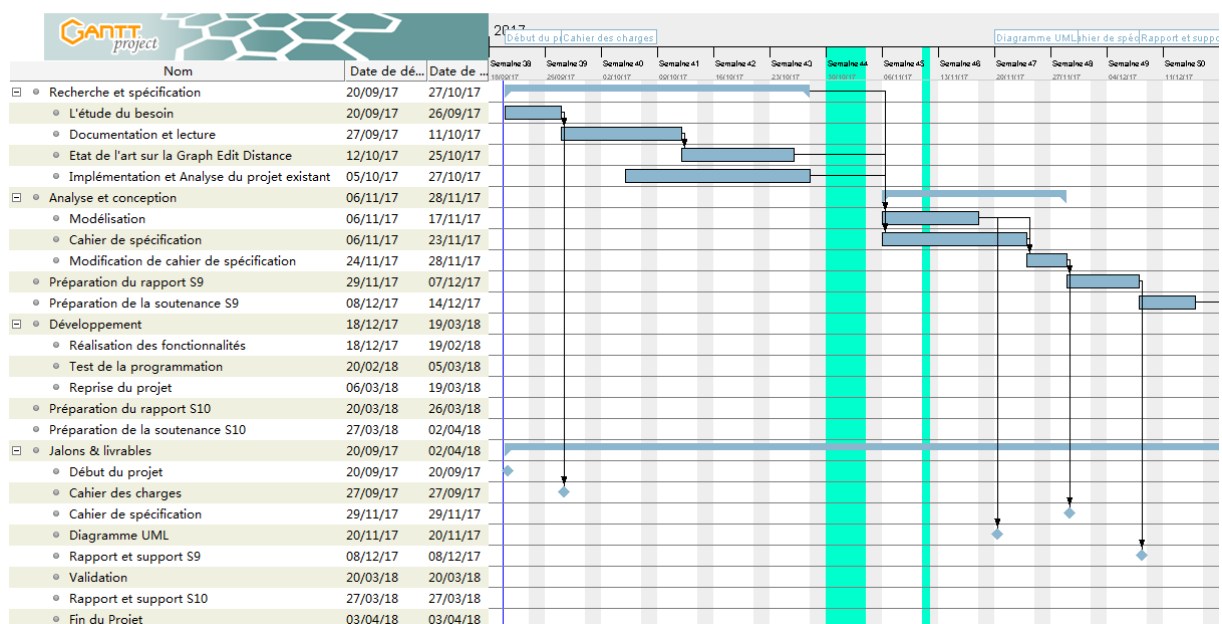


Figure 4 – Le diagramme de Gantt réel S9

Cette figure a quelques différences de temps et des retards par rapport au diagramme de Gantt prévisionnel décrit précédemment. Tout le retard est principalement dû à un manque de connaissances dans ce domaine, donc il est difficile d'évaluer le temps nécessaire à une tâche que je n'ai jamais réalisé. Les différences principales entre le diagramme de Gantt prévisionnel et réel contiennent :

- La tâche de Documentation et lecture a pris fin avec cinq jours de retard et a induit indirectement un retard de cinq jours dans l'achèvement de l'Etat de l'art sur la Graph Edit Distance. Les problèmes dans ce domaine sont nouveaux pour moi et plus complexes que je ne le pensais, et même si j'ai fait de mon mieux pour surmonter la barrière de la langue pendant un minimum de temps, je la retarde encore.
- La tâche d'Implémentation et Analyse du projet existant a été reportée d'une semaine parce que j'ai rencontré des bogues inattendus et passé un peu de temps à installer et déboguer.
- J'ai réarrangé la tâche d'écrire le cahier de spécification complètement après le festival de Toussaint, qui évidemment n'a pas suivi le plan original parce que je savais que deux professeurs professionnels allaient nous donner une explication spécifique pour les exigences, les contenus et le format du cahier de spécification.
- La modélisation s'est terminée quelques jours plus tôt que prévu, ce qui m'a donné plus de temps pour préparer le rapport.

5 Planning plus détaillé de développement S10

Pour la partie de « Réalisation des fonctionnalités », je vais me concentrer sur la façon de mettre en œuvre la fonctionnalité requise pour le projet. Conformément à la méthode de développement Agile, des sprints d'un nombre fixé de fonctionnalité seront à mettre en place. Une réunion avec MOA pourra être mise en place pour évaluer le résultat et voir si le prochain sprint doit être modifié, cela me permettra de réévaluer la demande à chaque fin de sprint. Donc je divise l'étape de « Réalisation des fonctionnalités » en quatre Sprints, comme le diagramme de Gantt prévisionnel **Figure 3** :

- Sprint 1 : Implémentation des heuristiques

Après quelques optimisations et modifications nécessaires du projet existant, je vais créer une classe « Parametre », qui associe à la classe « Methode », et cette classe a l'attribut de « Methode » qui représentent que ce paramètre appartient à cette méthode. Durant cette tâche, les déclarations nécessaires pour les méthodes heuristiques seront réalisées en back-end, ensuite je vais mettre en place à nouveau la base de données ainsi que modifier l'interface web. Enfin, je développerai les fonctionnalités d'importation de la méthode heuristique ainsi que d'transport de résultats dans le fichier local « .txt ».

- Sprint 2 : Les calculations des critères différents

Pour ce sprint, je vais principalement faire des calculs différents, et m'assurer que les résultats sont précis. Le calcul contient le nombre d'instances exécutées/échouées/à exécuter, les minimums, moyennes et maximums correspondant au critère, la déviation de la fonction objectif et l'appareillement des sommets et des arcs entre deux graphes.

- Sprint 3 : Les comparaisons affichées dans les tableaux

Pour ce sprint, je vais développer et améliorer la fonctionnalité de retourner le tableau HTML permettant l'affichage des appareillements utilisant des bibliothèques logicielles néces-

saïres, afin d'être en mesure d'atteindre le graphique affiché sur le web. D'ailleurs, durant cette tâche, le front-end et le back-end peuvent être reliés ensemble, le back-end peut recevoir la requête frontale, le front-end pour obtenir les données passées qui viennent du back-end, l'objectif de ce processus est d'optimiser la structure et le code Servlet.

- Sprint 4 : La visualisation entre deux graphes

Ce dernier sprint sert à visualiser les résultats correspondants entre les nœuds et les arcs des deux graphiques. J'ai actuellement une solution préliminaire, et cela a été discuté dans le quatrième chapitre. L'interface disponible après ce sprint sera donc complète.

6 Planning réel S10 et analyse

Voici le diagramme de Gantt réel S10 **Figure 5** permettant de montrer l'exécution de planning pour le S10.

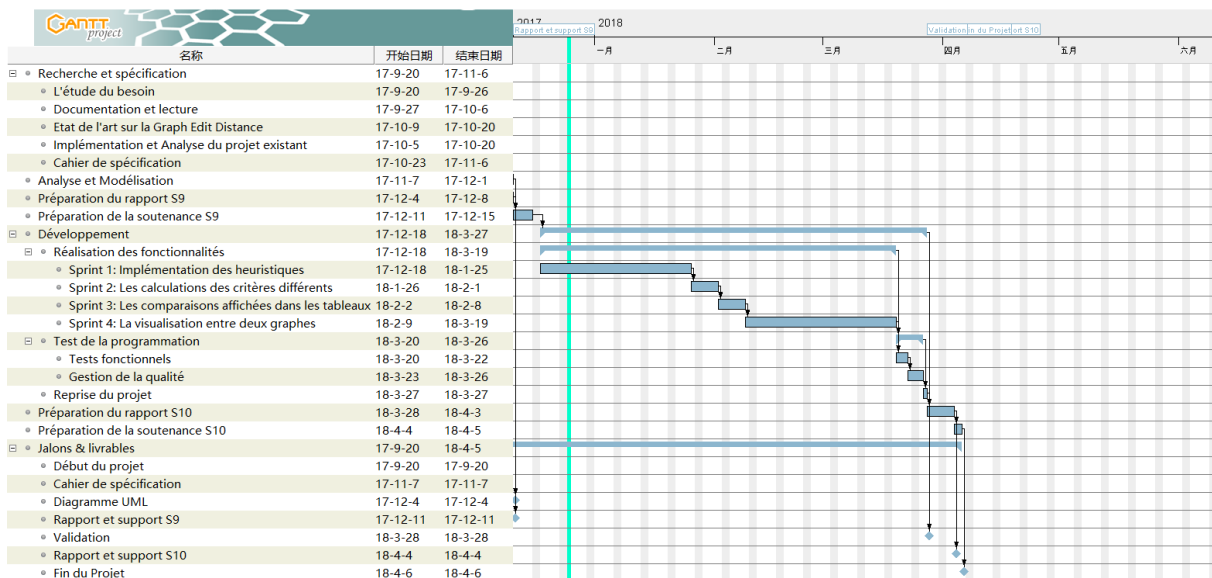


Figure 5 – Le diagramme de Gantt réel S10

Cette figure a quelques différences de temps et des retards par rapport au diagramme de Gantt prévisionnel décrit précédemment. Les différences principales entre le diagramme de Gantt prévisionnel et réel contiennent :

- Étant donné que la structure de la base de données modifie le temps d'installation de l'algorithme heuristique, je dois confirmer que la structure de la base de données peut répondre à tous les besoins de développement ultérieurs. Deuxièmement, lors de l'installation de l'algorithme heuristique, j'ai constamment trouvé des endroits irrationnels dans le système, donc je l'ai constamment optimisé pour améliorer l'efficacité de l'utilisateur.
- Le calcul des instances et des déviations a été réduit de 5 jours sur la durée prévue.
- Bien que la mise en œuvre de la visualisation a été distribuée le plus longtemps dans le déploiement du temps, cette partie soit encore plus difficile que prévu. Il m'a fallu beaucoup de temps pour trouver des plugins utiles pour traiter les données textuelles de l'image, et laisser la meilleure interface pour le développement ultérieur, afin de réaliser la meilleure correspondance et de faciliter la maintenance du projet.

- Le test fonctionnel du projet prend moins de temps que prévu, puisque de nombreux tests unitaires et tests d'intégration ont été réalisés au cours du processus de développement, cette partie du travail devient facile.
- Puisque la date limite pour le projet a été fixée, j'ai dû compresser le temps pour écrire le rapport.

Dans cette partie, je vais vous présenter le processus d'installation et de déploiement.

1 Installation locale

Le projet possède un dépôt sur Github. L'adresse pour y accéder est la suivante : https://github.com/littlecherisher/PRD_GraphMatching.git

Tout d'abord, assurez-vous que Java jdk et jre sont installés sur votre ordinateur et installez un environnement de programmation Java tel que Eclipse Java EE ou IntelliJ. N'oubliez pas de configurer les variables d'environnement sur votre ordinateur et configurer le chemin de compilation dans votre IDE.

Ensuite, vous devez télécharger et installer Apache Tomcat v9.0 en tant que serveur d'applications Web de votre projet afin de répondre à la requête d'accès de la page JSP. Prenez l'exemple de l'éclipse Java EE, vous pouvez directement ajouter Apache Tomcat v9.0 comme environnement d'exécution du serveur via la [Figure 1](#) suivante.

Après l'avoir ajouté avec succès, vous pouvez le choisir d'exécuter votre projet en tant que votre serveur. Le résultat après l'ajout réussi est comme indiqué ci-dessous [Figure 2](#) suivante. Actuellement le serveur a été démarré par moi.

Après avoir effectué toutes les opérations ci-dessus, vous pouvez importer notre projet. De plus, vous devez importer les packages « .jar » nécessaires et les ajouter dans « Java Build Path », que vous pouvez trouver dans le dossier du projet. Parmi eux, il convient de noter que le fichier « .jar » de la base de données H2, qui assure notre utilisation efficace de la base de données H2.

En outre, « ParseXML.exe » est utilisé dans le développement de projet comme un outil pour analyser XML, « Cplex1260.dll » est utilisé comme un outil de calcul interne pour l'algorithme. Ils doivent être importés dans le chemin déclaré dans le code du projet, ou modifier notre code en fonction du chemin de stockage des deux outils.

Les plugins ou bibliothèques utilisés dans le développement front-end (GoogleCharts, echarts, etc) sont déjà déclarés dans le code. Vous n'avez besoin d'aucune installation, mais vous devez savoir comment les utiliser pour continuer à développer notre projet.

Maintenant, vous pouvez exécuter notre projet à travers « Run on Server ». Mais pour rappel, il y a beaucoup de code et d'exécution liés au chemin de stockage local du projet. S'il existe une

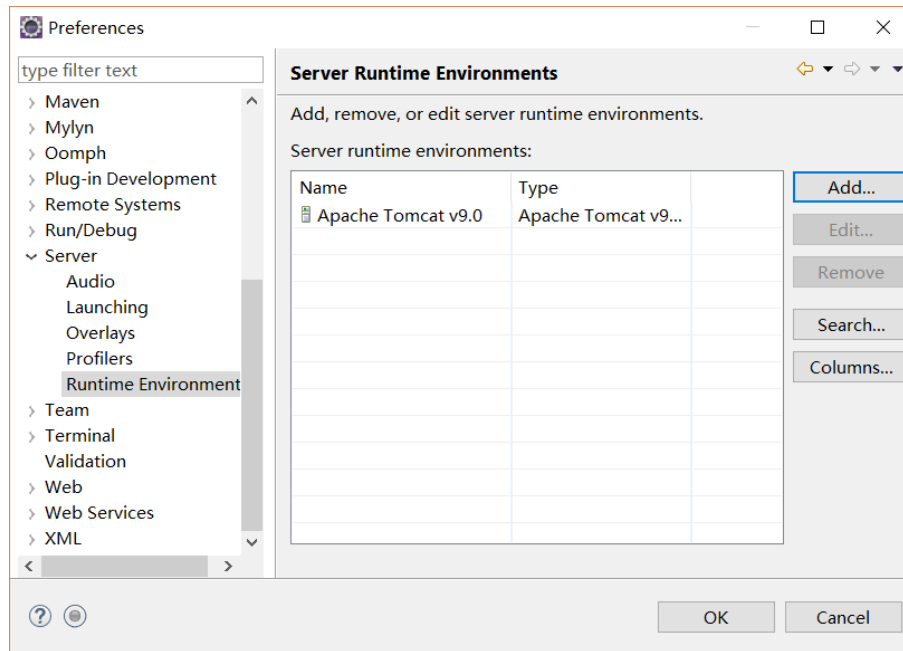


Figure 1 – Ajouter Apache Tomcat v9.0 comme environnement d'exécution du serveur

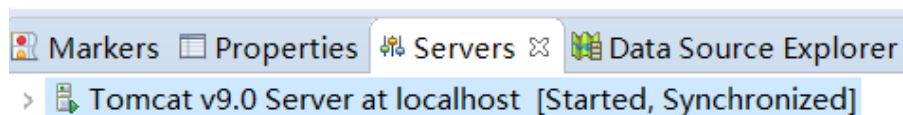


Figure 2 – Démarrer le serveur

erreur ou vous ne pouvez pas ouvrir la base de données, vérifiez le chemin du fichier et le path d'exécution de H2.

2 Déploiement sur le serveur

Une fois l'installation locale terminée, vous pouvez publier le projet sur le serveur.

Vous devez emporter le fichier « war » de ce projet, et puis le mettre dans le répertoire de « webapps » de « Apache Tomcat v9.0 », comme la Figure 3 ci-dessous.

Ensuite vous pouvez trouver « startup.bat » dans le répertoire de « bin », par lequel vous démarrez le serveur directement, également vous pouvez éteindre le serveur par « shutdown.bat ». Ce processus nous permet de lancer notre projet dans la navigateur.

Grâce à la Figure 4 suivante, vous pouvez voir que le serveur a été démarré avec succès.

Ouvrez un navigateur et entrez l'URL suivante : <http://localhost:8080/GraphMatching/>

Maintenant, vous pouvez utiliser notre système.

Si vous souhaitez accéder au contenu de la base de données, double-cliquez sur le fichier « .jar » dans la base de données H2. Lorsque la boîte de dialogue de la Figure 5 suivante apparaît, cliquez sur le bouton « connecter » pour vous connecter à la base de données.

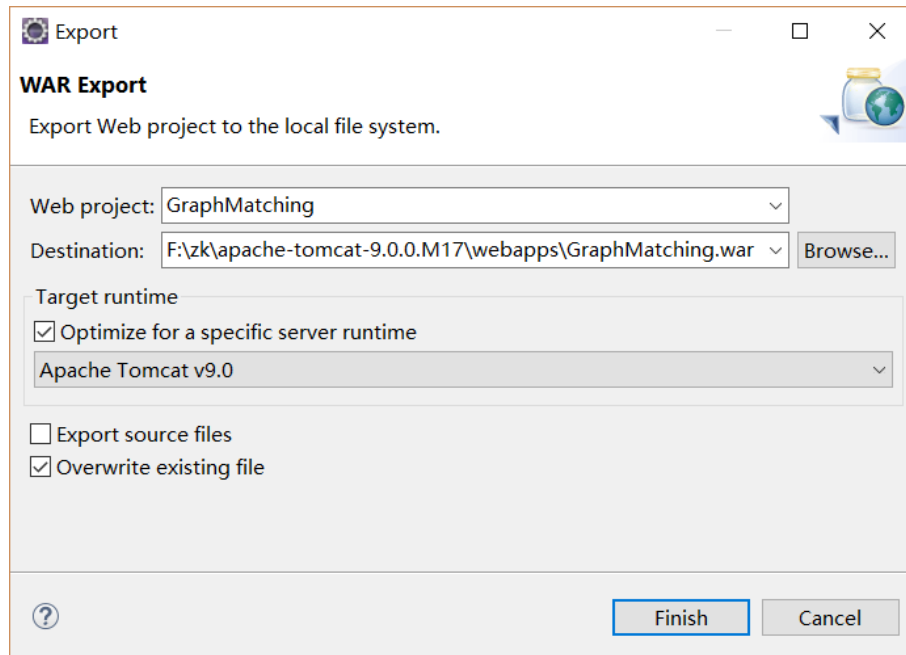


Figure 3 – Exporter le package de WAR

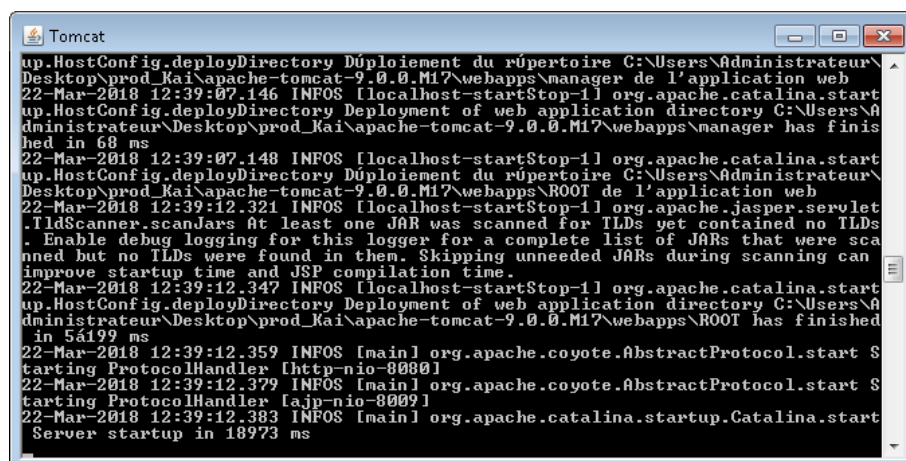


Figure 4 – Le serveur a été démarré avec succès

English ▼ Options Outils Aide

Connexion

Configuration enregistrée: Generic H2 (Embedded) ▼

Nom de configuration: Generic H2 (Embedded) Enregistrer Supprimer

Pilote JDBC: org.h2.Driver

URL JDBC: jdbc:h2:tcp://localhost/ProjetPRD/GraphMatching

Nom d'utilisateur:

Mot de passe:

Connecter Test de connexion

Figure 5 – *Connecter à la base de données H2*

Dans cette partie, je vais vous présenter le processus d'utilisation de notre système.

C'est la page d'accueil **Figure 1**. Il y a quatre options « Méthodes », « Datasets », « Tests » et « Exécution » qui contiennent essentiellement les fonctions principales du système, en cliquant sur le bouton de ces quatre options à réaliser.



Figure 1 – La page d'accueil

Pour l'option de « Méthodes », il y a trois choix y compris « Liste des méthodes », « Ajout d'une méthode exacte », « Ajout d'une méthode heuristique ». La première fois que l'utilisateur l'utilise, la liste des méthodes est vide. Donc vous devez ajouter une méthode exacte ou une méthode heuristique.

Quand vous ajoutez une méthode exacte, vous devez saisir son nom et importer un algorithme avec son format « .exe ». Attention, il convient de noter que l'algorithme importé doit être un fichier exécutable. Vous pouvez ajouter quelques descriptions de cet algorithme mais ce n'est pas obligatoire. Dans le coin inférieur droit de l'écran, vous pouvez choisir de continuer à ajouter des algorithmes ou à afficher une liste de méthodes ou d'autres opérations. Cette page est comme la **Figure 2** ci-dessous.

Quand vous ajoutez une méthode heuristique, toutes les opérations sont pareilles comme celle de méthode exacte sauf que vous devez ajouter ses paramètres avec le nom d'algorithme et le type d'algorithme (Vous devez sélectionner le type de paramètre en utilisant la liste déroulante), le nombre de paramètre n'est pas limité, mais il faut correspondre à cet algorithme que vous avez importer. Cette page est comme la **Figure 3** ci-dessous.

Pour les trois méthodes de IPFP / GNCCP / LocBra installées cette année, j'ai archivé le code

Méthodes Datasets Tests Exécutions

Ajout d'une méthode exacte :

Nom :

Executable :

Description :

Enregistrer
Liste des méthodes
+ Ajouter une méthode exacte
+ Ajouter une méthode heuristique

Figure 2 – Ajouter une méthode exacte

source des trois algorithmes sous le dossier « MILP » sur le bureau. Il est également nécessaire de spécifier les paramètres qui doivent être transmis dans l'algorithme.

- IPFP, deux paramètres : costType, nbIteration.
- GNCCP, deux paramètres : costType, d(la quantité à déduire de la variable ζ).
- LocBra, huit paramètres : costType, k, k_dv, total_time_limit, node_time_limit, l_max, dv_max, dv_cons_max.

L'entrée de tous les paramètres ci-dessus doit suivre l'ordre.

Ajout d'une méthode heuristique :

Méthodes Datasets Tests Exécutions

Ajout d'une méthode heuristique :

Nom :

Executable :

Description :

Paramètres nécessaires :
+ Ajouter

Nom : Type : Supprimer

Nom : Type : Supprimer

Enregistrer
Liste des méthodes
+ Ajouter une méthode exacte
+ Ajouter une méthode heuristique

Figure 3 – Ajouter une méthode heuristique

Maintenant vous pouvez retourner pour consulter la liste de méthode qui vous présente les algorithmes déjà ajoutés. C'est un table qui contient à quatre colonnes « Nom », « Description », « Type » et « Chemin exécutable », et vous pouvez rechercher un algorithme par nom/description. Dans cette page **Figure 4**, vous pouvez aussi continuer à ajouter un algorithme par les boutons au dessous.

D'ailleurs, après que vous avez ajouté un algorithme, le page va sauter à la page « Modification de la méthode » automatiquement pour faire quelques modifications, cette opération peut être également réalisée par double cliquer un algorithme dans la liste de méthode. Cette page affichera automatiquement la boîte de saisie du nom du paramètre, la liste déroulante du type de paramètre et les données historiques sous la méthode actuelle, comme la figure **Figure 5**.

Liste des méthodes

Rechercher :
Par nom / description :

Show entries

| Nom | Description | Type | Chemin exécutable |
|-------|-------------|-------------|--------------------------------------|
| F1 | | exacte | ProjetPRD\Methodes\2\IPPrepro_F1.exe |
| F2 | | exacte | ProjetPRD\Methodes\3\IPPrepro_F2.exe |
| GNCCP | | heuristique | ProjetPRD\Methodes\4\GNCCP.exe |
| IPFP | | heuristique | ProjetPRD\Methodes\1\IPFP.exe |

Previous 1 Next

Showing 1 to 4 of 4 entries

[+ Ajouter une méthode exacte](#)

[+ Ajouter une méthode heuristique](#)

Figure 4 – La liste de méthode

Modification de la méthode n° 1 :

Nom :

Executable :

[选择文件](#) 未选择任何文件

Description :

Paramètres nécessaires :

[+ Ajouter](#)

Nom : Type : [Supprimer](#)

Nom : Type : [Supprimer](#)

[Enregistrer](#)
[Masquer cette méthode](#)
[Liste des méthodes](#)
[+ Ajouter une méthode exacte](#)
[+ Ajouter une méthode heuristique](#)

Figure 5 – Modification de la méthode

Mais une fois vous avez ajouté un algorithme heuristique dans un test à exécuter et vous avez configuré ses valeurs de chaque paramètre, ces données sont stockées dans la base de données, c'est-à-dire que cet algorithme est un exécutable avec les valeurs particulières des paramètres et vous ne pouvez plus modifier cet algorithme heuristique. Si vous voulez le modifier, vous pouvez ajouter à nouveau un algorithme et saisir d'autres valeurs de paramètre quand vous ajoutez un test. Cette page est comme la [Figure 6](#) ci-dessous.

L'étape suivante est d'ajouter un DataSet/SubSet. La même chose que ajouter une méthode, vous devez saisir son nom et importer un DataSet/SubSet avec son format « .zip ». Attention, il convient de noter que le DataSet/SubSet importé doit être une collection de graphes. Vous pouvez aussi ajouter quelques descriptions mais ce n'est pas obligatoire. Dans le coin inférieur droit de l'écran, vous pouvez choisir de continuer à ajouter un DataSet/SubSet ou à afficher une liste de DataSet/SubSet. Cette page est comme la [Figure 7](#) ci-dessous.

Maintenant vous pouvez retourner pour consulter la liste de DataSet/SubSet qui vous présente les DataSets/SubSets déjà ajoutés. C'est un table qui contient à six colonnes « Nom », « Description », « Dirigé », « Nb graphes », « DataSet » et « SubSet », et vous pouvez rechercher

Modification de la méthode n° 1 :

Nom :

Executable :

Description :

Paramètres nécessaires :

Nom : Type :

Nom : Type :

Cette méthode a été ajoutée à la liste de test avec ses valeurs de paramètre, actuellement elle ne peut pas être modifiée

Liste des méthodes

+ Ajouter une méthode exacte

+ Ajouter une méthode heuristique

Figure 6 – Modification de la méthode non plus

Ajout d'une collection de graphes :

Nom :

Dataset : Dataset zip

Subset : Subset zip

Description :

Enregistrer

Liste des collections

+ Ajouter une collection

Figure 7 – Ajouter un DataSet/SubSet

un DataSet/SubSet par nom/description/type de graphe. Dans cette page, vous pouvez aussi continuer à ajouter un DataSet/SubSet par les boutons au dessous. Cette page est comme la Figure 8 ci-dessous.

Liste des collections de graphes

Rechercher :

Par nom / description : Par type de graphe :

Show entries

| Nom | Description | Dirigé | Nb graphes | Dataset | Subset |
|------|----------------------------------|------------|------------|------------------------------|-----------------------------|
| MUTA | La collection de graphes de MUTA | Non dirigé | 4337 | ProjetPRD\Datasets\2\Dataset | ProjetPRD\Datasets\2\Subset |
| PAH | La collection de graphes de PAH | Non dirigé | 1100 | ProjetPRD\Datasets\1\Dataset | ProjetPRD\Datasets\1\Subset |

Showing 1 to 2 of 2 entries

Previous Next

+ Ajouter une collection

Figure 8 – La liste de DataSet/SubSet

D'ailleurs, après que vous avez ajouté un DataSet/SubSet, le page va sauter à la page « Modification de la collection de graphes » automatiquement pour faire quelques modifications, cette

opération peut être également réalisée par double cliquer un DataSet/SubSet dans la liste de DataSet/SubSet.

L'étape suivante est d'ajouter un Test. Un test consiste en une comparaison de deux algorithmes exacts ou consiste en une comparaison de deux algorithmes heuristiques. Si vous choisissez deux types différents d'algorithme, la page va éjecter un avertissement comme le **Figure 9** ci-dessous.

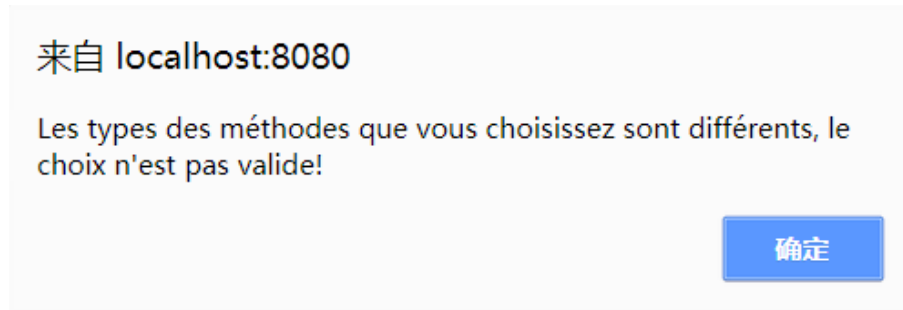


Figure 9 – Les choix de algo de type différent ne sont valides

Pour la comparaison de deux algorithmes exacts, c'est obligatoire de choisir au moins de deux algorithmes exacts, de choisir au moins d'une collection de graphes, de saisir les valeurs de paramètres pour ce test, d'indiquer le nom du test. Cette page est comme la **Figure 10** ci-dessous.

Ajout d'un test :

| <input type="checkbox"/> | Méthodes | Types |
|-------------------------------------|----------|-------------|
| <input type="checkbox"/> | IPFP | heuristique |
| <input checked="" type="checkbox"/> | F1 | exacte |
| <input checked="" type="checkbox"/> | F2 | exacte |
| <input type="checkbox"/> | GNCCP | heuristique |

Showing 1 to 4 of 4 entries 2 rows selected

| <input type="checkbox"/> | Datasets | Subsets |
|-------------------------------------|----------|--------------|
| <input type="checkbox"/> | PAH | mixed-graphs |
| <input checked="" type="checkbox"/> | PAH | train10 |
| <input type="checkbox"/> | PAH | train15 |
| <input type="checkbox"/> | PAH | train20 |
| <input type="checkbox"/> | PAH | train5 |
| <input type="checkbox"/> | MUTA | mixed-graphs |
| <input type="checkbox"/> | MUTA | train10 |

Showing 1 to 13 of 13 entries 1 row selected

Test

Nom : **Deux Exactes**

Description : La comparaison entre F1 et F2

Mode d'exécution : JP

Paramètres :

Mémoire limite (en Mo) :

Temps limite (en sec) :

Temps limite heuristique (en sec) :

Tolérance d'écart :

Nombre de threads :

Paramètres supplémentaires :

Figure 10 – Ajouter un test pour comparer les algo exactes

Pour la comparaison de deux algorithmes heuristiques, c'est obligatoire de choisir au moins de deux algorithmes heuristiques, de choisir au moins d'une collection de graphes, de saisir les valeurs de paramètres pour chaque algorithme heuristique, d'indiquer le nom du test. Cette page est comme la **Figure 11** ci-dessous.

La différence entre l'exacte et l'heuristique est la saisie de valeur de paramètre. Pour l'exacte, les paramètres sont indiqués pour ce test, mais pour l'heuristique, les paramètres sont pour chaque algorithme. Donc quand vous choisissez un algorithme heuristique, cette page va sauter une boîte de paramètres pour vous de saisir les valeurs de paramètres. En ce moment, ce n'est pas obligatoire d'entrer les paramètres « Mémoire limite (en Mo) » etc. Lorsque vous désélectionnez une méthode heuristique, la boîte de saisie des paramètres correspondante disparaît. Cette page est comme la **Figure 12** ci-dessous.

Maintenant vous pouvez retourner pour consulter la liste de Test qui vous présente les tests

Ajout d'un test :

☐ Méthodes

| | | |
|-------------------------------------|-------|-------------|
| <input checked="" type="checkbox"/> | IPFP | heuristique |
| <input type="checkbox"/> | F1 | exacte |
| <input type="checkbox"/> | F2 | exacte |
| <input checked="" type="checkbox"/> | GNCCP | heuristique |

Showing 1 to 4 of 4 entries 2 rows selected

☐ Datasets

| | | |
|-------------------------------------|------|--------------|
| <input type="checkbox"/> | PAH | mixed-graphs |
| <input type="checkbox"/> | PAH | train10 |
| <input type="checkbox"/> | PAH | train15 |
| <input type="checkbox"/> | PAH | train20 |
| <input type="checkbox"/> | PAH | train5 |
| <input type="checkbox"/> | MUTA | mixed-graphs |
| <input checked="" type="checkbox"/> | MUTA | train10 |
| <input type="checkbox"/> | MUTA | train20 |

Showing 1 to 13 of 13 entries 1 row selected

Test

Nom : Deux Heuristiques

Description : La comparaison entre IPFP et GNCCP

Mode d'exécution : IP

Figure 11 – Ajouter un test pour comparer les algos heuristiques

Veuillez entrer les valeurs des paramètres nécessaires pour IPFP :

| Nom | Type | Valeur |
|--------------|------|--------|
| costType | int | 2 |
| nbliteration | int | 20 |

Veuillez entrer les valeurs des paramètres nécessaires pour GNCCP :

| Nom | Type | Valeur |
|----------|------|--------|
| costType | int | 2 |
| d | int | 0.3 |

Figure 12 – La boîte de paramètres

déjà ajoutés. C'est un table qui contient à six colonnes « Nom », « Description », « Mode », « Nb exécution », « Méthode » et « DataSets », et vous pouvez rechercher un Test par nom/description/méthode/dataset. Dans cette page, vous pouvez aussi continuer à ajouter un Test par les boutons au dessous. Cette page est comme la [Figure 13](#) ci-dessous.

D'ailleurs, après que vous avez ajouté un Test, le page va sauter à la page « Modification du Test » automatiquement pour faire quelques modifications, cette opération peut être également réalisée

Liste des tests

Rechercher : Par nom / description : Par méthode Par dataset

Show entries

| Nom | Description | Mode | Nb executions | Méthode | Datasets |
|-------------------|------------------------------------|------|---------------|-------------|----------------|
| Deux Exactes | La comparaison entre F1 et F2 | IP | 0 | F1, F2 | train10 (PAH) |
| Deux Heuristiques | La comparaison entre IPFP et GNCCP | IP | 0 | IPFP, GNCCP | train10 (MUTA) |

Showing 1 to 2 of 2 entries

Previous Next

[+ Ajouter un test](#)

Figure 13 – La liste de Test

par double cliquer un Test dans la liste de Test. Cette page affiche également automatiquement toutes les données historiques sous le test en cours.

Maintenant vous pouvez exécuter un test. Lorsque vous cliquez sur un test spécial dans la table de test, une boîte de saisie qui s'appelle « Nouvelle exécution du test » apparaît au bas de la page pour vous permettre de saisir les détails de l'exécution, tels que le nom de l'exécution, la description, etc. Cette page est comme la Figure 14 ci-dessous.

| Nom | Description | Mode | Nb executions | Méthode | Datasets |
|-------------------|------------------------------------|------|---------------|-------------|----------------|
| Deux Exactes | La comparaison entre F1 et F2 | IP | 0 | F1, F2 | train10 (PAH) |
| Deux Heuristiques | La comparaison entre IPFP et GNCCP | IP | 0 | IPFP, GNCCP | train10 (MUTA) |

Showing 1 to 2 of 2 entries

Nouvelle exécution du test Deux Heuristiques :

Nom :

Description :

[Exécuter le test](#)

Figure 14 – Ajouter une Execution

Cliquez sur le bouton « Exécuter le test » et le système exécutera le test que vous avez sélectionné. Dans le même temps, la page va sauter à la page de la liste d'exécution, vous pouvez cliquer sur l'exécution pour voir les informations détaillées, telles que la progression de l'exécution, le statut de l'exécution, le nombre d'échecs, etc. Cette page est comme la Figure 15 ci-dessous.

En cliquant sur le bouton de « Résultats » ci-dessous, vous pouvez voir les résultats de l'exécution. Pour les comparaisons des algorithmes heuristiques, il y a cinq panneaux qui contiennent à « Général », « Temps CPU », « Instances traitées », « Déviation » et « Appariement ».

- « Général », comme Figure 16, afficher les informations de test correspondant cette exécution.
- « Temps CPU », comme Figure 17, indiquer le temps de CPU dépensé pour chaque méthode, y compris le temps minimum, le temps moyen et le temps maximal, en façon de table.
- « Instances traitées », comme Figure 18, présenter le nombre des instances traitées par chaque méthode en façon de table.
- « Déviation », comme Figure 19, montrer la déviation de la fonction objectif pour chaque méthode, y compris la déviation minimum, la déviation moyenne et la déviation maximale, en façon de table.

Liste des exécutions

Rechercher :
 Par nom / description : Par test : Par état :

Show entries

| Nom | Description | Test | Nb instances | Etat |
|-----|-------------------|------|--------------|----------|
| 1 | Deux Heuristiques | | 0/200 | En cours |

Showing 1 to 1 of 1 entries

Détails de l'exécution 1 :
 Exécutions : 106 / 200 53.0 %
 État : En cours
 0 echec(s)
 Résultats
 Masquer cette exécution

Figure 15 – Exécuter un test

Résultats de l'exécution 1 :

Général Temps CPU Instances traitées Déviation Appareillements

- Mode : IP
- 196 / 200 instances
- Méthodes : IPFP, GNCCP
- Datasets : train10 (MUTA)
- Paramètres :
 - nbliteration : 20
 - d : 0.3
 - costType : 2

Figure 16 – « Général »



Figure 17 – « Temps CPU »

- « Appareillement », comme Figure 20, permet de faire la visualisation de chaque paire de graphes. Tout d'abord, vous devez choisir la méthode que vous voulez utilisée, et puis vous pouvez sélectionner les graphes à comparer. Ensuite cliquez le button « Actualiser » pour regarder la table de correspondance, soit le résultat de correspondance de nœuds, soit le résultat d'arcs ou les deux.

Maintenant vous pouvez voir la visualisation de chaque graphe et le résultat de Graph Matching entre deux graphes, en cliquant le bouton « Visualiser ». Attention, c'est obligatoire de d'abord

Résultats de l'exécution 1 :

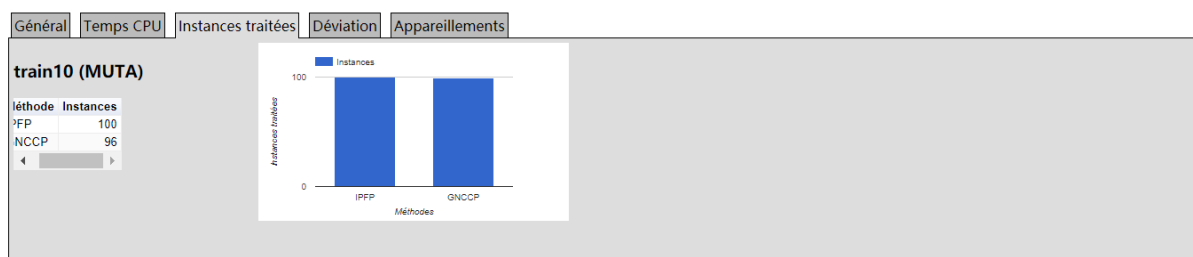


Figure 18 – « Instances traitées »



Figure 19 – « Déviation »

train10 (MUTA)

| Noeuds | | Arcs | |
|---------------|---------------|---------------|---------------|
| molecule_3601 | molecule_3875 | molecule_3601 | molecule_3875 |
| 1 | 6 | 1,2 | 3,6 |
| 2 | 3 | 1,3 | 5,6 |
| 3 | 5 | 1,4 | 6,10 |
| 4 | 10 | 2,5 | 1,3 |
| 5 | 1 | 2,6 | 3,9 |
| 6 | 9 | 3,7 | 2,5 |
| 7 | 2 | 3,8 | 5,7 |
| 8 | 7 | 5,7 | 1,2 |
| 9 | 4 | 5,9 | 1,4 |
| 10 | 8 | 7,10 | 2,8 |

Showing 1 to 10 of 10 entries

Showing 1 to 10 of 10 entries

Figure 20 – « Appariement »

cliquer le bouton « Actualiser » pour obtenir la correspondance de table, sinon une dialogue de « Veuillez mettre à jour le formulaire en premier » sera présentée. Par exemple, si vous voulez savoir la correspondance entre le graphe « molecule_3601 » et « molecule_3875 », la visualisation et la correspondance sont comme les figures **Figure 21** et **Figure 22** ci-dessous. Ces graphes sont dynamiques, peuvent être traînés, bougés, téléchargés et actualisés.

molecule_3601



molecule_3875



Figure 21 – « molecule_3601 » et « molecule_3875 »

La visualisation de GrapheMatching

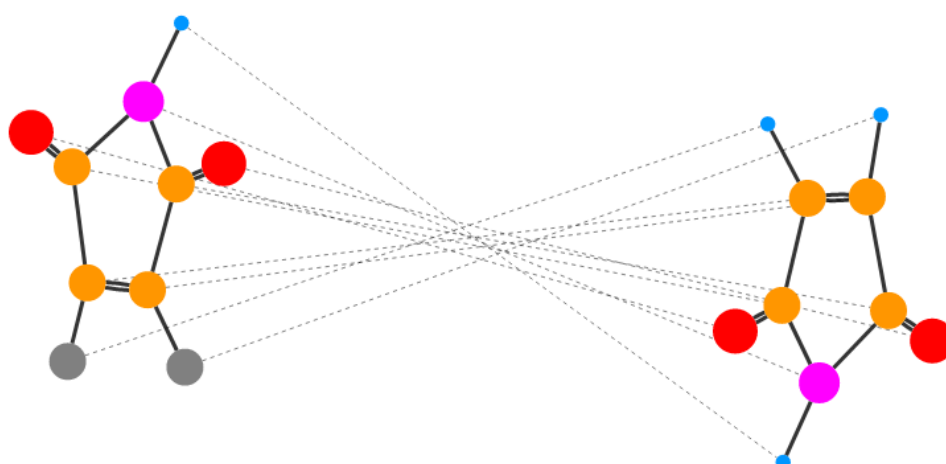


Figure 22 – La visualisation de Graph Matching

H

Dossiers de tests

1 Plan de test

Ce plan de test recense les objectifs et les moyens pour réaliser les tests. Dans le processus de développement du système, j'ai effectué des tests de petite à grande, dans l'ordre des tests unitaires, des tests d'intégration et des tests fonctionnels. Il y a à la fois des connexions et des différences entre eux.

1.1 Test unitaire

Les tests unitaires se rapportent à l'inspection et à la vérification de la plus petite unité testable du logiciel. L'unité en Java fait référence à une classe. Les unités individuelles du logiciel seront testées isolément du reste du programme.

1.2 Test d'intégration

Les tests d'intégration sont principalement utilisés pour tester l'interface entre plusieurs modules, généralement utilisée pour intégrer du code dans une équipe. Pour les petits systèmes, les tests d'intégration peuvent être intégrés dans les tests fonctionnels. Pour ce système, les modules montrent principalement l'interaction et la connexion entre back-end et front-end. J'ai testé les « servlet » de l'unité de transport principale entre les deux modules.

1.3 Test fonctionnels

Les tests fonctionnels consistent à tester si la fonction mentionnée dans le « cahier de spécifications » est omise et si elle est implémentée correctement.

2 Tests unitaires avec Junit

Chaque classe « JavaBean » de la couche « Model » est testée à l'unité par « Junit ». Les idées de test sont très similaires : en prenant la classe « Parametre » comme exemple, le code de test contient principalement trois éléments : « setUp() », « setDown() » et les methods à tester.

« setUp() » : Il est exécuté avant chaque méthode de test, il est utilisé pour obtenir les objets dans la base de données et établir les constructeurs associés.

« setDown() » : Il est exécuté après chaque méthode de test pour supprimer les objets temporaires utilisés par les tests.

Les methods à tester : Tirer parti de la méthode assert/fail de « Junit » et utiliser « assertEquals() » pour déterminer si la valeur de sortie réelle est égale à la valeur de sortie attendue.

La **Figure 1** ci-dessous montre que la classe « Parametre » a réussi le test unitaire.

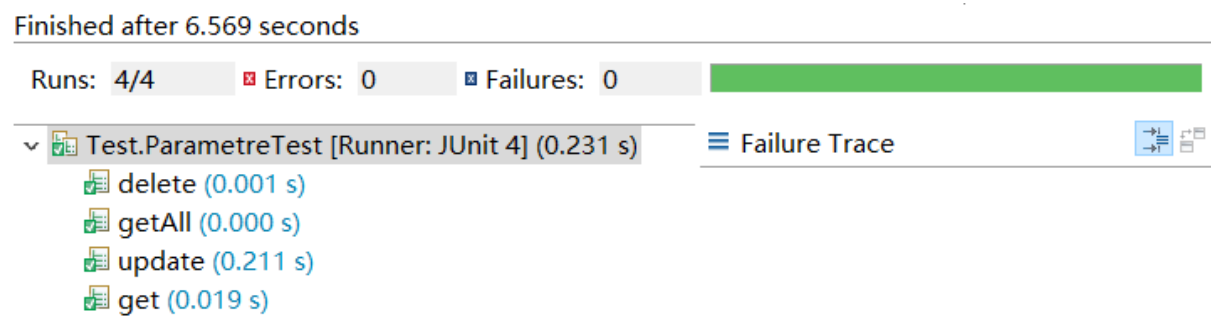


Figure 1 – Le test unitaire de classe Parametre

De même, pour les autres classes JavaBean de la couche « Model », j'utilise cette méthode pour les autres tests unitaires(**Figure 2**), et les résultats sont sans erreur.

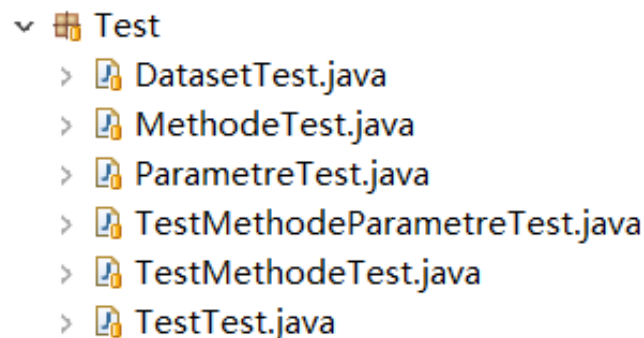


Figure 2 – Les tests unitaires de « Model »

3 Tests d'intégration

Dans ce projet, le « servlet » sert à la fois d'interfaces entre back-end et front-end et il est responsable de la livraison des données.

La méthode de test est simple, c'est-à-dire qu'après le démarrage du serveur, entrer directement l'adresse Web qui est déclarée dans le servlet sertissant à la connexion front-end et back-end. Si les données sont affichées ou livrées avec succès, la connexion entre les modules est réussie. C'est-à-dire on peut le tester par saisir « localhost : 8080 + nom du projet + nom du servlet ». Prenez la liste des méthodes dans le système à titre d'exemple, il suffit d'entrer « http://localhost:8080/GraphMatching/ListeMethode »

4 Tests fonctionnels

Dans cette section, on décrit les tests fonctionnels avec le cas d'utilisation.

| Activité d'utilisateur | Description | Résultats attendus | Etat du test |
|---|---|--|--|
| Ajouter un algorithme heuristique | Cliquez sur le bouton, le système saute à la page pour entrer les informations sur l'algorithme à ajouter. | La page affiche la zone de saisie pour le nom, la description, l'exécutable et les paramètres de l'algorithme heuristique. | OK |
| Modifier un algorithme heuristique | Double-cliquez sur l'algorithme dans la liste et le système saute à la page de modification. | La page affiche des informations historiques que l'utilisateur peut modifier et renvoyer. | OK |
| Ajouter un test | Cliquez sur le bouton, le système saute à la page pour entrer les informations sur le test à ajouter. | La page affiche la zone de saisie pour toutes les informations sur ce test à ajouter. | OK |
| Modifier un test | Double-cliquez sur le test dans la liste et le système saute à la page de modification. | La page affiche des informations historiques que l'utilisateur peut modifier et renvoyer. | OK |
| Entrer la valeur du paramètre pour l'algorithme heuristique | Lorsque l'utilisateur sélectionne un algorithme heuristique pour rejoindre test, il doit d'abord entrer la valeur du paramètre de l'algorithme. | Une fois l'heuristique sélectionnée, la page affiche automatiquement la zone de saisie de la valeur du paramètre. | OK |
| Désélectionner l'algorithme heuristique | Les utilisateurs ne sélectionnent plus un algorithme heuristique à ajouter au test. | La zone de saisie de la valeur du paramètre disparaît. | OK |
| Sélectionne plusieurs types de méthodes pour les comparer | Dans un test, l'utilisateur a choisi à la fois l'algorithme exact et l'algorithme heuristique. | Le système affiche automatiquement "Impossible de continuer". | OK |
| Sélectionne une collection de graphes pour comparer | Ajouter la collection de graphes au test. | Sélectionner au moins une collection de graphes pour exécuter une algorithme. | Le système supporte actuellement deux types d'collections de graphes, des diagrammes avec des coordonnées (x, y) ou pas. |

| Activité d'utilisateur | Description | Résultats attendus | Etat du test |
|--|---|--|--------------|
| Cliquer sur le bouton pour enregistrer le test | Enregistrer le test en cours pour l'exécution. | Enregistrer le test ajouté de l'utilisateur et affichez-le dans la liste. | OK |
| Ajout d'un exécution | Exécuter un test pour obtenir le résultat de la comparaison des algorithmes. | La page affiche les informations d'exécution, la progression de l'exécution. | OK |
| Voir le résultat d'exécution | Consulter les détails du résultat, tels que le temps CPU, les instances traitées et les déviations. | Entrer dans la page des résultats pour montrer la comparaison des algorithmes. | OK |
| Appareillement | Sélectionner la méthode et deux graphes à comparer pour afficher les résultats correspondants. | La page affiche la table correspondante. | OK |
| Visualisation | Visualiser les deux graphes et montrer la connexion dynamique. | La connexion entre les nœuds des deux graphiques peut être visualisée. | OK |

Tous les cas d'utilisation des tests fonctionnels ci-dessus ont été passés en revue conformément aux exigences de « cahier de spécifications », c'est-à-dire que le système peut maintenant être livré aux utilisateurs.



Webographie

- [WWW1] *Algorithme hongrois*. URL : https://fr.wikipedia.org/wiki/Algorithme_hongrois.
- [WWW2] *Méthode agile*. URL : https://fr.wikipedia.org/wiki/M%C3%A9thode_agile.

Bibliographie

- [1] Zeina ABU-AISHEH, Romain RAVEAUX et Jean-Yves RAMEL. *A graph database repository and performance evaluation metrics for graph edit distance*. In : *International Workshop on Graph-Based Representations in Pattern Recognition*. 2015.
- [2] Sébastien BOUGLEUX, Luc BRUN, Vincenzo CARLETTI, Pasquale FOGGIA, Benoit GAÜZERE et Mario VENTO. « A quadratic assignment formulation of the graph edit distance ». In : *arXiv preprint arXiv :1512.07494* (2015).
- [3] Sébastien BOUGLEUX, Luc BRUN, Vincenzo CARLETTI, Pasquale FOGGIA, Benoit GAÜZÈRE et Mario VENTO. « Graph edit distance as a quadratic assignment problem ». In : *Pattern Recognition Letters* (2017).
- [4] Horst BUNKE. *Graph matching : Theoretical foundations, algorithms, and applications*. In : *Proc. Vision Interface*. 2000.
- [5] Donatello CONTE, Pasquale FOGGIA, Carlo SANSONE et Mario VENTO. « Thirty years of graph matching in pattern recognition ». In : *International journal of pattern recognition and artificial intelligence* (2004).
- [6] Mostafa DARWICHE, Donatello CONTE, Romain RAVEAUX et Vincent T'KINDT. *A Local Branching Heuristic for the Graph Edit Distance Problem*. In : 2017.
- [7] Miquel FERRER, Francesc SERRATOSA et Kaspar RIESEN. « Improving bipartite graph matching by assessing the assignment confidence ». In : *Pattern Recognition Letters* (2015).
- [8] Julien LEROUGE, Zeina ABU-AISHEH, Romain RAVEAUX, Pierre HÉROUX et Sébastien ADAM. « Graph edit distance : a new binary linear programming formulation ». In : *arXiv preprint arXiv :1505.05740* (2015).
- [9] Michel NEUHAUS, Kaspar RIESEN et Horst BUNKE. *Fast suboptimal algorithms for the computation of graph edit distance*. In : *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*. 2006.
- [10] Kaspar RIESEN, Michel NEUHAUS et Horst BUNKE. « Bipartite graph matching for computing the edit distance of graphs ». In : *GbrPR* (2007).

Framework interactif pour tester les algorithmes de Graph Matching

Kai ZENG

Encadrement : Mostafa Darwiche, Romain Raveaux, Donatello Conte et Vincent T'kindt

Objectifs

L'objectif du projet est de mettre en œuvre un système pour l'évaluation des méthodes qui résolvent les problèmes de Graph Matching (comme le GED).

- Améliorer les fonctionnalités.
- Implémenter des heuristiques.
- Évaluer ces méthodes différentes.

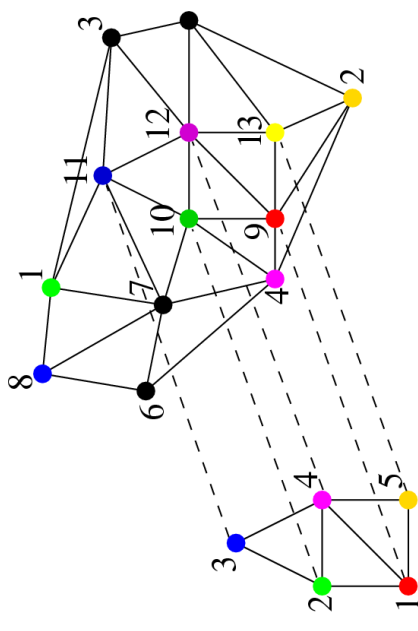
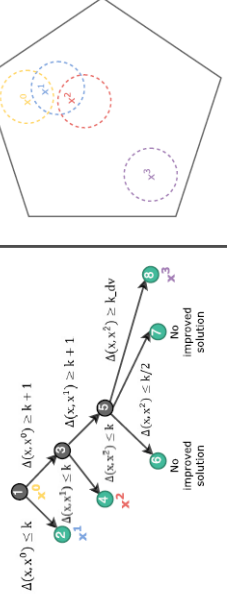
Mise en œuvre

Le projet est implémenté en Java, JSP, JavaScript intégrant plusieurs librairies ex. Google Charts. Utiliser CPLEX Optimizer pour calculer l'algorithme et le modèle. Afin d'interagir avec la base de donnée le site web fera appel à un serveur de base de données embarqué H2. Le Framework Hibernate est utilisé afin de faciliter le développement de la base de données.

Résultats attendus

- Exécuter les tests correctement
- Afficher les résultats dans des tableaux ou diagrammes
- L'analyse pour les expérimentations
- La visualisation de Matching

Heuristique



Framework interactif pour tester les algorithmes de Graph Matching

Résumé

Les problèmes de Graph Matching ou appariement de graphes sont des problèmes connus, intéressants et difficile à résoudre. Ils permettent de comparer des graphes entre eux et mesurer la similarité ou dis-similarité. Étant donné que les graphes sont considérés comme une bonne manière pour représenter des objets, la tâche de comparer ces graphes est donc un outil pour comparer les objets qu'ils représentent. On rencontre les problèmes de Graph Matching dans des applications comme : la reconnaissance de l'écriture manuscrite, la détection des objets dans les images, la détection des logiciels malveillants, etc.

Parmi les problèmes de Graph Matching, un des plus abordés est le Graph Edit Distance, où on s'intéresse à trouver la distance minimale entre deux graphes. C'est un problème NP-difficile, donc trouver un algorithme qui résout le problème et donne les meilleures solutions dans un temps acceptable n'est pas une tâche facile. Ce problème est traité par beaucoup des méthodes de type heuristiques ou exact comme la programmation linéaire en nombres entiers. Vu le nombre de méthodes croissant, il manque un système qui permet de les comparer entre eux, afin d'étudier leurs performances et précision.

Donc, l'objectif du projet est de mettre en œuvre un système pour l'évaluation des méthodes qui résolvent les problèmes de Graph Matching, soit des modèles linéaires, non linéaires ou des heuristiques.

Mots-clés

Graph Matching, Graph Edit Distance, heuristique

Abstract

Graph Matching or graph matching problems are known, interesting and difficult to solve. They make it possible to compare graphs with each other and to measure the similarity or dissimilarity. Since graphs are considered a good way to represent objects, the task of comparing these graphs is therefore a tool to compare the objects they represent. Graph Matching problems are encountered in applications such as handwriting recognition, object detection in images, malware detection, and more.

Among Graph Matching's problems, one of the most discussed is the Graph Edit Distance, where we are interested in finding the minimum distance between two graphs. This is an NP-hard problem, so finding an algorithm that solves the problem and gives the best solutions in an acceptable time is not an easy task. This problem is addressed by many heuristic or exact type methods such as integer linear programming. Given the number of methods growing, it lacks a system that allows to compare them to each other, to study their performance and accuracy.

So, the goal of the project is to implement a system for evaluating methods that solve Graph Matching problems, either linear, non-linear models or heuristics.

Keywords

Graph Matching, Graph Edit Distance, heuristic

Tuteurs académiques

Mostafa DARWICHE

Romain RAVEAUX

Donatello CONTE

Vincent T'KINDT

Étudiant

Kai ZENG (DI5)